Article

Novel Swarm Intelligence Algorithm for Global Optimization and Multi-UAVs Cooperative Path Planning: Anas Platyrhynchos Optimizer

Yong Zhang ^{1,2,*}, Pengfei Wang ^{3,*} , Liuqing Yang ^{1,2}, Yanbin Liu ⁴, Yuping Lu ³
and Xiaokang Zhu ^{1,2}

¹ Research Institute of Pilotless Aircraft, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; yangliuqing@nuaa.edu.cn (L.Y.); zxknuaa@163.com (X.Z.)

² Key Laboratory of Unmanned Aerial Vehicle Technology, Ministry of Industry and Information Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

³ College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; yplac@nuaa.edu.cn

⁴ College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; liuyb@nuaa.edu.cn

* Correspondence: yongzhang@nuaa.edu.cn (Y.Z.); wangpf@nuaa.edu.cn (P.W.)

Received: 8 June 2020; Accepted: 8 July 2020; Published: 14 July 2020

Abstract: In this study, a novel type of swarm intelligence algorithm referred as the anas platyrhynchos optimizer is proposed by simulating the cluster action of the anas platyrhynchos. Starting from the core of swarm intelligence algorithm, on the premise of the use of few parameters and ease in implementation, the mathematical model and algorithm flow of the anas platyrhynchos optimizer are given, and the balance between global search and local development in the algorithm is ensured. The algorithm was applied to a benchmark function and a cooperative path planning solution for multi-UAVs as a means of testing the performance of the algorithm. The optimization results showed that the anas platyrhynchos optimizer is more superior in solving optimization problems compared with the mainstream intelligent algorithm. This study provides a new idea for solving more engineering problems.

Keywords: swarm intelligence algorithm; anas platyrhynchos optimizer; global optimization; multi-UAVs cooperation; path planning

1. Introduction

The swarm intelligence algorithm is an important optimization method. This type of algorithm is highly suitable for problems that are difficult to be solved effectively or even cannot be solved by traditional technology, as establishing an effective formal model, such as NP-hard problems, is difficult to accomplish. The traveling salesman problem (TSP) is one of the most classical NP-hard problems. In TSP, a traveling salesman is expected to identify the shortest path to visit n cities in a country. Suppose that the traveling salesman arrives at a country with only 21 cities and visits the cities in any order (from one city and moves directly to another city), the size of the search space in this problem can be calculated as $(n - 1)!$. This formulation suggests that the TSP with $n = 21$ entails 2.432902×10^{18} possible travel searches. If a person takes one second to calculate a single travel, then 770 million years are needed to search the whole search space. If a computer with a 2.0 GHz CPU (i.e., 2 billion instructions per second) is used, then 38 years are required to identify all possible paths. If a city is added to this problem, then the time is increased accordingly to nearly 810 years. However, Reference [1] has shown that the swarm intelligence algorithm is outstanding in solving

these problems. Consequently, the swarm intelligence algorithm has become widely popular among researchers in recent years.

The swarm intelligence algorithm is inspired by the behavior of social creatures such as ants, termites, birds, and fish. The most popular swarm intelligence algorithm at the beginning of algorithm development is particle swarm optimization (PSO). The PSO algorithm [2] was proposed by Kennedy and Eberhart, and it was inspired by the social behavior of birds. With the popularization and development of the swarm intelligence concept, a variety of swarm intelligence algorithms have been proposed in succession, namely, ant colony optimization (ACO) [3], bat algorithm (BA) [4], dragonfly optimization [5], grasshopper optimization algorithm (GOA) [6], moth flame optimization algorithm (MFOA) [7], gray wolf optimization (GWO) [8], and whale optimization algorithm (WOA) [9], all of which have been widely used in recent years. Nature is the infinite source of inspiration in computational science, and more swarm intelligence algorithms are expected in the future.

Swarm intelligence algorithms have been widely applied to various practical engineering issues in recent years. Whale optimization algorithm, which was proposed in 2016, has been cited more than 1600 times as of April 2020 based on Google Scholar indicators and has been widely used in different fields such as energy [10,11], computer networks [12,13], cloud computing [14], image processing and machine vision [15,16], electronics and electrical engineering [17,18], antenna design [19], feature selection [20–22], aerospace [23], path planning [24], and structural optimization [25,26]. Moreover, GOA has been successfully applied to the trajectory optimization of multiple solar-powered UAVs [27], short-term load forecasting [28], electrical characterization of proton exchange membrane fuel cell stacks [29], data clustering [30], scheduling of thermal system [31], analyzing vibration signals from rotating machinery [32], short-term wind electric power forecasting [33], optional distribution system reconfiguration and distributed generation placement [34], classification for biomedical purpose [35], design of linear antenna arrays [36], automatic seizure detection of EEG signals [37], designing automatic voltage regulator systems [38], training neural network [39], and other issues. The multiverse optimizer (MVO) algorithm has also been widely used in different fields. In References [40,41], MVOs were used to train multilayer perceptron neural networks. Benmessahel [42] combined the MVO and the artificial neural network to develop an intrusion detection system. In Reference [43], the MVO was used to optimize the parameters of support vector machines. Rosenberg [44] solved the clustering problem by using the search ability of the MVO. Thus far, the studies conducted indicate that swarm intelligence algorithms can effectively solve many practical optimization problems.

The no free lunch (NFL) theorem [45] has proven logically that swarm intelligence algorithms cannot be regarded the most suitable solution for all optimization problems. In other words, the specific swarm intelligence algorithm may perform effectively in one group of problems but not for another group of problems. In the case of GWO, the original version of GWO can solve problems by using continuous variables. On this basis, the algorithm can be modified to solve binary or discrete problems. Emary [46] proposed the combination of the evolutionary operator with GWO to solve binary problems. A similar evolutionary operator was used in the literature [47]. Jayabarathi [48] combined the crossover and mutation operators with GWO to solve economic scheduling problems. Additionally, the binary version of GWO inspired by quantum mechanics was also proposed in the literature [49]. Moreover, Liu [50] designed an initialization method based on the greedy idea and mutation and position update strategies by combining the dynamic weighted average with the static average to solve path planning problems, including the lack of GWO development ability and inflexible position update strategy. In terms of combining the advantages of different algorithms, the method that mixes multiple algorithms to improve performance has also been widely used. Singh [51] proposed the hybrid GWO and PSO optimization algorithm to improve convergence speed and the ability to determine global optimal values. Rashid [52] adopted a hybrid algorithm based on the ACO and GWO to optimize assembly sequence planning. ElGayyar [53] proposed a hybrid GWO and BA method to solve global function optimization problems. Pan [54] proposed a hybrid GWO and

flower pollination algorithm to improve algorithm performance. The other combination optimization methods include GWO and DE [55], GWO and SCA [56], GWO and BBO [57], WOA and PSO [58], and WOA and BO [59]. The NFL theorem allows the research in the field of swarm intelligence to be extremely active, which is one of the important reasons why a large number of scholars have committed to improve the current algorithm and propose new swarm intelligence algorithms.

No existing algorithm in the reviewed literature simulates the behavior of anas platyrhynchos, hence, our attempt to model the behavior of this animal and design a new swarm intelligence algorithm based on the model. Inspired by the swarm behavior of anas platyrhynchos, this study proposes a novel swarm intelligence algorithm referred as the anas platyrhynchos optimizer (APO). In general, researchers pursue a lot of performance in order to improve performance. Although the performance is improved, the introduction of additional parameters not only complicates the algorithm, but also masks the core mechanism of the swarm intelligence algorithm to a certain extent. In this study, starting from the core of the cluster intelligence algorithm, we designed the APO algorithm period to have not only better performance but also as few parameters as possible. The performance of the algorithm was tested on 23 classical benchmark functions and compared with PSO, DE, WOA, and GWO, to verify the effectiveness of APO algorithm. To further verify the performance of the algorithm in practical engineering applications, APO was used to solve the cooperative path planning problem of multi-UAVs, and the results were compared with GWO, IGWO, HGWO, PSO, and DE algorithms. The main contributions of this paper are to propose a new SI technique with inspiration from the behavior of anas platyrhynchos. A new swarm intelligence algorithm is proposed, and its performance was verified in this paper. In addition, the algorithm was applied to solve the problem of multi-UAVs cooperative path planning, thereby demonstrating its potential in engineering applications and providing new solutions for other engineering optimization problems.

The structure of the rest of this paper is as follows: Section 2 describes in detail the anas platyrhynchos optimizer proposed in this study. Section 3 present the test problem and optimization results of the benchmark function optimization and the multi-UAV cooperative path planning. The main study contents are concluded, and the future study direction is proposed in Section 4.

2. Anas Platyrhynchos Optimizer

2.1. Inspiration

Anas platyrhynchos who prefer the group life usually live in groups on banks of freshwater lakes and in rivers, lakes, reservoirs, bays, and other water bodies. In summer, anas platyrhynchos in small groups inhabit freshwater rivers, lakes, and swamps where aquatic plants flourish. In autumn, they change their feathers and often gather in large groups of hundreds or even thousands of birds in migration. In addition, anas platyrhynchos live in a group of hundreds in winter.

