Article

A Comparative Study of Swarm Intelligence Algorithms for UCAV Path-Planning Problems

Haoran Zhu [1,†,‡], Yunhe Wang [2,*,‡], Zhiqiang Ma [2] and Xiangtao Li [1,*]

[1] School of Artificial Intelligence, Jilin University, Changchun 130012, China; zhuhr20@mails.jlu.edu.cn
[2] School of Computer Science and Technology, Northeast Normal University, Changchun 130117, China; mazqnenu@126.com
* Correspondence: wangyh082@nenu.edu.cn (Y.W.); lixt314@jlu.edu.cn (X.L.)
† Current address: School of Artificial Intelligence, Jilin University, Changchun 130012, China.
‡ These authors contributed equally to this work.

Abstract: Path-planning for uninhabited combat air vehicles (UCAV) is a typically complicated global optimization problem. It seeks a superior flight path in a complex battlefield environment, taking into various constraints. Many swarm intelligence (SI) algorithms have recently gained remarkable attention due to their capability to address complex optimization problems. However, different SI algorithms present various performances for UCAV path-planning since each algorithm has its own strengths and weaknesses. Therefore, this study provides an overview of different SI algorithms for UCAV path-planning research. In the experiment, twelve algorithms that published in major journals and conference proceedings are surveyed and then applied to UCAV path-planning. Moreover, to demonstrate the performance of different algorithms in further, we design different scales of problem cases for those comparative algorithms. The experimental results show that UCAV can find the safe path to avoid the threats efficiently based on most SI algorithms. In particular, the Spider Monkey Optimization is more effective and robust than other algorithms in handling the UCAV path-planning problem. The analysis from different perspectives contributes to highlight trends and open issues in the field of UCAVs.

Keywords: swarm intelligence; UCAV path-planning; optimization

Citation: Zhu, H.; Wang, Y.; Ma, Z.; Li, X. A Comparative Study of Swarm Intelligence Algorithms for UCAV Path-Planning Problems. *Mathematics* **2021**, *9*, 171. https://doi.org/10.3390/math9020171

Received: 11 November 2020
Accepted: 12 January 2021
Published: 15 January 2021

1. Introduction

Path-planning for uninhabited combat air vehicle (UCAV) task is one of the key problems in the UCAV system, which aims to find a path with the shortest distance while ensuring safety. A safe path denotes that the aircraft can reach the destination without hitting an obstacle or being detected by radar. The development of the UCAV system has been receiving increasing attention since it can accomplish difficult tasks in a complicated environment. In the previous study, many models have been taken to simulate the intelligent behavior that the aircraft might find a suitable path automatically [1,2]. Under that model, a series of methods to produce an optimal path have been proposed [3]. The angle, velocity, and height and many other factors should be involved in those methods. In [4], You et al., proposed a three-dimensional (3D) path-planning approach based on the situational space to provide the tactical requirements of UCAVs for tracking targets and avoiding collisions. In [5], a two-stage method for solving the terrain-following (TF)/terrain-avoidance (TA) path-planning problem for UCAV is presented, where the first stage of planning takes an optimization approach for generating a 2D path on a horizontal plane with no collision with the terrain, and in the second stage of planning, an optimal control approach is adopted to generate a 3D flyable path for the UCAV that is as close as possible to the terrain. In [6], the online collaborative path-planning method based on dynamic task allocation is proposed. However, they always suffer from the weakness of UCAVs' real-time response capability due to the high complexity.

Swarm intelligence algorithms with a new simple model for the path-planning task have been widely proposed to overcome the complexity. For example, Pehlivanoglu et al. [7] proposed a multi-frequency vibrational genetic algorithm to address the path-planning of UCAV. A new mutation application strategy and diversity variety are designed. In [8], Zhang et al. used the grey wolf optimizer to deal with UCAV path-planning using the traditional cost function model. In [9], an improved bat algorithm is employed for three-dimensional UCAV path-planning by Wang et al., in which BA is combined with differential evolution. Recently, Yi et al. proposed a quantum-inspired monarch butterfly optimisation [10] for the UCAV path-planning navigation problem. The worst individuals are updated by quantum operators to prevent being trapped in local optima, enhancing the search efficiency. In [11], Pan et al. proposed CIJADE to deal with the UCAV path-planning problem. It is a hybrid differential evolution algorithm that taking advantages of the modified CIPDE (MCIPDE) and modified JADE (MJADE) with great searchability. Moreover, Huang et al. [12] applied the SI algorithms to multi-model cooperative task assignment and path-planning of multiple UCAV formation, where the task assignment model of the UCAV formation is developed based on the flight characteristics of the UCAV formation and constraints in the battlefield. Dewangan et al. [13] applied the grey wolf optimization to the multi-UCAV path-planning problem in the three-dimensional environment. It expresses the trajectory of UCAV in the real 3D battlefield more vividly. Even more, in [14], Paszkiel et al. discussed the use of brain–computer interface to control unmanned aerial vehicles. A novel method is proposed to control the UCAV with a microcomputer connected with the human brain.

All those SI algorithms in solving such complex problems have common specific characteristics [15], such as sharing information between individuals to enhance the population quality [16,17]. However, each swarm intelligence algorithm has its own strengths and weaknesses [18]; different swarm intelligence algorithms provide various performance for cases with different UCAV scales. It is hardly believed that a single swarm intelligence algorithm can address all complex problems without detailed analysis from multiple perspectives [19]. Therefore, we provide a comprehensive comparison by studying similarities and differences between the twelve SI algorithms in UCAV path-planning. Particularly, a recently swarm intelligence algorithm called Spider monkey optimization [20] is applied to address the UCAV path-planning under the self-organization and division of labor. Moreover, we design thirty UCAV path-planning cases with different scales to demonstrate the performance of each algorithm. The effectiveness and robustness of each algorithm are further analyzed from different perspectives on those thirty different cases. Experimental results demonstrate that Spider monkey optimization can provide better performance over other state-of-the-art algorithms with good robustness.

The rest of this paper is organized as follows. Section 2 gives overviews of SI algorithms and Section 3 presents the model of path-planning and the probability density model. Performance comparisons are summarised in Section 4, and Section 5 presents conclusions and directions for future research.

2. Problem Formulation

Path-planning of Uninhabited Combat Air Vehicles (UCAV) is a numerical problem where emergencies should be considered. It aims to arrive at the destination safely by finding a superior path to avoid any threat, which accounts for artificial threats and natural constraints. The UCAV path-planning problem is usually represented by the classic 2D model, as shown in Figure 1. As depicted in this figure, we can clearly see that a path can be found to link up the starting point and the terminal end, and threats presented in circles are avoided. In the first, a segment ST that directly connected the starting and terminal point is drawn, then it is divided into $(D+1)$ equal parts by D perpendicular lines. After that, these D lines are taken as a new axis, onto which a series of points are set and connected one by one to form a path. Under this model, the whole path is divided into $D+1$ steps represented by a vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$, where x_{ij} is the jth dimension of the ith solution vector, and it indicates a discrete position of each step in the vertical axis. The way of representing the path (a solution) is using a vector in the form of $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$. For example, considering the coordinate $(0,0)$ is the starting point and $(T,0)$ is the terminal point, then the distance between the starting and the terminal point is T. A path with D steps can be performed by that vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$, thus the coordinate of the first step is $(\frac{T}{D}, x_{i1})$, the coordinate of the second step is $(\frac{T}{D} \times 2, x_{i2})$, and so on. With all the steps linked up, the whole path can be formed. Since the paths can not be away from the battle area, the restriction of each solution path on each case can be set manually related to the size of the battlefield.