American biologists have found that anas platyrhynchos have a habit of controlling parts of their brains to keep on sleeping and another part to keep them awake. Anas platyrhynchos can also open one eye and close the other eye during sleeping. This study provides the first evidence that animals can control their sleeping state. The half-asleep and half-awake habit of anas platyrhynchos help them evade predator animals in a dangerous environment. During gathering in crowds and groups of anas platyrhynchos, a leading duck can be observed in the anas platyrhynchos swarm, and other ducks will flock to the leading duck. As a result, the whole duck group will move towards the destination. When a single duck is out of the flock, the lone duck can find its companion by calling and moving towards them, eventually moving with the whole flock. As a group, anas platyrhynchos can coordinate themselves into the whole flock. This type of swarm intelligence can be associated with the target problem requiring optimization. Inspired by the collective behavior of anas platyrhynchos, a new swarm intelligence algorithm can be designed.

The anas platyrhynchos optimizer mainly stimulates the warning behavior and the moving process of anas platyrhynchos swarm. In solving optimization problems, each duck is regarded as an individual particle that can move freely, while several ducks form the duck flock. At the beginning, the anas platyrhynchos are randomly distributed in the search space. The duck with the best fitness function at present is considered the leading duck. As the following object of all ducks, the leading duck guides its companions to go with it, all anas platyrhynchos mainly go through two stages of warning and moving. The warning and sentry behavior of anas platyrhynchos and the moving process of the whole flock are shown in Figure 1. In the early warning stage, the danger degree of each particle is represented by the value of its fitness function. The lower the degree of danger, the lower the probability of flying in the presence of danger. When a duck is in distress, it will fly to the least dangerous leading duck. At the same time, each duck will fly towards the leading duck during the moving stage. If the fitness function value is worse after the flight, it means that the duck is separated from the swarm. At this time, the duck randomly selects one duck: If the fitness function value of the selected duck is lower than that of the stray duck, the stray duck flies to the selected duck; if the fitness function value of the selected duck is equal to the stray duck, the stray duck fails to find a partner; if the fitness function value of the selected duck is higher than that of the stray duck, it means that the selected duck is another stray duck and the selected duck also flies to the stray duck. In this study, a new swarm intelligence algorithm referred as the anas platyrhynchos optimizer was designed on the basis of the mathematical modeling of the warning behavior and the moving process.

Figure 1. The warning behavior and the moving process of anas platyrhynchos swarm.

2.2. Mathematical Model and Algorithm

In this section, we propose a novel swarm intelligence algorithm named anas platyrhynchos optimizer (APO) which contains two main behaviors: warning behavior and moving process.

2.2.1. Warning Behavior

Before the warning behavior, according to the anas platyrhynchos behavior model, the population in APO is initialized as follows:

$$Pop_i = rand \times (up - low) + low, \quad (1)$$

where Pop_i is the i th population, N is the population size, low and up are the lower and upper bounds of the search space, and $rand$ is a random value selected from the range $[0, 1]$.

In the warning behavior, the flying in danger operation realized by the select probability P_c is proposed. The main process of the warning behavior is introduced as follows.

Step 1: Calculate the probability of distress P_c by Equation (2).

$$P_{c_i} = \frac{\text{rank}(\text{fit}(\text{Pop}_i))}{N}, \quad (2)$$

where $\text{fit}(\text{Pop}_i)$ is the fitness value of Pop_i and $\text{rank}(\text{fit}(\text{Pop}_i))$ is considered as the rank of the individual Pop_i among the other individuals in the population.

Step 2: If the probability P_c is satisfied, generate the new individual by Equation (3).

$$\text{Pop}_i(t+1) = \text{Pop}_i(t) + \text{sign}\left(\text{rand} - \frac{1}{2}\right) \times \alpha_0 \times |\text{Pop}_i(t) - \text{Pop}_{\text{best}}(t)| \times \text{Levy}(s), \quad (3)$$

where t refers to the current iterations, and Pop_{best} is the leading duck, $\alpha_0 > 0$ is the step length scale factor, and sign refers to the sign function. Levy flight provides the random walk for the Equation (6), and its distribution equation is expressed as Equation (4).

$$\text{Levy} \sim \mu = t^{-\lambda}, \quad 1 < \lambda \leq 3. \quad (4)$$

Levy flight is a special type of random walk, and the probability distribution of its step length obeys a heavy-tailed distribution which is shown in Equation (5).

$$s = \frac{\mu}{|\nu|^{\frac{1}{\beta}}}, \quad (5)$$

where s refers to the Levy flight step length. Moreover, in Equation (4), $\lambda = 1 + \beta$, we use $\alpha_0 = 0.01$ and $\beta = 3/2$ as CS set in [60]. μ and ν are selected from the normal distribution $\mu = N(0, \delta_\mu^2)$ and $\nu = N(0, \delta_\nu^2)$, where

$$\delta_\mu = \left[\frac{\Gamma(1+\beta) \sin(\pi\beta/2)}{\beta \times \Gamma((1+\beta)/2) \times 2^{(\beta-1)/2}} \right]^{\frac{1}{\beta}}, \quad \delta_\nu = 1 \quad (6)$$

2.2.2. Moving Process

Reference [61] describes an intelligent algorithm for simulating biological migration. Compared with the self-organizing migration algorithm [61] (SOMA), APO mainly focus on the anas platyrhynchos, imitates its migration movement and models it. It is completely different from the location update strategy and algorithm flow in SOMA. In this behavior of APO, the main process is introduced as follows.

Step 1: After the optimal particle is defined, other search particles will attempt to move to the optimal particle. The mathematical model of this behavior is given by Equation (7).

$$\text{Pop}_i(t+1) = \text{Pop}_i(t) - A[C \times \text{Pop}_{\text{best}}(t) - \text{Pop}_i(t)], \quad (7)$$

where A and C refer to the coefficient vector, which can be obtained from Equations (8) and (9).

$$A = 2a \times \text{rand} - a, \quad (8)$$

$$C = 2 \times \text{rand}, \quad (9)$$

where a refers to a coefficient vector that linearly decreases with the iterations. The value of a can be obtained from Equation (10).

$$a = 2 - t \frac{2}{T} \quad (10)$$

where T is the maximum number of iterations.

Figure 2a presents the basic principle of Equation (7) in solving 2D problems. The position of particles (X, Y) can be updated according to the position of the current optimal solution (X^*, Y^*) . By adjusting the coefficient vectors A and C , the development search can be implemented for different positions around the optimal solution according to the current position. Figure 2b shows the possible position of the particles after they are updated in 3D space. Due to the randomness of vectors A and C , the particles can reach any area among the key points in the search space. Therefore, the particles can be updated to an area near the optimal solution by using Equation (7) as a means of simulating the behavior of moving to the leading duck.

Figure 2. 2D and 3D position vectors and their succeeding possible positions.

Step 2: If the new individual is worse than the old solution, select another particle Pop_{rand} at random.

Step 3: If Pop_{rand} is better than Pop_i , the i th individual moves to the random particle Pop_{rand} by Equation (11).

$$Pop_i(t+1) = (Pop_{rand}(t) - Pop_i(t)) \times e^{-l^2} + Pop_i(t), \quad (11)$$

where l refers to the distance of the random particle and the i th individual.

Step 4: If Pop_{rand} is equal to Pop_i , it keeps unchanged.

Step 5: If Pop_{rad} is worse than Pop_i , the random particle moves to the i th individual by Equation (11).

$$Pop_{rand}(t+1) = (Pop_i(t) - Pop_{rand}(t)) \times e^{-l^2} + Pop_{rand}(t), \quad (12)$$

2.2.3. Flow of APO

Starting from the core of the swarm intelligence algorithm, global exploration and local development are carried out reasonably at the beginning to ensure a rapid convergence to a feasible solution. The operator uses random factors to ensure that the algorithm can move out of the local optimum. Moreover, the algorithm should have as few parameters as possible for simplification and ease in implementation. The anas platyrhynchos optimizer mainly consists of two operators, namely, the warning behavior and the moving process. These two operators ensure a balance between global exploration and local development. The pseudo code of the anas platyrhynchos optimizer is shown in Table 1.

Table 1. Pseudo code of the anas platyrhynchos optimizer (APO).

Setting: *low, up, N, D, ObjFun*, and *T*
 Initial population according to Equation (1)
 Calculating fitness function of population by $fit(Pop_i) = ObjFun(Pop_i)$
 Finding the leading duck Pop_{best}
 $t = 1$
while ($t \leq T$)
 Calculating the P_c according to Equation (2)//warning behavior
 for i from 1 to N **do**
 if $rand < P_c$
 Updating position according to Equation (3)
 end if
 Updating parameters A and C according to Equations (8) and (9)//moving process
 Updating position according to Equation (7)
 if $fit(Pop_i)_{new} > fit(Pop_i)_{old}$ //take the minimum as an example
 Selecting another particle Pop_{rand} at random
 if $fit(Pop_i) > fit(Pop_{rand})$
 Updating position according to Equation (11)
 elseif $fit(Pop_i) = fit(Pop_{rand})$
 Keeping unchanged
 elseif $fit(Pop_i) < fit(Pop_{rand})$
 Updating position according to Equation (12)
 end if
 end for
 Finding the leading duck Pop_{best}
 $t = t + 1$
end while
return Pop_{best}

3. Experiment and Application

To study our APO algorithm, experiments were carried out on both unconstrained and constrained global optimization problems. For unconstrained problems, 23 classic benchmark functions were employed. For constrained benchmark functions, the problem of multi-UAVs cooperative path planning were used. In addition, all the experiments in this section were performed on computer with 2.20 GHz Intel(R) Core(TM) i5-5200U processor and 8 GB of RAM using MATLAB 2016b.

3.1. Performance Analysis: Classic Benchmark Functions

To illustrate the relative success of APO on classic benchmark functions, a series of 23 widely used benchmarks selected from References [62–67] were introduced in this test. Tables 2 and 3 present the detailed information about the different types of benchmarks. Our APO algorithm was compared with four well-known nature-based algorithms: GWO, WOA, PSO, and DE.

Table 2. High-dimensional benchmark functions used in Test 1.

Test Function	D	Range	Optimum
$f_1(x) = \sum_{i=1}^n x_i^2$	30	[−100,100]	0
$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	[−10,10]	0
$f_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	30	[−100,100]	0
$f_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	30	[−100,100]	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

Table 2. Cont.

Test Function	D	Range	Optimum
$f_6(x) = \sum_{i=1}^n ([x_i + 0.5])^2$	30	[-100,100]	0
$f_7(x) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1)$	30	[-1.28,1.28]	0
$f_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	[-500,500]	-418.9825*5
$f_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$	30	[-32,32]	0
$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
$f_{12}(x) = \frac{\pi}{n} \{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50,50]	0
$f_{13}(x) = 0.1 \{ \sin^2(3\pi x_i) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	[-50,50]	0

D: dimension; range: limits of search space; optimum: global optimal value.

Table 3. Low-dimensional benchmark functions used in Test 1.

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

D: dimension; range: limits of search space; optimum: global optimal value.

In this test, the population size N was set to 30. The maximum function evolution numbers (T) was set to 500. For each test function, 30 independent runs were carried out with random seeds. The parameter settings for the other compared algorithms in this test are detailed in Table 4. Such values represent the best parameter sets for these algorithms according to the original paper of GWO [8], WOA [9], PSO [2], and DE [68].

Table 4. The parameter settings for the other compared algorithms.

Algorithms	Parameter Settings
Gray wolf optimization (GWO) [8]	$a = 2 - t*2/T$
Whale optimization algorithm (WOA) [9]	$a_1 = 2 - t*2/T, a_2 = -1 + t*(-1)/T$
Particle swarm optimization (PSO) [2]	$w_{max} = 0.9, w_{min} = 0.2, c_1 = c_2 = 2$
Differential evolution (DE) [68]	$F = 0.5, CR = 0.9$

Table 5 summarizes the statistical results including mean, worst, best and SD of objective function values obtained by 30 independent runs for high-dimensional functions in Table 2. Compared with the other four algorithms, APO exhibited the best performance on the high-dimensional benchmark functions except function f_4 . Gray wolf optimization had a promising performance for optimizing f_4 , and APO exhibited the second-best performance on this function; meanwhile, APO sometimes has the best value for optimizing f_4 . From Table 5, it can be found that APO nearly finds all the best values of f_9 and f_{11} . According to the test results of Table 5, APO outperforms the other algorithms on high-dimensional benchmark functions.

Table 5. Best, worst, mean, and SD obtained on functions from Table 2.

F	Result	APO	GWO	WOA	PSO	DE	Rank
f_1	Best	7.8190×10^{-145}	1.0102×10^{-29}	1.9587×10^{-80}	9.3358×10^{-6}	2.3626×10^{-6}	1
	Worst	6.9690×10^{-108}	1.6995×10^{-26}	7.1646×10^{-63}	0.0029	2.1183×10^{-4}	
	Mean	2.3236×10^{-109}	2.1408×10^{-27}	2.8808×10^{-64}	2.6064×10^{-4}	3.3728×10^{-5}	
	Std	1.2723×10^{-108}	4.1907×10^{-27}	1.3261×10^{-63}	5.4256×10^{-4}	4.4013×10^{-5}	
f_2	Best	5.5304×10^{-93}	1.1204×10^{-17}	2.0437×10^{-56}	0.0036	5.1036×10^{-4}	1
	Worst	2.4478×10^{-73}	3.6284×10^{-16}	3.4372×10^{-46}	0.1201	0.0033	
	Mean	1.3539×10^{-74}	9.5431×10^{-17}	2.0876×10^{-47}	0.0309	0.0016	
	Std	5.2303×10^{-74}	8.7339×10^{-17}	7.4074×10^{-47}	0.0268	6.9524×10^{-4}	
f_3	Best	1.5871×10^{-99}	1.1907×10^{-63}	4.7969×10^{-12}	5.4371×10^{-27}	2.2540×10^{-35}	1
	Worst	8.1085×10^{-78}	5.6634×10^{-50}	0.0293	5.0743×10^{-22}	1.8224×10^{-29}	
	Mean	6.0509×10^{-79}	2.0580×10^{-51}	0.0033	3.7279×10^{-23}	1.6513×10^{-30}	
	Std	1.7318×10^{-78}	1.0323×10^{-50}	0.0068	1.1100×10^{-22}	3.9570×10^{-30}	
f_4	Best	1.2424×10^{-16}	6.5653×10^{-8}	5.3046	0.5731	5.7123	2
	Worst	0.0312	1.9500×10^{-6}	89.6219	1.5222	37.5491	
	Mean	0.0029	5.7114×10^{-7}	59.6414	1.1226	16.8984	
	Std	0.0078	4.6159×10^{-7}	29.3820	0.2522	7.2691	

Table 5. Cont.

<i>F</i>	Result	APO	GWO	WOA	PSO	DE	Rank
<i>f</i> ₅	Best	0.0078	26.0434	27.1465	25.3115	19.0580	1
	Worst	28.7807	28.7485	28.8434	188.8556	181.2647	
	Mean	26.6971	27.2864	28.3636	85.1733	56.5542	
	Std	7.2574	0.7167	0.4466	40.5425	40.1468	
<i>f</i> ₆	Best	1.2702×10^{-6}	1.3209×10^{-4}	0.7374	7.9911×10^{-6}	3.6153×10^{-6}	1
	Worst	4.3859×10^{-5}	1.5119	3.1981	4.1143×10^{-4}	1.8978×10^{-4}	
	Mean	1.3972×10^{-5}	0.6602	1.9556	9.7234×10^{-5}	4.7963×10^{-5}	
	Std	9.8769×10^{-6}	0.3692	0.6576	8.7218×10^{-5}	4.8581×10^{-5}	
<i>f</i> ₇	Best	1.9336×10^{-5}	4.8037×10^{-4}	6.4622×10^{-5}	0.0804	0.0360	1
	Worst	0.0077	0.0034	0.0194	0.3469	0.1085	
	Mean	8.5533×10^{-4}	0.0019	0.0050	0.1729	0.0797	
	Std	0.0015	7.0049×10^{-4}	0.0057	0.0584	0.0158	
<i>f</i> ₈	Best	−12569	−7250.9	N/A	−8727.2	−7317.7	1
	Worst	−12150	−3727.9	N/A	−2629.0	−4691.7	
	Mean	−12529	−6129.3	N/A	−4978.3	−5946.6	
	Std	106.3406	767.8156	N/A	1569.8	667.0457	
<i>f</i> ₉	Best	0	5.6843×10^{-14}	0	33.8862	1.4607×10^2	1
	Worst	0	16.9267	5.6843×10^{-14}	83.6992	2.1638×10^2	
	Mean	0	4.0526	1.8948×10^{-15}	58.1507	1.8134×10^2	
	Std	0	4.5006	1.0378×10^{-14}	14.4306	16.9916	
<i>f</i> ₁₀	Best	8.8818×10^{-16}	5.7732×10^{-14}	8.8818×10^{-16}	0.0016	5.7932×10^{-4}	1
	Worst	7.9936×10^{-15}	1.3589×10^{-13}	7.9936×10^{-15}	1.5175	0.0226	
	Mean	2.6645×10^{-15}	1.0309×10^{-13}	3.8488×10^{-15}	0.1757	0.0024	
	Std	2.2372×10^{-15}	1.7645×10^{-14}	2.6526×10^{-15}	0.4313	0.0040	
<i>f</i> ₁₁	Best	0	0	0	1.2377×10^{-6}	1.5597×10^{-5}	1
	Worst	0	0.0384	0.1508	0.0246	0.0340	
	Mean	0	0.0030	0.0086	0.0071	0.0032	
	Std	0	0.0090	0.0333	0.0081	0.0072	
<i>f</i> ₁₂	Best	8.3024×10^{-8}	0.0063	0.0295	1.3634×10^{-7}	5.3389×10^{-6}	1
	Worst	0.0065	0.0939	0.7203	0.1037	8.3340×10^2	
	Mean	2.1901×10^{-4}	0.0373	0.1473	0.0104	62.5083	
	Std	0.0012	0.0186	0.1722	0.0316	1.9621×10^2	
<i>f</i> ₁₃	Best	1.9591×10^{-6}	0.2159	0.6067	1.8128×10^{-6}	1.0309×10^{-5}	1
	Worst	4.1407×10^{-5}	1.1442	1.9571	0.0178	7.4778×10^2	
	Mean	1.1372×10^{-5}	0.6458	1.3359	0.0057	26.6912	
	Std	1.0199×10^{-5}	0.2486	0.3731	0.0058	1.3623×10^2	

The statistical results of the algorithms on low-dimensional benchmark functions are shown in Table 6. Compared with the other four algorithms, APO exhibited the best performance on f_{15} , f_{21} , f_{22} and f_{23} . For f_{16} and f_{17} , each algorithm nearly found all the best values. From Table 6, there is almost no difference in the performance on f_{18} , f_{19} and f_{20} of each algorithm. In addition, although APO performs moderately on f_{14} , it can also find the optimal value most of the time.