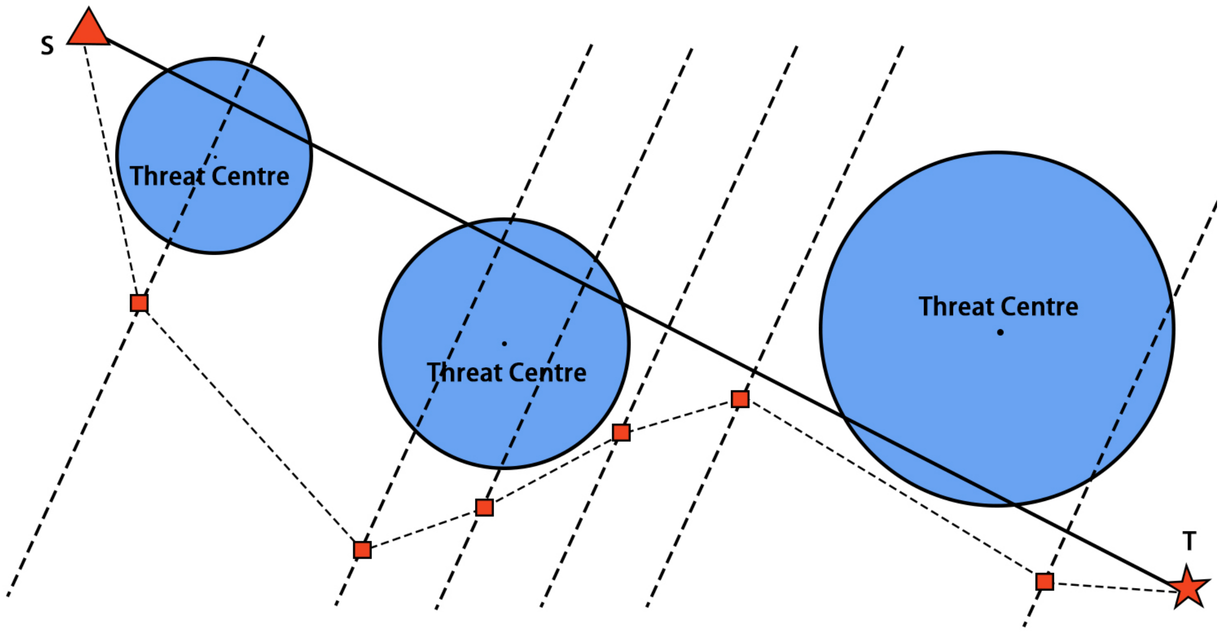

Figure 1. The schematic diagram for combat field modeling. The X and Y axles are the length and width of the horizontal battlefield. Note that all nodes are linked up to form a feasible path. During the flight, all threats are supposed to be avoid.

To measure those paths, a probability density model is formulated as Equation (1). Unlike the traditional cost function model, there is no distinct boundary where the damage risk remains zero. As distance increases, the probability of being attacked by a threat regularly decreases, but it's going to in no way be zero. $||D_{kj}||$ indicates the distance between the jth step and the kth threat center, δ is a parameter to control the shape of the density function:

$$Cost_{TI} = \sum_{k=1}^{n} exp(-\frac{\sum_{j=1}^{D} ||D_{kj}||}{.}\delta) \tag{1}$$

Indeed, the UCAV speed and the entire distance of a path should be taken into consideration. Simplify, it is assumed that the UCAV maintains a constant speed, and then the fuel consumption corresponds with the total distance of the path. Under that, the complete cost function can be described as:

$$Cost_{TI} = \gamma \cdot \sum_{k=1}^{n} exp(-\frac{\sum_{j=1}^{D} ||D_{kj}||}{\delta}) + (1-\gamma) \cdot \frac{\sum_{j=1}^{D} d_{ij}}{||ST||} \tag{2}$$

where γ is the weighting factor ranging in [0,1], and γ is set to 0.5. $||ST||$ is the length of the segment ST. $\sum_{j=1}^{D} d_{ij}$ refers to the length of the whole flight path, and d_{ij} is the length of the jth step of the ith solution. Since each step is connected directly as a segment, d_{ij} can be calculated by the pythagorean theorem:

$$d_{ij} = \sqrt{a_{ij}^2 + b_{ij}^2} \tag{3}$$

where a_{ij} and b_{ij} are the horizontal and vertical projection of each step. In this study, Equation (2) is adopted as the objective function, a solution with smaller objective value denotes a path with better quality.

3. Methodology

Many computational methods have been proposed for addressing path-planning of UCAV [1–3]. Unfortunately, those computational methods often suffer from numerical instability and premature convergence due to their inefficient search capabilities and high complexity. Recently, swarm intelligence (SI) algorithms have been proven to be a competitive approach to deal with such difficult problems [21–24]. Generally speaking, SI algorithms are inspired by nature or population-based. They are heuristic methods that exchange information among the entire population iteratively. However, each swarm intelligence algorithm has its own strengths and weaknesses; different swarm intelligence algorithms provide various performance on different UCAV scales due to their algorithmic parameters based on the problem characteristics. To demonstrate the performance of different swarm intelligence algorithms in the UCAV path-planning problem, a comparison of 12 SI algorithms is studied. The mechanical characteristics and parameters of each algorithm for the UCAV path-planning problem are listed in Table 1. Moreover, four algorithms including Grey Wolf Optimizer, Firefly Algorithm, Harmony Search, Spider Monkey Optimization are introduced.

Table 1. An algorithmic parameter summary of twelve swarm intelligence (SI) algorithms.

Algorithm	Biological Motivation	Parameter	Exploration Mechanism
ABC [25]	Honey Collecting Behavior of Bee Colony	trial = 0.1·FEs	An employed bee denotes a feasible solution, the unemployed bees do local search.

BA [26]	Echolocation of bats	Fmin = 1
		Fmax = 4
		A = 2.5
		r0 = 1
		alpha = 0.97
		gamma = 2.5
			Bats fly randomly with a velocity at a position with a fixed frequency, varying wavelength and loudness to search for prey.

CS [27]	brood parasitism	beta = 1.5	Eggs with better quality have more chances to survive.

DE [28]	Mutation and crossover of chromosome	F = 0.5
		CR = 0.1
			All the individuals conduct the mutation and crossover operators

FA [29]	The blinking behavior of fireflies	alpha = 0.5
		beta = 1
		gamma = 0.01
		theta = $10^{(-5/FEs)}$
			A firefly can be attracted by others that are brighter than it.

GCMBO [30]	The migration behavior of monarch butterflies	p = 0.5
		peri = 0.5
		beta = 1.5
		Smax = 1
		BAR = 0.8
			The monarch butterflies in Land 1 and Land 2 are composed of the whole population, and the worse subgroup moves to the better one randomly.

GWO [31]	Encircling and hunting behavior of gray wolves	None	Take average of the three leader wolves.

Table 1. *Cont.*

Algorithm	Biological Motivation	Parameter	Exploration Mechanism

HS [32]	Search for better harmony	HMCR = 0.9
		PAR = 0.2
		BAR = L·0.01
			A random dimension of the solution vector is picked for updating.

MSA [33]	Phototropism of moths	Smax = 1
		beta = 0.5
		phy = $(1+\sqrt{5})/2$
			A moth can be attracted by lights. When it's far away, it will fly straight, or it will do random fly.

PSO [25]	Bird migration	c1 = 1.4962
		c2 = 1.4962
		w = 0.7298
			Individuals update their positions by their velocity and current positions.

SMO [20]	Fission-fusion system of spider monkey	MG = N/4
		GLL = N
		LLL = D·N
		pr = 0.375
			Divide the group into a certain number of subgroups if stagnated.

WOA [34]	Encircling and preying behavior of whales	a = 5 -iter·/FEs
		b = 1.5
		A = 2·a·rand-a
		C = 2·rand
		I = rand·2-1 p = rand
			Do exploration by spiral updating.

3.1. Grey Wolf Optimizer

Grey wolf optimizer was inspired by the special predatory behavior of grey wolves, which has a hierarchy system in the population and encircles their prey in a clear labor division. The leaders are called the alphas. They are mostly responsible for making decisions about hunting, sleeping place and so on. The second level in the hierarchy is beta, which helps the alpha make decisions, like the military adviser. The lowest level, omega, plays the role of the scapegoat. They submit to all the other dominant wolves. If a wolf is not an alpha, beta or omega, it's called a delta. Delta wolves have to submit to alphas and betas, but they dominate the omega.

In GWO, it considers the best solution as the alpha (α), then the second and the third best solutions are named beta (β) and delta (δ) respectively. The rest individuals in the population are omegas (ω), which are guided by α, β and δ during the hunting process. In each iteration, each individual is updated by:

$$\begin{gathered} D_\alpha = |C_1 \cdot X_\alpha - X_t|, D_\beta = |C_2 \cdot X_\beta - X_t|, D_\delta = |C_3 \cdot X_\delta - X_t| \\ X_1 = X_\alpha - A_1 \cdot (D_\alpha), X_2 = X_\beta - A_2 \cdot (D_\beta), X_3 = X_\delta - A_3 \cdot (D_\delta) \\ X_{t+1} = \frac{X_1 + X_2 + X_3}{3} \end{gathered} \tag{4}$$

where t indicates the current iteration. A_i and C_i are coefficient vectors, $X_\alpha, X_\beta, X_\delta$ are the three leader wolves' vectors, and X is the position vector of a grey wolf. A_i and C_i are calculated as follows:

$$\begin{gathered} A = 2a \cdot r_1 - a \\ C = 2 \cdot r_2 \end{gathered} \tag{5}$$

where a is linearly decreased from 2 to 0 over iterations, r_1 and r_2 are random vectors in [0,1]. Equation (4) mathematically models the behaviour of encircling and hunting.