Table 6. Best, worst, wean, and SD obtained on functions from Table 3.

F	Result	APO	GWO	WOA	PSO	DE	Rank
f_{14}	Best	0.9980	0.9980	0.9980	0.9980	0.9980	4
	Worst	12.6705	12.6705	10.7632	10.7632	2.9821	
	Mean	4.2973	4.4937	2.0822	3.1636	1.0641	
	Std	4.0039	4.0086	2.1440	2.8550	0.3622	
f_{15}	Best	3.1012×10^{-4}	3.0749×10^{-4}	3.2572×10^{-4}	5.0404×10^{-4}	3.0749×10^{-4}	1
	Worst	0.0023	0.0204	0.0043	0.0011	0.0204	
	Mean	5.2403×10^{-4}	0.0037	0.0012	8.2951×10^{-4}	0.0018	
	Std	5.1504×10^{-4}	0.0076	8.6353×10^{-4}	1.5813×10^{-4}	0.0051	
f_{16}	Best	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	=
	Worst	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	
	Mean	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	
	Std	8.4272×10^{-9}	1.7842×10^{-8}	2.1111×10^{-9}	6.3877×10^{-16}	6.7752×10^{-16}	
f_{17}	Best	0.3979	0.3979	0.3979	0.3979	0.3979	=
	Worst	0.3979	0.3979	0.3979	0.3979	0.3979	
	Mean	0.3979	0.3979	0.3979	0.3979	0.3979	
	Std	9.2877×10^{-7}	2.0274×10^{-4}	1.4524×10^{-5}	0	0	
f_{18}	Best	3.0000	3.0000	3.0000	3.0000	3.0000	≈
	Worst	30.0000	3.0000	30.0033	3.0000	3.0000	
	Mean	6.6000	3.0000	3.9003	3.0000	3.0000	
	Std	9.3351	4.3052×10^{-5}	4.9301	1.6472×10^{-15}	2.0301×10^{-15}	
f_{19}	Best	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628	≈
	Worst	−3.8549	−3.8549	−3.8445	−3.8628	−3.8628	
	Mean	−3.8605	−3.8613	−3.8611	−3.8628	−3.8628	
	Std	0.0036	0.0027	0.0038	2.5829×10^{-15}	2.7101×10^{-15}	
f_{20}	Best	−3.3220	−3.3220	−3.3218	−3.3220	−3.3220	≈
	Worst	−3.0839	−3.0499	−3.1627	−3.2031	−3.2031	
	Mean	−3.2599	−3.2672	−3.2599	−3.2784	−3.2507	
	Std	0.0838	0.0861	0.0651	0.0583	0.0592	
f_{21}	Best	−10.1532	−10.1529	−10.1526	−10.1532	−10.1532	1
	Worst	−0.8810	−2.6302	−2.6020	−2.6305	−2.6305	
	Mean	−9.5341	−8.8857	−8.2591	−7.0685	−9.4850	
	Std	2.3522	2.3709	2.9335	3.1409	2.0722	
f_{22}	Best	−10.4029	−10.4025	−10.4009	−10.4029	−10.4029	1
	Worst	−10.3988	−5.0876	−1.8302	−2.7519	−3.7243	
	Mean	−10.4015	−10.2239	−7.4542	3.1828	−10.1803	
	Std	0.0011	0.9701	3.0686	−8.2324	1.2193	
f_{23}	Best	−10.5364	−10.5360	−10.5363	−10.5364	−10.5364	1
	Worst	−10.5303	−2.4217	−1.6447	−2.4217	−2.8711	
	Mean	−10.5348	−10.2638	−6.9538	−9.5065	−10.0575	
	Std	0.0015	1.4811	3.7470	2.3863	1.8268	

According to Tables 5 and 6, the effectiveness of the APO algorithm can be verified. In very few test functions, the results of APO are slightly inferior to other swarm intelligence algorithms, but still meet the discussion of the no free lunch theorem. In 70% of the test functions, APO performs significantly better than other algorithms. In addition, in the 22% function, the APO result is not significantly different from other functions. Therefore, the performance of APO is obviously superior to other algorithms, and the APO has a higher priority in the choice of algorithm when solving optimization problems.

Berg [69] reported that mutation occurs during particle movement in the initial iteration process, and this mutation is conducive to the extensive and comprehensive exploration of particles in the search space. As the number of iterations increases, the mutation should be reduced accordingly to be able to fully demonstrate the local development ability of the algorithm. Column 3 in Figure 3 lists the search history and track of the first particle in the first dimension. Mutation occurs in the initial stage of the iteration. As the iteration continues, the mutation gradually decreases. According to Reference [69], this behavior ensures that the anas platyrhynchos optimizer eventually converges to a point in the search space. Meanwhile, Column 2 in Figure 3 shows the search history of the particles. The particles of the anas platyrhynchos optimizer tend to search for a prospective area in the search space to develop a better area. In this problem, we set the population number to 50 and the maximum number of iterations to 100 and used different dimensions from those in Tables 2 and 3.

Figure 3. Cont.

Figure 3. Search history and track of the first particle in the first dimension (part).

From a graphical point of view, Figures 4 and 5 present the partly converged curves abstained by the five algorithms with 30 independent runs. When Figures 4 and 5 are examined, APO converges rapidly for most test functions. Especially for the unimodal benchmark functions, APO has a faster convergence rate compared to the other algorithms. For the most multimodal functions, APO also presents the promising performance in convergence rate as shown in Figures 4 and 5.

Figure 4. Cont.

Figure 4. Evolution curves of the fitness value for some of the functions in Table 2.

Figure 5. Evolution curves of the fitness value for some of the functions in Table 3.

For detailed comparison, the Wilcoxon signed tanks test [70], in which Iman–Davenport’s procedure was used as a post-hoc procedure, was firstly used to make a pairwise comparison so as to illustrate the superiority of APO on the 23 test functions. The statistical results are summarized in Table 7. The p -values in Table 7 additionally present the superiority of the APO because most of the p -values are much smaller than 0.05.

Table 7. The results of Wilcoxon signed ranks test based on the solution for each benchmark with 30 independently runs.

Functions	APO Versus GWO	APO Versus WOA	APO Versus PSO	APO Versus DE
f_1	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_2	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_3	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_4	0.0039	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_5	6.6392×10^{-4}	0.1156	8.4661×10^{-6}	0.0018
f_6	1.7344×10^{-6}	1.7344×10^{-6}	5.7517×10^{-6}	1.6046×10^{-4}
f_7	2.4118×10^{-4}	1.6046×10^{-4}	1.7344×10^{-6}	1.7344×10^{-6}
f_8	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_9	1.7224×10^{-6}	1	1.7344×10^{-6}	1.7344×10^{-6}
f_{10}	1.6521×10^{-6}	0.0623	1.7344×10^{-6}	1.7344×10^{-6}
f_{11}	0.5000	0.1250	1.7344×10^{-6}	1.7344×10^{-6}
f_{12}	1.7344×10^{-6}	1.7344×10^{-6}	0.0132	5.2165×10^{-6}
f_{13}	1.7344×10^{-6}	1.7344×10^{-6}	2.1266×10^{-6}	1.7344×10^{-6}
f_{14}	0.5440	0.0519	0.2059	1.7344×10^{-6}
f_{15}	0.0859	4.1955×10^{-4}	0.0028	0.0859
f_{16}	1.0570×10^{-6}	0.0786	1.7344×10^{-6}	1.7344×10^{-6}
f_{17}	8.9443×10^{-6}	6.9838×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_{18}	1.9209×10^{-6}	1.4936×10^{-6}	1.7344×10^{-6}	1.7344×10^{-6}
f_{19}	0.8936	0.5577	1.7344×10^{-6}	1.7344×10^{-6}
f_{20}	0.5038	0.2134	0.0285	0.8130
f_{21}	0.0013	1.8910×10^{-4}	0.0449	0.0018
f_{22}	0.0300	1.7344×10^{-6}	0.6435	3.1123×10^{-5}
f_{23}	0.0350	5.2165×10^{-6}	0.0571	0.0350

In summary, the results of this section experimentally prove that the APO shows very competitive results and outperforms other well-known algorithms on the classic benchmark functions, and we state that the APO has merit among the other five algorithms. The next section inspects the performance of the APO in solving the problem of multi-UAVs cooperative path planning, and multi-UAVs cooperative path planning has high-dimension, multi-constraint, and space–time cooperation characteristics which are a challenging problem.

3.2. Engineering Application: Multi-UAVs Cooperative Path Planning

At present, the research methods used for path planning can be divided into the following categories: methods based on graph theory, methods based on potential field, search algorithms based on heuristic information, and algorithms based on swarm intelligence optimization. The results show that the swarm intelligence algorithm has obvious advantages in processing high-dimensional and

multi-constraint challenges. The novel proposed APO can then be used to solve the problem on multi-UAVs cooperative path planning to further verify the algorithm's effectiveness. In this part, we will cite the model in reference [71] and the algorithms involved to verify algorithm performance.

3.2.1. Modeling the UAV Cooperative Path Planning

In path planning, an appropriate planning space must be established in accordance with the flight environment and mission requirements. In the present work, taking a mountain background as a task environment, a digital elevation model is established using a random function to simulate peaks and other threat obstacles. The mountain model function is proposed in Reference [72]. This model consists of the original digital and threat equivalent terrain models. The former is expressed as:

$$z_1(x, y) = \sin(y + a) + b \sin x + c \cos\left(d \sqrt{x^2 + y^2}\right) + e \cos y + f \sin\left(f \sqrt{x^2 + y^2}\right) + g \cos y \quad (13)$$

where x and y refer to the point coordinates on a horizontal projection plane; z_1 refers to the height coordinate that corresponds to the coordinate points on a horizontal plane; a, b, c, d, e, f , and g are the coefficients. The topography of a different landform can be obtained by changing parameters.

The threat equivalent terrain model is:

$$z_2(x, y) = \sum_{i=1}^k h(i) \exp\left(-\left(\frac{x - x_{0i}}{x_{si}}\right)^2 - \left(\frac{y - y_{0i}}{y_{si}}\right)^2\right), \quad (14)$$

where z_2 refers to the height of the peak; $h(i)$ refers to the height of the highest point of peak i on a base terrain; x_{0i} and y_{0i} refer to the coordinates of the highest point of peak i ; x_{si} and y_{si} refer to the variables related to the slope of peak i along the x, y axis. If x_{si} and y_{si} are large, then the slope of the peak is flat and abrupt.

The final mountain threat model is obtained by integrating the original digital terrain model into the threat equivalent terrain model:

$$z(x, y) = \max[z_1(x, y), z_2(x, y)], \quad (15)$$

We set the starting point of a certain UAV as $S(x_0, y_0, z_0)$ and the target point as $E(x_e, y_e, z_e)$. The number of waypoints is n , and the waypoints searched can be represented by $\{S, P_1, P_2, \dots, P_n, E\}$; among these variables, the coordinate of a track node is $P = (x_p, y_p, z_p)$.

3.2.2. Path Cost Function

The purpose of multi-UAV cooperative path planning is that, on the premise of satisfying the requirements of a safe flight and space-time cooperation, every UAV can search the corresponding path, and the synthetic path cost of UAV fleet must be the least. Therefore, path planning requires the establishment of a path cost function as an index to evaluate the quality of a path. The satisfaction of the spatial and temporal cooperative constraints of multi-UAV by considering the dynamics and threat constraints of a single UAV in multi-UAV cooperative path planning is required. Thus, given the planning objective, the following cost indexes are considered in the present work: the performance indexes of a single UAV include fuel consumption, maximum climb/slide angle, flying altitude, peak threat, and multi-UAV time cooperation. Spatial cooperation is manifested in multi-UAV path collision avoidance. We set different flight altitudes that must be avoided by each UAV. The synthetic cost function is established as:

$$J = \omega_1 J_{fuel} + \omega_2 J_{angle} + \omega_3 J_{height} + \omega_4 J_{threat} + \omega_5 J_{coop}, \quad (16)$$

where $\omega_1, \omega_2, \omega_3, \omega_4$ and ω_5 refer to the weights of different cost indexes, and the sum of weights is 1. The paths that satisfy different requirements can be obtained by adjusting the weights. To ensure that all cost indexes are involved in path planning, the functions are normalized in accordance with the range of their values, and then weighted summation is performed.

Fuel consumption cost is related to the length of flight path and flying speed. Assuming that UAVs consistently fly at a certain speed, fuel costs can be replaced by the length of the path:

$$J_{fuel} = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}, \quad (17)$$

where $(x_{i+1}, y_{i+1}, z_{i+1})$ and (x_i, y_i, z_i) correspond to the coordinates of the adjacent path points.

J_{angle} refers to the cost of the maximum climb/slide angle and is expressed as:

$$J_{angle} = \sum_{i=1}^n \theta_i |\theta_i| = \arctan \left(\frac{|z_{i+1} - z_i|}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \right), \quad (18)$$

where θ_i refers to the climb/slide angle of the adjacent points of a certain path.

To satisfy the requirements of flight safety and concealment, the flight altitude cannot be overly low or high. Height cost can be expressed as:

$$J_{height} = \sum_{i=1}^n |h_i - safth_i|, \quad (19)$$

where h_i refers to the height of path point i on a certain path, and $safth_i$ refers to the minimum safety height for each UAV.

Collision with the mountain in the flying course of the UAV must be avoided. In Reference [73], the peak model is represented by a cone approximate representation. The path is divided into m equal sections, and $m-1$ sampling points are obtained in the center. The threat cost of the whole path is expressed as

$$J_{threat} = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^K threat(j, k), \quad (20)$$

where n refers to the number of path points, K represents the number of peaks, and $threat(j, k)$ is the threat cost of the sampling point (x_i, y_i, z_i) in the current section and a certain peak and is expressed as:

$$threat(j, k) = \begin{cases} 0 & h_j > H(k) \text{ or } d_T > R_T + d_{min} \\ R_T(h) + d_{T_{min}} - d_T & h_j < H(k) \text{ and } d_T < R_T + d_{T_{min}} \end{cases} \quad (21)$$

$$R_T(h) = (H(k) - h) / \tan \theta$$

where n refers to the number of path points, K represents the number of peaks, $H(k)$ refers to the height of peak k , and R_T refers to the maximum extension radius. Moreover, h_j is the flying altitude of the current UAV, d_T refers to the distance from the UAV to the symmetrical axis of the peak, $d_{T_{min}}$ denotes the minimum distance allowed on the terrain, and θ refers to the slope of the terrain. The terrain threat is depicted in Figure 6.

Figure 6. Terrain threat map.

Cooperative cost function implies time cooperation. All UAVs are required to reach the target point simultaneously as far as possible. If the course of a certain path cannot satisfy the time cooperative constraints, then the path must be corrected. Assuming that the flying speed of the UAV is in the range of $[v_{min}, v_{max}]$ and the course of the UAV i is L_i , its flight time period is $T_i \in [T_{min}^i, T_{max}^i]$. Similarly, assuming that the flying time of UAV j is in the range of $T_j \in [T_{min}^j, T_{max}^j]$, if the flight time of the two UAVs intersects, then temporal cooperation is feasible, that is

$$T_{inter} = T_i \cap T_j \neq \emptyset, \quad (22)$$

in accordance with the temporal cooperation evaluation formula between paths in Reference [74], the temporal cooperation cost function is obtained on the basis of the planning model in the present work:

$$J_{coopT} = \begin{cases} 1 & T_{inter} = \emptyset \\ 2 & 0 < T_{inter} < rand \times T_{min} \\ \frac{3T_{inter}}{T_{min}} & T_{inter} > \frac{1}{3} T_{min} \end{cases} \quad (23)$$

where T_{min} refers to a time period with a small range of a certain path in the flight time period, and T_{inter} represents the intersection of the flight time for two paths.

3.2.3. Multi-Population Track Coding

The idea of multi-population is combined in this work to construct the multi-UAVs track set. When the algorithm is adopted to perform a search, the number of populations is determined in accordance with the number of UAVs, and the process of UAV searching path is mapped to the updating process of the population. The position of the anas platyrhynchos in each sub-population corresponds to a flying path of the mapped UAV. The optimal solution in the sub-population corresponds to the optimal path of this UAV. The optimal solution of all sub-populations corresponds to the cooperative path of UAVs.

If all UAVs on a mission are set as $U = \{U_i, i = 1, 2, \dots, N_u\}$, then the corresponding number of anas platyrhynchos population is N_u . The number of anas platyrhynchos individuals in each sub-population is set as m , and the anas platyrhynchos individuals in the sub-population can be represented as $X = \{X_i, i = 1, 2, \dots, m\}$. The position of anas platyrhynchos i in the search space is $X_i = \{X_{i1}, X_{i2}, \dots, X_{in}\}$, which represents the midway point of a path in addition to the starting and the target points. The coordinate of each path point is $X_n = \{x_{in}, y_{in}, z_{in}\}$. The fitness value of the anas platyrhynchos individual corresponds to the cost function value of a certain path. If the fitness is favorable, then the path is optimal. In the process of searching and comparing the fitness values of individuals in the population, the anas platyrhynchos individuals at the favorable position are obtained as the leading duck. The algorithm is guided to search continuously in the direction with small fitness values, thereby resulting in many cooperative paths.