3.2. Firefly Algorithm

Firefly algorithm is an effective swarm intelligence method based on the behavior of fireflies that brighter fireflies would attract other fireflies. As observed from the natural world, a group of fireflies would always fly towards the brightest one. Brightness measures the quality of fireflies, therefore the brightness or light intensity in the algorithm can be represented by the objective function. To simplify, all fireflies are set unisex so that one firefly is attracted to other fireflies regardless of their sex. Attractiveness is related to their distance, which can be calculated by the equation below:

$$attr = \beta_0 \cdot exp(-gamma \cdot dist^2) \tag{6}$$

where β_0 is to control the size of attractiveness, and β_0 is set to 1 for the UCAV path-planning problem. $gamma$ is the consumption factor and set to 0.01. $dist$ is the cartesian distance of two fireflies. For firefly i, it checks every firefly in the whole group. If it finds a brighter or better firefly j, firefly i is attracted and moves towards firefly j with a random walk in all dimensions by:

$$\begin{aligned} X_i = X_i + attr \cdot (X_j - X_i) + step; \\ step = alpha \cdot (rand(1, D) - 0.5) \cdot L \end{aligned} \tag{7}$$

where $step$ is the randomization walk for local searching. $alpha$ is a coefficient decreasing with iterations, $rand(1, D)$ is a random vector with D dimensions ranging in [0,1] and L is the difference between the upper and lower bounds.

3.3. Harmony Search

Harmony search algorithm is a heuristic algorithm derived from an artificial phenomenon found in music performance to search for better harmony. A harmony is composed of many different kinds of instruments, and only the same instruments can adjust from each other. For instance, the piano part of harmony is subpar, and the piano part can only improve itself from the experience of pianos in other similar harmonies. Aesthetic estimation measures the quality of a harmony. Practice by practice makes a fantastic harmony, just as the values for better objective function evaluation can be improved iteration by iteration.

In Harmony Search (HS), a new harmony is improvised from the population in each iteration. The way each dimension updates depends on a probability $HMCR$. If a random number $rand$ is smaller than $HMCR$, the certain dimension of a randomly selected harmony is used for the newly generated harmony. Or it is generated randomly. For example, suppose there are a total of ten vectors in the population. For the $j^t h$ dimension of the new harmony, the third vector in the population is selected randomly for the new harmony. After that, a further adjustment is executed by probability PAR. If $rand < PAR$, the adjustment proceeds as equation below:

$$X_i^j = X_i^j + 2 \cdot rand(0, 1) \cdot BAR - BA, R \tag{8}$$

where BAR is the difference between the upper and lower bounds. Or no change is made. This step guarantees the global search ability of HS. If the newly created harmony is better than the worst harmony in the population, it will replace the worst one. An overview of HS is summarized in [35].

3.4. Spider Monkey Optimization

Spider Monkey Optimization (SMO) is a novel optimization algorithm that is inspired by the fission-fusion social system of the spider monkey. These smart creatures have an impressive mechanism to find the best food source: the leader of the population, usually a female monkey, leads the group. However, if she can't find a sufficient target for the whole group, she divides the whole group into smaller subgroups that explore respectively. Each subgroup has a leader. Communications are made among the individuals in every subgroup and with other subgroup individuals as follows:

$$X_i = X_i + rand \cdot (X_{LL} - X_i) + (rand - 0.5) \cdot 2 \cdot (X_k - X_i) \tag{9}$$

where *rand* is a random number in the range [0,1] and X_{LL} is the local leader in the subgroup. X_k is a randomly selected individual in the entire group.

If the global leader does not update herself in a certain number of iterations, namely *GlobalLeaderLimit*, the whole group breaks into certain numbers of subgroups. If a local leader remains the same for a certain number of iterations, namely *LocalLeaderLimit*, the subgroup is re-generated randomly or simply attracted by the global leader by:

$$X_i = X_i + rand \cdot (X_{GL} - X_i) + (rand - 0.5) \cdot 2 \cdot (X_k - X_i) \tag{10}$$

where X_{GL} is the global leader. The number of subgroups has a limitation MG. If MG is reached, all the subgroups reunion into an entire group.

3.5. Proposed Framework for UCAV Path-Planning Problem

In this study, the UCAV path-planning task is a numerical problem, where a superior path with the lowest objective value needs to be found. It has been proven that SI methods are effective approaches to deal with optimal problems. First, the population composed of some individuals is randomly initialized, and the probability density model evaluates each individual. Then the process of different search strategies proceeds to optimize the population after the iterations. The SMO algorithm is used to address the UCAV path-planning problem.

3.5.1. Initialization

In Section 2, the 2D model transforms the battlefield to the top view of the map, and a path divided by several steps is linked up from the starting point and the terminal ending. Thus, the path can be represented by a vector $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,D})$, where $i = 1, 2, ..., N$ and D denotes the number of steps in each path, and $x_{i,j}$ indicates the position of the jth step of the ith individual in the whole population.

The population consists of a certain number of individuals (N). The initial population covers the entire battlefield as much as possible by randomizing the paths within the upper and lower bounds $X_{i,j}^{max}$ and $X_{i,j}^{min}$ of the map. Therefore, the population is initialized as follows:

$$\begin{gathered} X_{i,j} = X_{i,j}^{min} + rand(0,1) \times (X_{i,j}^{max} - X_{i,j}^{min}), \\ j = 1, 2, ..., D, i = 1, 2, ..., N \end{gathered} \tag{11}$$

3.5.2. Local Leader Phase

After initialization, each path in every subgroup updates its current position based on the experience of its local leader and local group paths in local lea, by:

$$X_{new_{i,j}} = X_{i,j} + U(0,1) \times (X_{LL,j} - X_{i,j}) + U(-1,1) \times (X_{r,j} - X_{i,j}) \tag{12}$$

where $X_{LL,j}$ is the jth step of the local leader under the current subgroup. X_{rj} is a randomly chosen path among the subgroup and $r \neq i$, and $U(a,b)$ is a random number ranging in [a,b] that obeys uniform distribution. Algorithm 1 summarised the local leader phase of SMO for UCAV path-planning problem. $pr \in [0.1, 0.8]$ is the perturbation rate which controls the frequency of perturbation in the current position. After that, the objective value is calculated by the probability density model, and the lower the objective value of a path is, the more qualified the path is. According to the greedy rule, if the newly updated path is better than the old one, the path is replaced with the new one.

Algorithm 1 Local leader phase

```
for each path $X_i \in k$th subgroup do
    for each $j \in$ 1,2,...,D do
        if $U(0,1) \geq pr$ then
            update $X_{new_{ij}}$ by $X_{new_{i,j}} = X_{i,j} + U(0,1) \times (X_{LL,j} - X_{i,j}) + U(-1,1) \times (X_{r,j} - X_{i,j})$
        else
            $X_{new_{ij}} = X_{ij}$
        end if
    end for
end for
```

3.5.3. Global Leader Phase

After the local leader phase, all the paths update their positions using the experience of the global leader and local group members by $prob_i$. The global leader phase is summarized in Algorithm 2, from which we can see that the better paths with higher $prob_i$ are more likely further to update themselves. If $U(0,1) < prob_i$, the evolutionary search strategy proceeds and can be described as follows:

$$X_{new_{i,j}} = X_{i,j} + U(0,1) \times (X_{GL,j} - X_{i,j}) + U(-1,1) \times (X_{r,j} - X_{i,j}) \tag{13}$$

where $X_{GL,j}$ is the jth step of the global leader and $j \in \{1, 2, ..., D\}$ is a randomly chosen step. The $prob_i$ could be calculated by:

$$prob_i = \frac{fitness_i}{\sum_{i=1}^{N} fitness_i}, \tag{14}$$

where $fitness_i$ is the $fitness$ value of the ith path. $prob_i$ can be calculated by anyway related to the objective value of the paths.

Algorithm 2 Global leader phase

```
count = 0
while count < groupsize do
    for each member $X_i \in$ group do
        if $U(0,1) < prob_i$ then
            count = count + 1
            Randomly choose $j$ and $r$ and $r \neq i$
            Update $X_{new_{i,j}}$ by $X_{new_{i,j}} = X_{i,j} + U(0,1) \times (X_{GL,j} - X_{i,j}) + U(-1,1) \times (X_{r,j} - X_{i,j})$
        end if
    end for
end while
```

Further, the objective of the newly generated path is compared with the old one and the better one is adopted.

3.5.4. Local Leader Decision Phase

In order to prevent stagnation, the local leader decision phase is proposed. The local leader is the best path in the current subgroup, and if any newly updated paths are found better than the local leader, the better path is maintained. If any local leader is not optimized for a threshold called $LocalLeaderLimit$, then all paths in this subgroup update their positions by either rerandom generation or by using combined information from the global leader and the local leader in this group, which depends on the probability pr. In this way, any paths have chances to move closer to the best path in the whole group, which guarantees the search capability. The local leader decision phase is summarized in Algorithm 3.

Algorithm 3 Local leader decision phase

```
if $Locallimitcount > Localleaderlimit$ then
    $Locallimitcount = 0$
    for each path in this subgroup do
        if $U(0,1) \geq pr$ then
            re-generate $X_{i,j}$
        else
            $X_{new_{i,j}} = X_{i,j} + U(0,1) \times (X_{GL,j} - X_{i,j}) + U(0,1) \times (X_{i,j} - X_{LL,j})$
        end if
    end for
end if
```

3.5.5. Global Leader Decision Phase

The global leader decision phase is another method to prevent stagnation. The global leader is the best path in the entire population; thus, it has the lowest objective value and the smoothest route. Similar to the local leader decision phase, if any newly updated paths are better than the global leader, it will replace the global leader. Therefore, there might be a situation where the global leader is trapped in stagnation. The global leader decision phase is summarised in Algorithm 4. If the global leader is not updated after a certain number of iterations, namely $GlobalLeaderLimit$, the global leader divides the population into smaller subgroups, or simply reunion all the subgroups into a single group and start a new round of search, which depends on the current number of subgroups MG. The global leader decision phase guarantees the ability of local search, which can prevent the path from trapping into local optimum, and that's the reason why paths provided by SMO are always smoother than others'.

Algorithm 4 Global leader decision phase

```
if $Globallimitcount > Globalleaderlimit$ then
    $Globallimitcount = 0$
    if $Numberofgroups < MG$ then
        Divide the population into smaller subgroups
    else
        Reunion all the subgroups into a single group
    end if
    Update Local and Global leaders position
end if
```

3.5.6. Time Complexity Analysis

In this section, we analyze the time complexity of SMO for addressing UCAV path-planning problem. Each individual in the population needs to be traversed in local leader phase, and N of individuals are further updated, therefore the time complexity is $O(2ND)$. After that, the population needs to be evaluated with all probability density model threats, as shown in Equation (2). The time complexity of the objective function evaluation is $O(KD)$, where K is the number of threats. To sum up, the overall time complexity is $O(t(2ND + NKD))$, where t is the number of iterations, N is the population size, D is the number of dimensions.

4. Experimental Results and Discussion

Twelve algorithms are used for comparison in this work, including artificial bee colony algorithm (ABC), bat algorithm (BA), cuckoo search (CS), differential evolution (DE), firefly algorithm (FA), monarch butterfly optimization with greedy strategy (GCMBO), grey wolf optimization (GWO), harmony search (HS), moth search algorithm (MSA), particle swarm optimization (PSO), spider monkey optimization (SMO), and whale optimization algorithm (WOA).

4.1. Cases Design

We have adopted 30 different cases including 10 small-scale (D = 20), 10 medium-scale (D = 30), and 10 large-scale cases (D = 60). Each case contains a certain number of threats represented by circles, as shown in Figure 2, and an outstanding path should avoid the threats and arrives at the terminal ending safely.

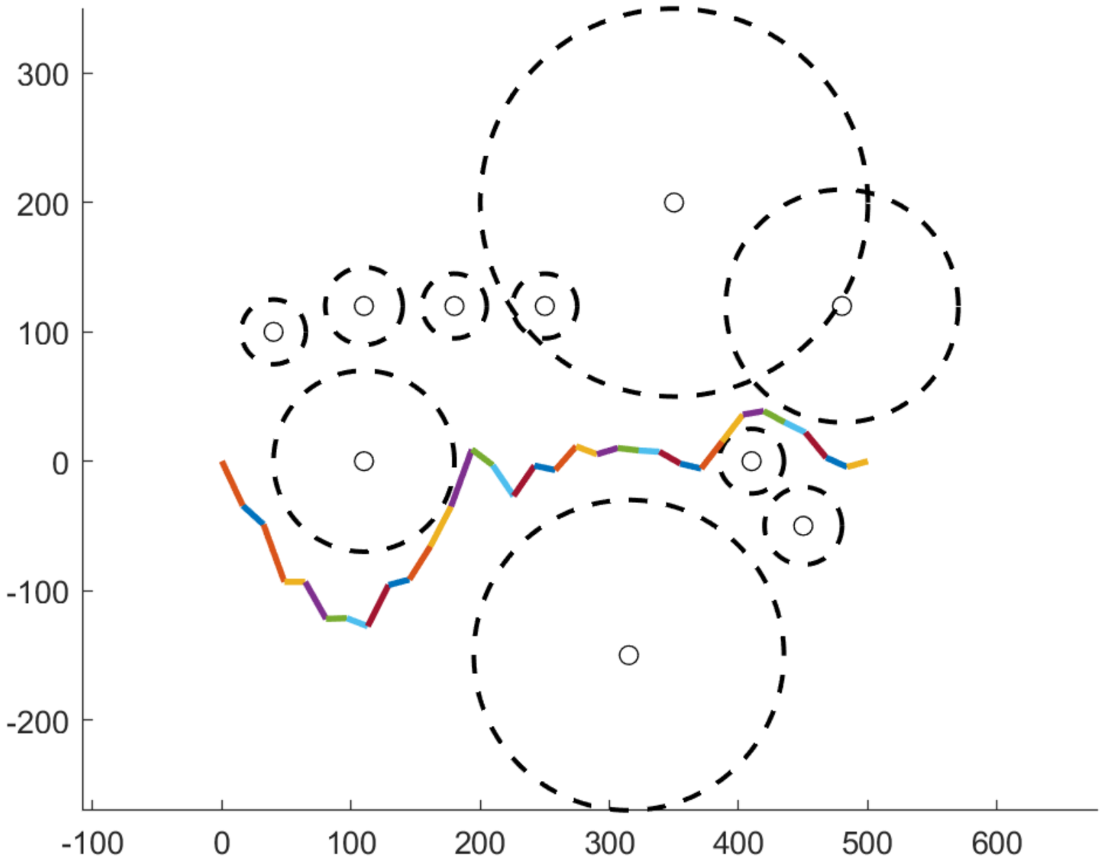

Figure 2. A case with ten threats. The horizontal axis is the length of the battlefield and the vertical axis is the width of the battlefield. The unit can be any unit that represents the length.

Moreover, we intentionally design some cases where some of the threats in the map are at an intersection to demonstrate the robustness of our proposed model.

4.2. Parameter Setting

The comparisons are made for thirty cases with the probability density model to optimize the objective function. For all swarm intelligence algorithms, the number of population N is set to 20 for small-scale and 30 for medium-scale and large-scale cases. Other parameters of different SI algorithms are listed in Table 1. In SMO, it can be noticed that $pr \in [0.1, 0.8]$ is the perturbation rate, which controls the frequency of perturbation in the current position. If the value of pr is too large, the result might be converged after a certain number of iterations. Conversely, if pr is too small, the results can get converged after a less number of iterations with worse value. Therefore, we selected pr from $[0.2, 0.375, 0.5, 0.625, 0.8]$. By conducting lots of prior experiments, the best result for SMO is obtained when $pr = 0.375$. Thus we set pr to 0.375 in SMO. The number of fitness evaluations depends on the operators and the population update model. Different operators of the algorithms can lead to very different numbers of fitness evaluations per iteration. Therefore, the number of fitness evaluations is adopted as the stopping criteria instead of the number of iterations [22]. In our study, we set the $FEs = 120 \times N$. To make statistical analysis, each algorithm runs twenty-five times independently. The average and standard deviation results over 25 independent runs are calculated.

4.3. Performance Comparison on Small-Scale Cases

In this section, ten small-scale cases for D = 20 are employed to demonstrate the effectiveness of each SI algorithm for the UCAV path-planning problem. We take Case 8 in those cases for an example to show the best paths provided by each method over 25 independent runs. The comparative performance of each algorithm on Case 8 is visualized in Figures 3–5. The horizontal axis is the length of the battlefield and the vertical axis is the width of the battlefield. The unit can be any unit that represents the length. As shown in Figures 3–5, it demonstrates that most SI algorithms can successfully avoid the obstacles and reach the terminal ending safely. For each case, the best value, the worst value, average value, and standard deviation are summarized in Table 2. From Table 2, it is observed that SMO can obtain better result (on average, worst) than the other algorithms on all the cases. For Case 2, BA and FA can provide the similar best values with SMO. For Case 3, FA and SMO achieve the similar best values. In particular, SMO shows its strong robustness for four cases including Cases 3, 4, 7, and 8 since its std deviation is zero over multiple runs. For Case 9, HS can reach a slightly smaller std deviation than SMO. However, SMO performs better on best, worst, and average value than HS across 25 times. In terms of those 10 small-scale cases, SMO reaches the best average result for Case 2 and the worst average result for Case1 when multiple runs are made. Moreover, SMO can reach the best value to 1.47449 for Case 2 while WOA can only achieve to 1.73079. Based on those analysis, we can conclude that paths made by SMO have the best performance and the least standard deviation, which reflects that SMO has the best optimization stability for UCAV path-planning problems.