When the APO algorithm is initialized, a group of anas platyrhynchos is randomly generated in the region of search. When an individual's three-dimensional position coordinates are randomly generated, the computational load is large, and the search efficiency is low. Therefore, in this work, an equal division was performed for the coordinates of individuals in the direction of the x -axis based on the number of the waypoints set at initialization. In the searching process, the positions in the y - and z -axes were updated. Boundary control was performed when the position was updated. The planning problem was transformed into a search optimization problem in a two-dimensional space.

Based on the APO algorithm in this work, the specific flow of the multi-UAVs cooperative path planning is exhibited in Figure 7.

Figure 7. The specific flow of the multi-UAVs cooperative path planning based on APO.

3.2.4. Simulation Validation

The planning space was set as $100 \text{ km} \times 100 \text{ km} \times 500 \text{ m}$, including six peaks. The parameters of the original digital terrain model were set to $a = 0.1$, $b = 0.01$, $c = 1$, $d = 0.1$, $e = 0.2$, $f = 0.4$, and $g = 0.02$. The height of the peak, horizontal coordinates of the highest point, and slope parameters are listed in Table 8. Multi-UAV cooperative path planning was conducted under a known mission assignment scheme. In the simulation experiment, the path sub-population was initialized in accordance with the number of UAVs. The numbers of individuals in the sub-population, iterations, and path points were 50, 100, and 10, respectively. The weight coefficients of all cost functions corresponded to 0.4, 0.2, 0.1, 0.2, and 0.1. The flight speed range of the UAV was 40–60 m/s.

Table 8. Model parameters of peaks.

Number	Altitude (m)	Center Location (x,y)/(km)	Slope
1	130	(56,82)	(10,10)
2	150	(75,20)	(10,10)
3	300	(50,45)	(12,12)
4	100	(22,20)	(8,8)
5	150	(20,70)	(8,8)
6	150	(77,73)	(10,10)

Case 1: Three UAVs begin from the starting point and arrived at the designated target point to perform tasks. The coordinates of the starting and target points are presented in Table 9.

Table 9. Coordinates of the starting and target points of all UAVs (case 1).

Number	Start Point	Target Point
1	(1,1,0)	(100,30,70)
2	(1,30,0)	(100,40,70)
3	(1,60,0)	(100,50,70)

The three-dimensional path planning and contour map of each UAV are displayed in Figure 8a,b. The cost and synthetic path cost convergence curves of each UAV are plotted in Figure 8c,d. Figure 8 illustrates that all UAVs could effectively avoid the threat and reach the target point. The cost function values of each UAV gradually converge with the increase in iteration times, thereby verifying the effectiveness of the algorithm. Through a simulation, the flight time intervals (unit: s) and range (unit: km) of each UAV are presented in Table 10. The time intersection is [1849.0089, 2554.5307]. The time synergy requirement can be satisfied by setting different flight speeds for all UAVs.

Figure 8. The result of case 1. (a) 3D UAV cooperative path map; (b) contour map of cooperative path; (c) convergence curve of each UAV; (d) path cost convergence curve.

Table 10. Range and flight time of each aircraft (case 1).

Number	Range (km)	Flight Time (s)
1	110.9405	[1849.0089, 2773.5134]
2	102.1812	[1703.0205, 2554.5307]
3	102.259	[1704.3164, 2556.4746]

Case 2: Four UAVs fly to two target points. The coordinates of the starting and the target points are listed in Table 11.

The planned path diagrams of UAVs are obtained as depicted in Figure 9a,b. The path cost convergence and synthetic cost curves of each UAV are plotted in Figure 9c,d. Through a simulation, the flight time intervals (unit: s) and range (unit: km) of each UAV are presented in Table 12. The time intersection is [1961.975, 2417.0379]. The planned paths and voyages are close and can effectively avoid obstacles. If the UAVs are close to one another, then collisions can be avoided by setting different flight altitudes.

Table 11. Coordinates of the starting and target points of all UAVs (case 2).

Number	Start Point	Target Point
1	(1,1,0)	(90,40,100)
2	(1,30,0)	(95,40,100)
3	(1,55,0)	(95,85,100)
4	(1,90,0)	(80,85,100)

Figure 9. The result of case 2. (a) 3D UAV cooperative path map; (b) Contour map of cooperative path; (c) Convergence curve of each UAV; (d) Path cost convergence curve.**Table 12.** Range and flight time of each aircraft (case 2).

Number	Range (km)	Flight Time (s)
1	117.7185	[1961.975, 2942.9626]
2	99.8391	[1663.9855, 2495.9783]
3	106.0902	[1768.1699, 2652.2548]
4	97.3108	[1621.8469, 2432.7704]

Case 3: Six UAVs fly to six target points. The coordinates of the starting and the target points are listed in Table 13.

The result of case 3 was obtained as depicted in Figure 10. Through a simulation, the flight time intervals (unit: s) and range (unit: km) of each UAV are presented in Table 14. The time intersection is [1924.111, 2481.215]. The planned paths and voyages are close and can effectively avoid obstacles.

Table 13. Coordinates of the starting and target points of all UAVs (case 3).

Number	Start Point	Target Point
1	(1,1,0)	(99,10,70)
2	(1,20,0)	(99,18,70)
3	(1,40,0)	(99,30,70)
4	(1,60,0)	(99,40,70)
5	(1,75,0)	(99,70,70)
6	(1,90,0)	(99,90,70)

Figure 10. The result of case 3. (a) 3D UAV cooperative path map; (b) contour map of cooperative path; (c) convergence curve of each UAV; (d) path cost convergence curve.**Table 14.** Range and flight time of each aircraft (case 3).

Number	Range (km)	Flight Time(s)
1	99.2486	[1654.1433, 2481.215]
2	104.8832	[1748.0538, 2622.0807]
3	115.4467	[1924.111, 2886.1664]
4	110.7951	[1846.5853, 2769.8779]
5	104.5509	[1742.5156, 2613.7734]
6	100.002	[1666.7006, 2500.0509]

Case 4: Eight UAVs fly to four target points. The coordinates of the starting and the target points are listed in Table 15.

Table 15. Coordinates of the starting and target points of all UAVs (case 4).

Number	Start Point	Target Point
1	(1,1,0)	(99,10,70)
2	(1,15,0)	(99,10,70)
3	(1,30,0)	(99,32,70)
4	(1,40,0)	(99,32,70)
5	(1,55,0)	(99,68,70)
6	(1,70,0)	(99,68,70)
7	(1,80,0)	(99,85,70)
8	(1,95,0)	(99,85,70)

The result of case 4 was obtained as depicted in Figure 11. Through a simulation, the flight time intervals (unit: s) and range (unit: km) of each UAV are presented in Table 16. The time intersection is [1922.8382, 2493.2072]. The planned paths and voyages are close and can effectively avoid obstacles.

(a) 3D UAV cooperative path map **(b)** contour map of cooperative path **(c)** convergence curve of each UAV **(d)** path cost convergence curve

Figure 11. The result of case 4. **(a)** 3D UAV cooperative path map; **(b)** contour map of cooperative path; **(c)** convergence curve of each UAV; **(d)** path cost convergence curve.

Table 16. Range and flight time of each aircraft (case 4).

Number	Range (km)	Flight Time (s)
1	99.7531	[1662.5517, 2493.8276]
2	100.9139	[1681.8984, 2522.8476]
3	115.3703	[1922.8382, 2884.2573]
4	101.3747	[1689.5775, 2534.3663]
5	101.0788	[1684.6466, 2526.97]
6	102.4978	[1708.2963, 2562.4445]
7	103.6654	[1727.7567, 2591.635]
8	100.3576	[1672.6264, 2508.9396]

The performance of the anas platyrhynchos optimizer is further verified in this work. In particular, the PSO, DE, and GWO algorithms, the improved GWO based on chaos thought (HGWO) [71], and the improved GWO with dynamic weighted average combined with the static average method (IGWO) [50] are used to implement multi-cooperative path planning in different cases. The simulation result was subsequently compared with the algorithms in this study to further verify the effectiveness of the novel algorithm. Each algorithm was run 30 times to avoid the influence of randomness, with the number of populations set to 50 and the number of iterations set to 100. Fair comparison was ensured by setting the same evaluation approach for all common parameters such as the population size, dimension, and maximum iteration of the different algorithms. The relevant parameters of these algorithms are shown in Table 17. Such values represent the best parameter sets for these algorithms according to the original paper of GWO [8], HGWO [71], IGWO [50], PSO [2], and DE [68]. The convergence curve of the 3D-integrated path cost of each case is shown in Figure 12, and the distribution of the operation results (30 times of testing) is shown in Figure 13. The average time of each algorithm in each scenario are shown in Table 18.

Table 17. The parameter settings for the other compared algorithms.

Algorithms	Parameter Settings
APO	$a = 2 - t*2/T$
GWO, HGWO, IGWO	$a = 2 - t*2/T$
PSO	$w_{max} = 0.9, w_{min} = 0.2, c_1 = c_2 = 2$
DE	$F = 0.5, CR = 0.9$

Figure 12. The convergence curve of the 3D-integrated path cost of each case. (a) case 1; (b) case 2; (c) case 3; (d) case 4.