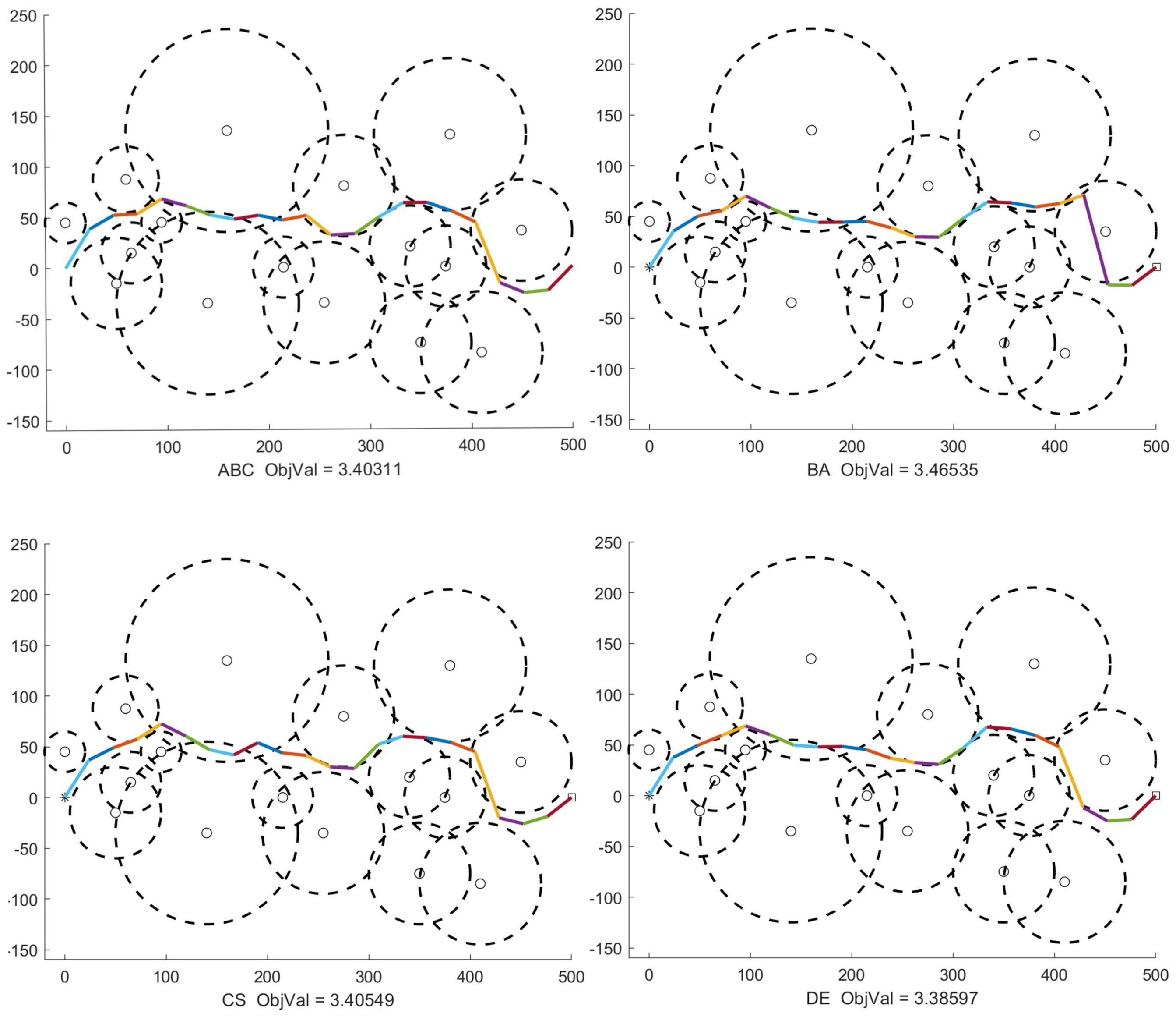

Figure 3. The comparison performance of different swarm intelligence algorithms including artificial bee colony algorithm (ABC), bat algorithm (BA), cuckoo search (CS), differential evolution (DE) for Case 8 with $D = 20, N = 40$.

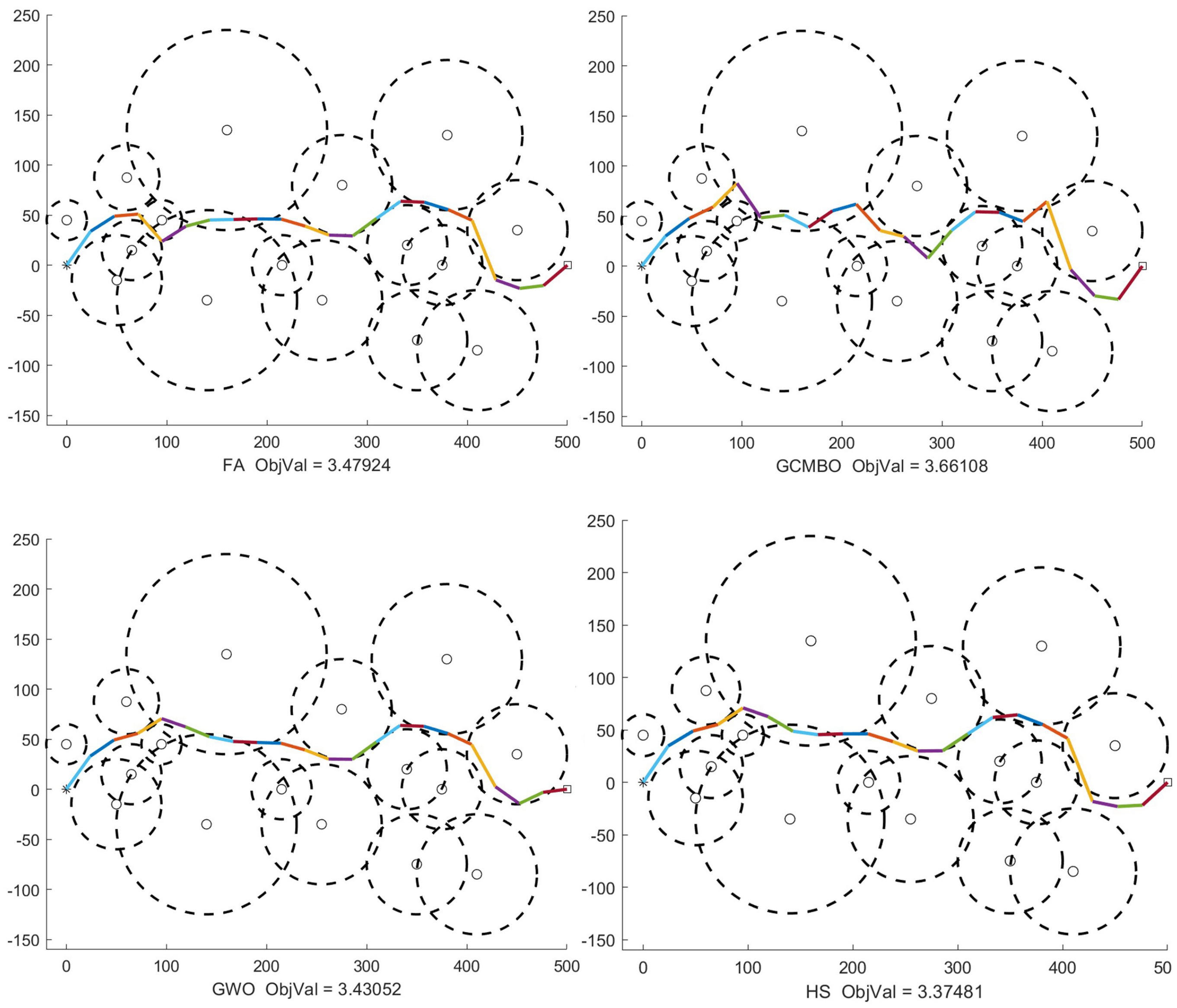

Figure 4. The comparison performance of different swarm intelligence algorithms including firefly algorithm (FA), monarch butterfly optimization with greedy strategy (GCMBO), grey wolf optimization (GWO), harmony search (HS) for Case 8 with $D = 20, N = 40$.