Figure 13. The distribution of the operation results (30 times of testing) of each case. (a) case 1; (b) case 2; (c) case 3; (d) case 4.

Table 18. The average time of each algorithm in each case (unit: s).

Case	APO	GWO	HGWO	IGWO	PSO	DE
1	5.1268	5.6972	7.1175	56.6774	14.0171	27.2034
2	7.5716	8.1782	9.9407	64.2276	19.2131	38.6358
3	12.7386	12.4842	15.829	65.7687	33.3886	61.1923
4	18.7229	18.8581	21.8224	81.9172	45.57	81.6728

The results show that the cost functions of each algorithm can converge to a certain value, which meets the actual requirements of planning. As can be seen from Figure 11, the final convergence value of APO in each case was smaller than other algorithms, and the convergence speed was also faster. The comparison shows that the convergence speed and the convergence accuracy of the method proposed in this study was faster than those of the other algorithms. In addition, it can be seen from Table 17 that the calculation time of APO was better than other algorithms in case 1, 2, and 4. In case 3, APO was only 0.25 s worse than the best performing GWO. Thus, APO's advantage in computing time was verified. The results of multiple runs in various scenarios in Figure 12 show that APO can find a better solution than other algorithms in multiple runs, and then verifies the stability of the algorithm.

In summary, the APO algorithm solves the problems of slow convergence speed and low convergence accuracy of other algorithms. The calculation time is obviously better than other algorithms, and it has good stability. Therefore, the APO was superior in solving the problem of multi-UAVs cooperative path planning. Furthermore, the successful application of APO algorithm in the path planning problem provides new ideas for other practical engineering problems.

4. Conclusions

Inspired by the cluster action of anas platyrhynchos, a novel type of swarm intelligence algorithm referred as the anas platyrhynchos optimizer was proposed in this study. The global optimization function problems and multi-UAVs cooperative path planning engineering problems were solved using this algorithm. The performance of APO was analyzed using 23 classical benchmark function. The comparative experiment with the PSO, DE, WOA, and GWO algorithms verified the superiority of the anas platyrhynchos optimizer. Moreover, the anas platyrhynchos optimizer was applied to multi-UAV cooperative path planning with high-dimension, multi-constraint, time-space cooperation characteristics to further verify the algorithm's effectiveness. The multi-UAV cooperative path planning model was first constructed. Four challenging flight scenarios were set-up. Then, the multi-UAV cooperative path planning problems were solved using the anas platyrhynchos optimizer in four flight scenarios. The digital simulation experiment was carried out using MATLAB. The simulation result was further compared with the results from using the PSO, DE, GWO, IGWO, and HGWO algorithms. The final results indicate that the anas platyrhynchos optimizer was better than the other algorithms in terms of convergence speed, operation time, and calculation accuracy.

The current research mainly focuses on the anas platyrhynchos optimizer and its application in solving multi-UAV cooperative path planning problems. In the future, we intend to apply the anas platyrhynchos optimizer to more practical engineering problems and examine further the performance of this algorithm in terms of task allocation, image detection, pattern recognition, and other practical engineering problems. We also intend to develop much more complicated problems and enhance the solutions by using the anas platyrhynchos optimizer.

Author Contributions: Conceptualization, Y.Z., P.W., Y.L. (Yanbin Liu) and X.Z.; Data curation, P.W.; Formal analysis, Y.Z. and P.W.; Funding acquisition, Y.Z. and L.Y.; Investigation, Y.Z. and P.W.; Methodology, Y.Z. and P.W.; Project administration, Y.Z. and L.Y.; Resources, P.W.; Supervision, Y.Z., L.Y. and Y.L. (Yuping Lu); Validation, P.W.; Visualization, P.W.; Writing—original draft, P.W.; Writing—review and editing, Y.Z. and P.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Six Talent Peaks Project in Jiangsu Province of China (KTHY-025).

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

References

1. Branke, J.; Guntsch, M. Solving the probabilistic TSP with ant colony optimization. *J. Math. Model. Algorithms* **2004**, *3*, 403–425. [\[CrossRef\]](#)
2. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57. [\[CrossRef\]](#)
3. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [\[CrossRef\]](#)
4. Yang, X.S.; Gandomi, A.H. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **2012**, *29*, 1–18. [\[CrossRef\]](#)
5. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [\[CrossRef\]](#)
6. Mirjalili, S.Z.; Mirjalili, S.; Saremi, S.; Faris, H.; Aljarah, I. Grasshopper optimization algorithm for multi-objective optimization problems. *Appl. Intell.* **2018**, *48*, 805–820. [\[CrossRef\]](#)
7. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **2015**, *89*, 228–249. [\[CrossRef\]](#)
8. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
9. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)
10. Chen, Y.; Vepa, R.; Shaheed, M.H. Enhanced and speedy energy extraction from a scaled-up pressure retarded osmosis process with a whale optimization based maximum power point tracking. *Energy* **2018**, *153*, 618–627. [\[CrossRef\]](#)

11. Saha, A.; Saikia, L.C. Performance analysis of combination of ultra-capacitor and superconducting magnetic energy storage in a thermal-gas AGC system with utilization of whale optimization algorithm optimized cascade controller. *J. Renew. Sustain. Energy* **2018**, *10*, 014103. [[CrossRef](#)]
12. Jadhav, A.R.; Shankar, T. Whale optimization based energy-efficient cluster head selection algorithm for wireless sensor networks. *arXiv* **2017**, arXiv:1711.09389.
13. Kumawat, I.R.; Nanda, S.J.; Maddila, R.K. Positioning LED panel for uniform illuminance in indoor VLC system using whale optimization. In *Optical and Wireless Technologies*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 131–139.
14. Sreenu, K.; Sreelatha, M. W-Scheduler: Whale optimization for task scheduling in cloud computing. *Clust. Comput.* **2019**, *22*, 1087–1098. [[CrossRef](#)]
15. Mousavirad, S.J.; Ebrahimpour-Komleh, H. Multilevel image thresholding using entropy of histogram and recently developed population-based metaheuristic algorithms. *Evol. Intell.* **2017**, *10*, 45–75. [[CrossRef](#)]
16. Hassan, G.; Hassaniien, A.E. Retinal fundus vasculature multilevel segmentation using whale optimization algorithm. *Signal Image Video Process* **2018**, *12*, 263–270. [[CrossRef](#)]
17. Yuan, P.; Guo, C.; Zheng, Q.; Ding, J. Sidelobe suppression with constraint for MIMO radar via chaotic whale optimisation. *Electron. Lett.* **2018**, *54*, 311–313. [[CrossRef](#)]
18. Pathak, V.K.; Singh, A.K. Accuracy control of contactless laser sensor system using whale optimization algorithm and moth-flame optimization. *TM-Tech. Mess.* **2017**, *84*, 734–746. [[CrossRef](#)]
19. Zhang, C.; Fu, X.; Ligthart, L.P.; Peng, S.; Xie, M. Synthesis of broadside linear aperiodic arrays with sidelobe suppression and null steering using whale optimization algorithm. *IEEE Antennas Wirel. Propag. Lett.* **2018**, *17*, 347–350. [[CrossRef](#)]
20. Hegazy, A.E.; Makhoulf, M.A.; El-Tawel, G.S. Dimensionality reduction using an improved whale optimization algorithm for data classification. *Int. J. Mod. Educ. Comput. Sci.* **2018**, *11*, 37. [[CrossRef](#)]
21. Zamani, H.; Nadimi-Shahraki, M.H. Feature selection based on whale optimization algorithm for diseases diagnosis. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 1243.
22. Mafarja, M.; Mirjalili, S. Whale optimization approaches for wrapper feature selection. *Appl. Soft Comput.* **2018**, *62*, 441–453. [[CrossRef](#)]
23. Yu, Y.; Wang, H.; Li, N. Automatic carrier landing system based on active disturbance rejection control with a novel parameters optimizer. *Aerosp. Sci. Technol.* **2017**, *69*, 149–160. [[CrossRef](#)]
24. Wu, J.; Wang, H.; Li, N. Path planning for solar-powered UAV in urban environment. *Neurocomputing* **2018**, *275*, 2055–2065. [[CrossRef](#)]
25. Kaveh, A.; Ghazaan, M.I. Enhanced whale optimization algorithm for sizing optimization of skeletal structures. *Mech. Based Des. Struct. Mach.* **2017**, *45*, 345–362. [[CrossRef](#)]
26. Kaveh, A. Sizing optimization of skeletal structures using the enhanced whale optimization algorithm. In *Applications of Metaheuristic Optimization Algorithms in Civil Engineering*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 47–69.
27. Wu, J.; Wang, H.; Li, N. Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm. *Aerosp. Sci. Technol.* **2017**, *70*, 497–510. [[CrossRef](#)]
28. Barman, M.; Choudhury, N.B.D.; Sutradhar, S. A regional hybrid GOA-SVM model based on similar day approach for short-term load forecasting in Assam, India. *Energy* **2018**, *145*, 710–720. [[CrossRef](#)]
29. El-Fergany, A.A. Electrical characterisation of proton exchange membrane fuel cells stack using grasshopper optimiser. *IET Renew. Power Gener.* **2017**, *12*, 9–17. [[CrossRef](#)]
30. Łukasik, S.; Kowalski, P.A.; Charytanowicz, M. Data clustering with grasshopper optimization algorithm. In Proceedings of the 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), Prague, Czech Republic, 3–6 September 2017; pp. 71–74.
31. Rajput, N.; Chaudhary, V.; Dubey, H.M. Optimal generation scheduling of thermal System using biologically inspired grasshopper algorithm. In Proceedings of the 2017 2nd International Conference on Telecommunication and Networks (TEL-NET), Noida, India, 10–11 August 2017; pp. 1–6.
32. Zhang, X.; Miao, Q.; Zhang, H. A parameter-adaptive VMD method based on grasshopper optimization algorithm to analyze vibration signals from rotating machinery. *Mech. Syst. Signal Process.* **2018**, *108*, 58–72. [[CrossRef](#)]