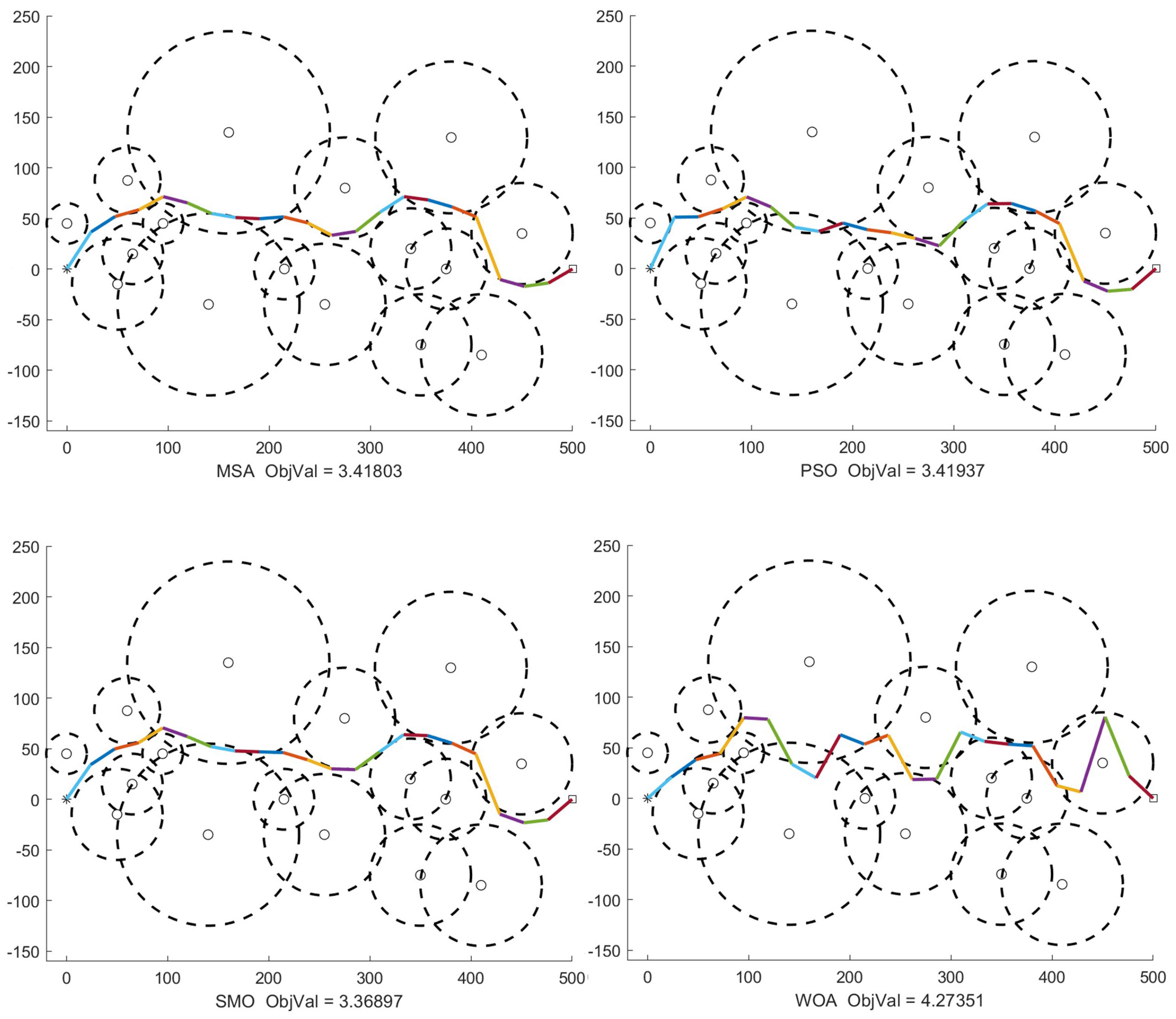

Figure 5. The comparison performance of different swarm intelligence algorithms including moth search algorithm (MSA), particle swarm optimization (PSO), spider monkey optimization (SMO), and whale optimization algorithm (WOA) for Case 8 with $D = 20, N = 40$.

Table 2. Performance comparison of different swarm intelligence algorithms for small-scale uninhabited combat air vehicle (UCAV) path-planning. The best, worst, average and standard deviation over 25 runs of each algorithm for ten cases are provided. The best values are in bold.

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Case 7**	**Case 8**	**Case 9**	**Case 10**	
ABC	4.02419	1.55182	1.69836	2.00564	2.10935	2.08304	3.17832	3.40311	3.03541	2.78953	Best
	4.22836	1.71690	1.87450	2.34539	2.37855	2.39144	3.53045	3.69164	3.45004	3.15891	Worst
	4.14491	1.61041	1.75736	2.15096	2.25920	2.22037	3.32170	3.57828	3.22825	2.96556	Average
	0.05401	0.03998	0.04304	0.08755	0.06807	0.07445	0.09690	0.07010	0.10477	0.09753	Std deviation
BA	4.06832	**1.47449**	1.69652	2.08734	2.10935	2.22917	3.08888	3.46535	2.98014	2.92027	Best
	4.55974	1.81447	2.28030	2.82178	2.37855	2.85678	3.97422	4.36706	4.85309	3.67888	Worst
	4.26404	1.64581	1.96575	2.36674	2.25920	2.51155	3.60318	3.91960	3.84687	3.36492	Average
	0.13025	0.09027	0.15832	0.22850	0.06807	0.14218	0.25250	0.25231	0.36851	0.19427	Std deviation
CS	4.03220	1.49796	1.64508	1.96107	2.10634	1.99048	3.11039	3.40549	2.95024	2.79610	Best
	4.18388	1.59921	1.69286	2.09279	2.27114	2.29661	3.17957	3.52887	3.13843	2.98839	Worst
	4.09087	1.54096	1.66259	1.99728	2.16817	2.10217	3.15049	3.44878	3.02180	2.85613	Average
	0.04843	0.02967	0.01233	0.02792	0.03341	0.06515	0.01860	0.03094	0.04584	0.03932	Std deviation
DE	4.02880	1.50325	1.63592	1.94842	2.08429	2.11028	3.09575	3.38597	2.93400	2.78577	Best
	4.12530	1.61329	1.65845	1.98861	2.17788	2.27987	3.11684	3.44025	3.03077	2.85924	Worst
	4.07083	1.53355	1.64587	1.96551	2.12827	2.19684	3.10590	3.40691	2.97620	2.81632	Average
	0.02170	0.02514	0.00658	0.01144	0.02312	0.05239	0.00563	0.01165	0.02309	0.02036	Std deviation
FA	3.98033	**1.47449**	**1.61806**	1.95343	2.08232	1.97192	3.11165	3.47924	2.91277	2.77118	Best
	4.41767	1.68793	1.87802	2.38280	2.16536	2.37051	3.91961	3.94338	3.48155	2.90343	Worst
	4.18981	1.57556	1.66160	2.10564	2.08971	2.15159	3.62811	3.60082	3.31600	2.85962	Average
	0.12989	0.05752	0.09639	0.11432	0.01857	0.11398	0.17178	0.09872	0.16063	0.05177	Std deviation
GCMBO	4.09861	1.67749	1.85437	2.18445	2.36318	2.30052	3.41527	3.66108	3.41110	3.14582	Best
	4.61633	1.96159	2.40352	2.85107	3.94095	2.84723	4.20496	4.46385	4.13645	3.76983	Worst
	4.35093	1.81784	2.06054	2.50667	2.82242	2.61433	3.69809	4.03578	3.78223	3.44099	Average
	0.13047	0.07070	0.14633	0.16969	0.32262	0.13625	0.23585	0.16473	0.25659	0.18228	Std deviation
GWO	3.99392	1.51986	1.61942	1.93603	2.03664	1.94153	3.13821	3.43052	2.92174	2.76177	Best
	4.30288	1.68710	2.04082	2.44024	2.29864	2.33305	3.86821	4.04737	3.79556	3.19019	Worst
	4.11663	1.59373	1.69373	2.06802	2.10303	2.07372	3.43738	3.64063	3.18509	2.90663	Average
	0.08042	0.04810	0.11241	0.11212	0.07381	0.10230	0.18621	0.17303	0.29254	0.12241	Std deviation

Table 2. *Cont.*

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Case 7**	**Case 8**	**Case 9**	**Case 10**	
HS	3.96302	1.47787	1.62287	**1.91028**	2.04462	1.94352	3.09337	3.37481	2.91970	2.76248	Best
	4.08843	1.64180	1.66543	1.95335	2.13986	2.09013	3.11112	3.49135	2.97966	2.95445	Worst
	4.02537	1.51918	1.63472	1.93401	2.08287	2.00332	3.09912	3.38584	2.94064	2.84744	Average
	0.04445	0.05018	0.00879	0.01022	0.02929	0.05375	0.00433	0.02250	**0.01508**	0.05337	Std deviation
MSA	4.04798	1.53099	1.62493	1.97107	2.05623	1.99782	3.17287	3.41803	2.97542	2.85214	Best
	4.50929	1.84316	1.95150	2.69312	2.71180	2.40953	4.12896	4.54438	3.83439	3.39389	Worst
	4.22686	1.67300	1.75042	2.24537	2.18386	2.21390	3.59242	3.68847	3.40229	3.09557	Average
	0.10797	0.08239	0.12822	0.18634	0.16910	0.12837	0.28264	0.25160	0.23950	0.16223	Std deviation
PSO	4.20501	1.51803	1.65574	2.04092	2.11362	2.07533	3.27830	3.41937	3.04995	2.91570	Best
	4.46044	1.73712	1.88085	2.37730	2.48445	2.37978	3.93095	3.74395	3.47818	3.34519	Worst
	4.34820	1.61448	1.71962	2.24712	2.20476	2.25161	3.56903	3.58126	3.19644	3.05964	Average
	0.07488	0.05803	0.05977	0.10038	0.08953	0.08583	0.14681	0.08585	0.11082	0.10172	Std deviation
WOA	4.62953	1.73079	2.13197	2.74961	2.57314	2.50992	3.89892	4.27351	3.85183	3.48746	Best
	5.20931	2.31805	2.62385	3.41324	3.68988	3.31362	4.74324	5.32713	5.07037	4.47647	Worst
	4.89909	2.05008	2.35926	3.04231	3.16261	2.89391	4.32977	4.75819	4.45843	4.00377	Average
	0.15727	0.14920	0.15173	0.17118	0.34023	0.18900	0.23192	0.27605	0.32343	0.26026	Std deviation
SMO	**3.95556**	**1.47449**	**1.61806**	1.91451	**2.03334**	**1.93365**	**3.08887**	**3.36897**	**2.82486**	**2.71818**	Best
	3.98066	**1.50205**	**1.61807**	**1.91451**	**2.03753**	**1.97235**	**3.08887**	**3.36897**	**2.90891**	**2.77118**	Worst
	3.95903	**1.47564**	**1.61806**	**1.91451**	**2.03366**	**1.94550**	**3.08887**	**3.36897**	**2.86160**	**2.74048**	Average
	0.00708	**0.00550**	**0.00000**	**0.00000**	**0.00084**	**0.01591**	**0.00000**	**0.00000**	0.03739	**0.01317**	Std deviation

4.4. Performance Comparison on Medium-Scale Cases

This section is devoted to show the comparative performance of each algorithm on ten medium-scale cases (D = 30). Case 8 of those ten cases is taken as an example to present the best paths gained by each algorithm over 25 runs. The path results are shown in Figures 6–8. As depicted in Figures 6–8, there is a noticeable difference between each path. ABC, BA, CS, GCMBO, and PSO generate less smooth path than those made by the other algorithms. Some paths of them can be trapped in local optimum. It can be noted that SMO can perform the smoothest path among all the comparative algorithms. To further demonstrate the effectiveness of each algorithm, Table 3 summarizes the cost objective values of each algorithm on each case. From Table 3, we can find that SMO has the best performance while WOA performs the worst. Across all those ten cases, SMO provides the best results on average, best, and worst with the least std deviation. FA can obtain the similar result on Case 3 with SMO. For HS, it achieves the similar std deviation with SMO on Case 6. In terms of Cases 7 and 8, SMO can always produce the best results over 25 times. The average results for Cases 7 and 8 gained by SMO are 4.04452 and 4.42171 respectively while WOA only reaches to 6.41485 and 6.93741 on Cases 7 and 8. It demonstrates that SMO can enhance the performance may due to the division characteristic of SMO. Besides, the minimum and maximum best result are 1.6787 and 5.2963 generated by SMO on Cases 2 and 1 with different threat conditions. In conclusion, it can be summarized that SMO is more effective than other algorithms and suitable for solving the UCAV path-planning problem in medium-scale with high stability.

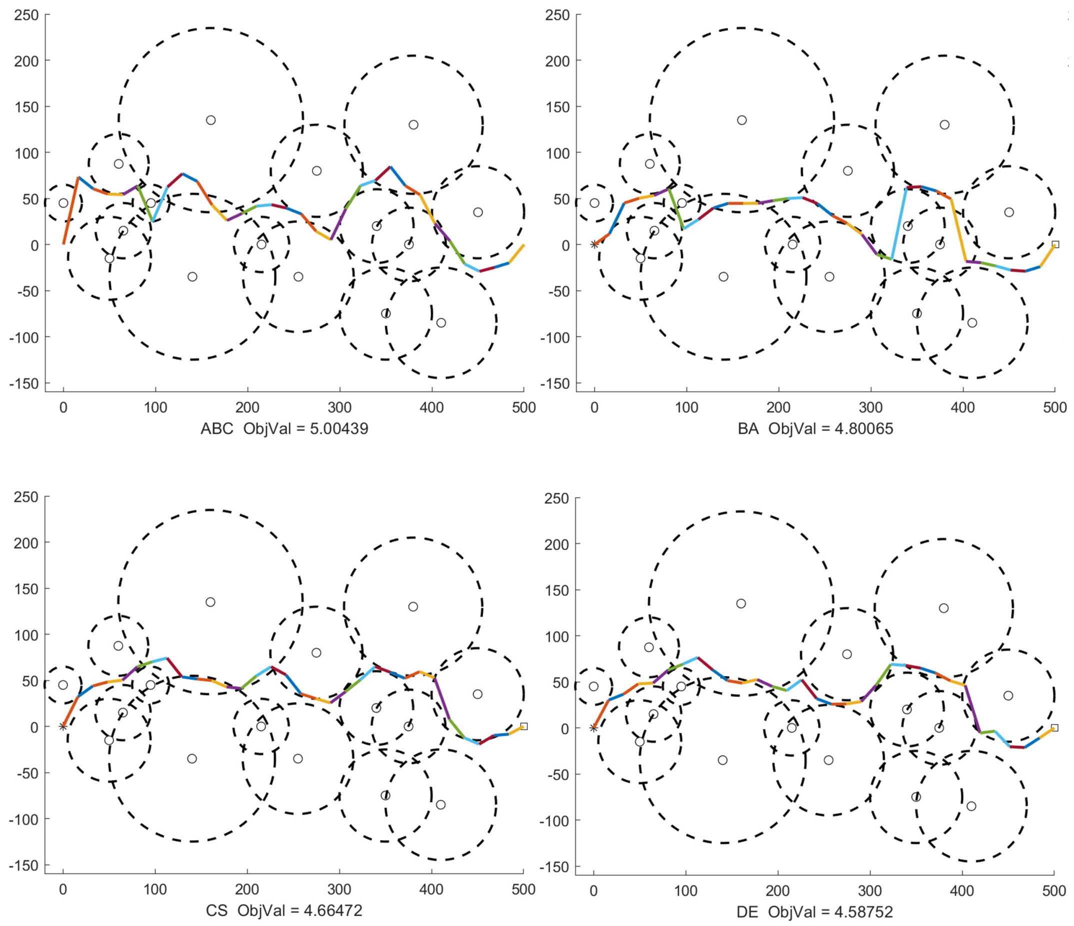

Figure 6. The comparison performance of different swarm intelligence algorithms including ABC, BA, CS and DE for Case 8 with $D = 30, N = 60$.