33. Zhao, H.; Zhao, H.; Guo, S. Short-term wind electric power forecasting using a novel multi-stage intelligent algorithm. *Sustainability* **2018**, *10*, 881. [\[CrossRef\]](#)
34. Ahanch, M.; Asasi, M.S.; Amiri, M.S. A Grasshopper Optimization Algorithm to solve optimal distribution system reconfiguration and distributed generation placement problem. In Proceedings of the 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran, 22 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 659–666.
35. Ibrahim, H.T.; Mazher, W.J.; Ucan, O.N. A grasshopper optimizer approach for feature selection and optimizing SVM parameters utilizing real biomedical data sets. *Neural Comput. Appl.* **2019**, *31*, 5965–5974. [\[CrossRef\]](#)
36. Amaireh, A.A.; Alzoubi, A.; Dib, N.I. Design of linear antenna arrays using antlion and grasshopper optimization algorithms. In Proceedings of the 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, Jordan, 11–13 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
37. Hamad, A.; Houssein, E.H.; Hassanien, A.E. Hybrid grasshopper optimization algorithm and support vector machines for automatic seizure detection in EEG signals. In *International Conference on Advanced Machine Learning Technologies and Applications*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 82–91.
38. Hekimoğlu, B.; Ekinici, S. Grasshopper optimization algorithm for automatic voltage regulator system. In Proceedings of the 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE), Istanbul, Turkey, 3–5 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 152–156.
39. Heidari, A.A.; Faris, H.; Aljarah, I. An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Comput.* **2019**, *23*, 7941–7958. [\[CrossRef\]](#)
40. Faris, H.; Aljarah, I.; Mirjalili, S. Evolving radial basis function networks using moth–flame optimizer[M]. In *Handbook of Neural Computation*; Academic Press: Cambridge, MA, USA, 2017; pp. 537–550.
41. Hassanin, M.F.; Shueb, A.M.; Hassanien, A.E. Designing multilayer feedforward neural networks using multi-verse optimizer. In *Handbook of Research on Machine Learning Innovations and Trends*; IGI Global: Hershey, PA, USA, 2017; pp. 1076–1093.
42. Benmessahel, I.; Xie, K.; Chellal, M. A new evolutionary neural networks based on intrusion detection systems using multiverse optimization. *Appl. Intell.* **2018**, *48*, 2315–2327. [\[CrossRef\]](#)
43. Faris, H.; Hassonah, M.A.; Ala'M, A.Z. A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture. *Neural Comput. Appl.* **2018**, *30*, 2355–2369. [\[CrossRef\]](#)
44. Rosenberg, A.; Hirschberg, J. V-measure, A conditional entropy-based external cluster evaluation measure. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 28–30 June 2007; Association for Computational Linguistics: Stroudsburg, PA, USA, 2007; pp. 410–420.
45. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [\[CrossRef\]](#)
46. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [\[CrossRef\]](#)
47. Panwar, L.K.; Reddy, S.; Verma, A. Binary grey wolf optimizer for large scale unit commitment problem. *Swarm Evol. Comput.* **2018**, *38*, 251–266. [\[CrossRef\]](#)
48. Jayabarathi, T.; Raghunathan, T.; Adarsh, B.R. Economic dispatch using hybrid grey wolf optimizer. *Energy* **2016**, *111*, 630–641. [\[CrossRef\]](#)
49. Srikanth, K.; Panwar, L.K.; Panigrahi, B.K. Meta-heuristic framework, Quantum inspired binary grey wolf optimizer for unit commitment problem. *Comput. Electr. Eng.* **2018**, *70*, 243–260. [\[CrossRef\]](#)
50. Liu, C.A.; Wang, X.P.; Liu, C.Y.; Wu, H. Three-dimensional route planning for unmanned aerial vehicle based on improved grey wolf optimizer. *J. Huazhong Univ. Sci. Technol.* **2017**, *45*, 38–42.
51. Singh, N.; Singh, S.B. Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *J. Appl. Math.* **2017**, 2030489. [\[CrossRef\]](#)
52. Ab Rashid, M.F.F. A hybrid Ant-Wolf Algorithm to optimize assembly sequence planning problem. *Assem. Autom.* **2017**, *37*, 238–248. [\[CrossRef\]](#)
53. ElGayyar, M.; Emary, E.; Sweilam, N.H. A hybrid Grey Wolf-bat algorithm for global optimization. In *International Conference on Advanced Machine Learning Technologies and Applications*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 3–12.

54. Pan, J.S.; Dao, T.K.; Chu, S.C. A novel hybrid GWO-FPA algorithm for optimization applications. In *International Conference on Smart Vehicular Technology, Transportation, Communication and Applications*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 274–281.
55. Debnath, M.K.; Mallick, R.K.; Sahu, B.K. Application of hybrid differential evolution–grey wolf optimization algorithm for automatic generation control of a multi-source interconnected power system using optimal fuzzy–PID controller. *Electr. Power Compon. Syst.* **2017**, *45*, 2104–2117. [[CrossRef](#)]
56. Singh, N.; Singh, S.B. A novel hybrid GWO-SCA approach for optimization problems. *Eng. Sci. Technol. Int. J.* **2017**, *20*, 1586–1601. [[CrossRef](#)]
57. Zhang, X.; Kang, Q.; Cheng, J.; Wang, X. A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer. *Appl. Soft Comput.* **2018**, *67*, 197–214. [[CrossRef](#)]
58. Trivedi, I.N.; Jangir, P.; Kumar, A. A novel hybrid PSO-WOA algorithm for global numerical functions optimization. In *Advances in Computer and Computational Sciences*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 53–60.
59. Kaveh, A.; Rastegar Moghaddam, M. A hybrid WOA-CBO algorithm for construction site layout planning problem. *Sci. Iran.* **2018**, *25*, 1094–1104. [[CrossRef](#)]
60. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, India, 9–11 December 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 210–214.
61. Zelinka, I. SOMA—Self-Organizing Migrating Algorithm. In *New Optimization Techniques in Engineering*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 167–217.
62. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.
63. Digalakis, J.G.; Margaritis, K.G. On benchmarking functions for genetic algorithms. *Int. J. Comput. Math.* **2001**, *77*, 481–506. [[CrossRef](#)]
64. Molga, M.; Smutnicki, C. Test functions for optimization needs. *Test Funct. Optim. Needs* **2005**, *101*, 48.
65. Yang, X.S. Test problems in optimization. *arXiv* **2010**, arXiv:1008.0549.
66. Mirjalili, S.; Lewis, A. S-shaped versus V-shaped transfer functions for binary particle swarm optimization. *Swarm Evol. Comput.* **2013**, *9*, 1–14. [[CrossRef](#)]
67. Mirjalili, S.; Mirjalili, S.M.; Yang, X.S. Binary bat algorithm. *Neural Comput. Appl.* **2014**, *25*, 663–681. [[CrossRef](#)]
68. Price, K.V. Differential Evolution. In *Handbook of Optimization*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 187–214.
69. Van den Bergh, F.; Engelbrecht, A.P. A study of particle swarm optimization particle trajectories. *Inf. Sci.* **2006**, *176*, 937–971. [[CrossRef](#)]
70. Derrac, J.; García, S.; Molina, D. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [[CrossRef](#)]
71. Yang, L.; Guo, J.; Liu, Y. Three-Dimensional UAV Cooperative Path Planning Based on the MP-CGWO Algorithm. *Int. J. Innov. Comput. Inf. Control* **2020**, *16*, 991–1006.
72. Cheng, Z.; Tang, X.Y.; Liu, Y.L. 3-D path planning for UAV based on chaos particle swarm optimization. *Appl. Mech. Mater.* **2012**, *232*, 625–630. [[CrossRef](#)]
73. Hu, Z.H. *Research on Some Key Techniques of UAV Path Planning Based on Intelligent Optimization algorithm*; Nanjing University of Aeronautics and Astronautics: Nanjing, China, 2011.
74. Ye, Y.Y.; Min, C.P. A co-evolutionary method for cooperative UAVs path planning. *Comput. Simul.* **2007**, *24*, 37–39.