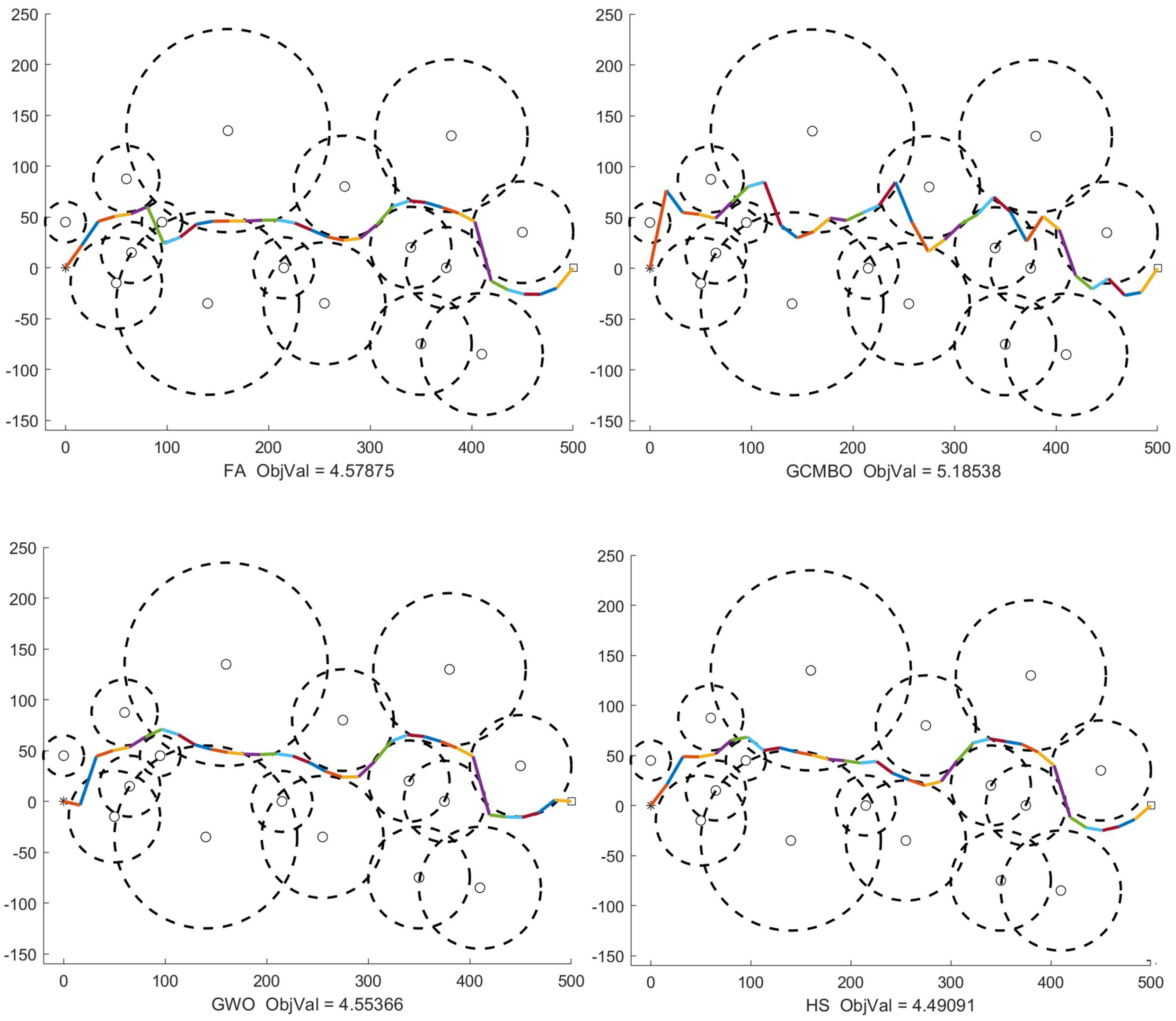

Figure 7. The comparison performance of different swarm intelligence algorithms including FA, GCMBO, GWO and HS for Case 8 with $D = 30, N = 60$.

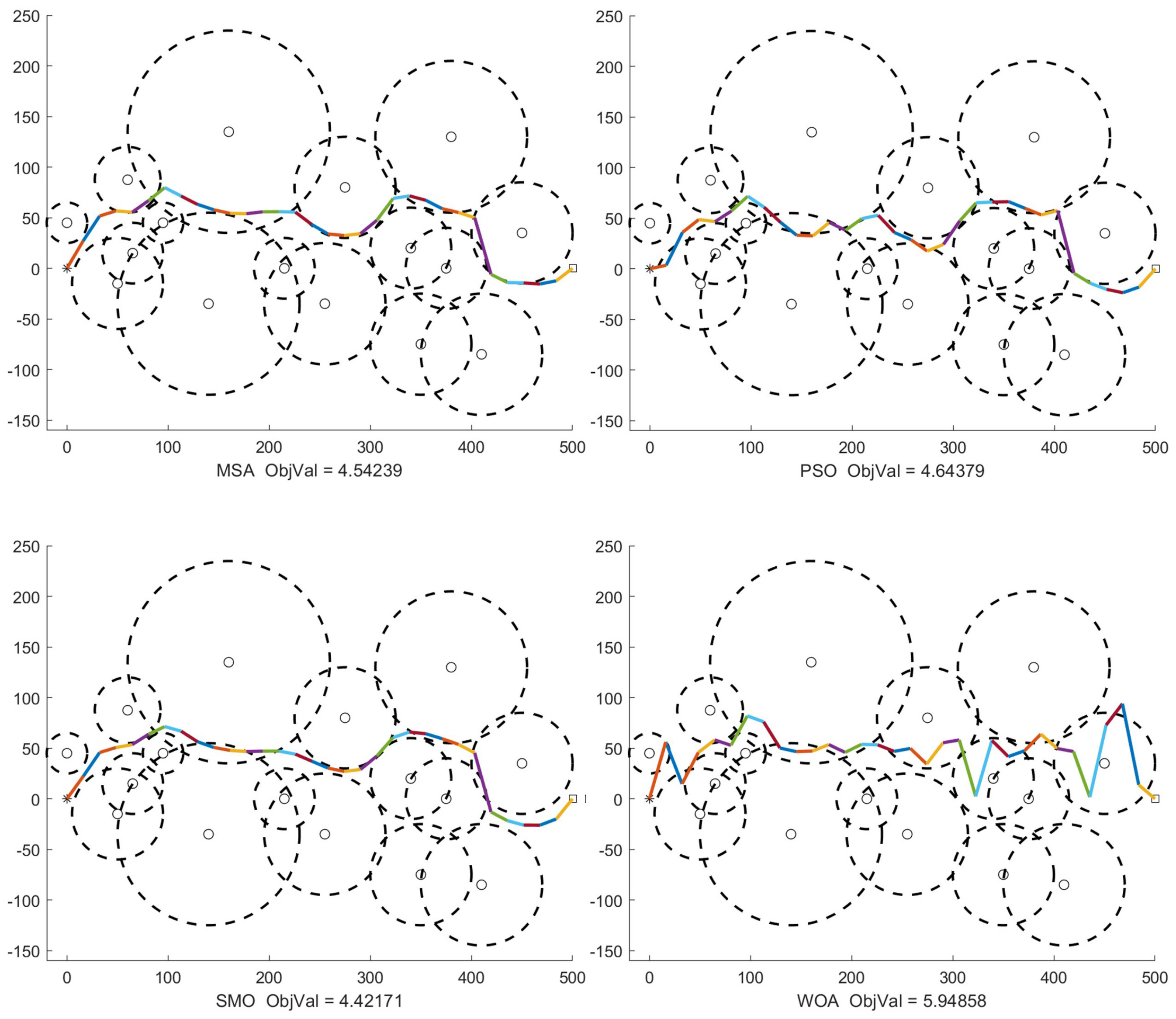

Figure 8. The comparison performance of different swarm intelligence algorithms including MSA, PSO, SMO and WOA for Case 8 with $D = 30, N = 60$.

Table 3. Performance comparison of different swarm intelligence algorithms for medium-scale UCAV path-planning. The best, worst, average and standard deviation over 25 runs of each algorithm for ten cases are provided. The best values are in bold.

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Case 7**	**Case 8**	**Case 9**	**Case 10**	
ABC	5.65076	1.81274	2.21308	2.61850	3.07183	2.99558	4.69979	5.00439	4.15122	3.92333	Best
	6.08632	2.21838	2.66935	3.34486	3.81704	3.50178	5.23968	5.64008	5.21948	4.66747	Worst
	5.91296	2.06146	2.43436	3.04549	3.42027	3.20440	4.96524	5.37282	4.74236	4.39441	Average
	0.11717	0.09269	0.11428	0.15528	0.19154	0.15197	0.14500	0.16854	0.30034	0.18788	Std deviation
BA	5.48759	1.77326	1.90156	2.46398	3.04086	2.98142	4.29104	4.80065	4.46421	4.01242	Best
	6.34582	2.15701	2.66515	3.86547	4.76201	3.93330	5.56908	6.15390	5.82402	5.00900	Worst
	5.88167	1.92546	2.32488	3.01165	3.64501	3.27451	5.07498	5.46074	5.16717	4.43730	Average
	0.21399	0.09885	0.22482	0.36855	0.40921	0.22400	0.34054	0.34729	0.35195	0.27593	Std deviation
CS	5.65665	1.84557	2.06679	2.52672	2.83356	2.76835	4.26268	4.66472	4.09219	3.78004	Best
	5.92436	2.08908	2.20753	2.74825	3.07316	3.16608	4.53927	4.98162	4.42214	4.13001	Worst
	5.77357	1.96735	2.14125	2.62565	2.96140	2.95712	4.41239	4.80506	4.23083	3.99211	Average
	0.08256	0.05361	0.04085	0.06282	0.07160	0.10896	0.08635	0.08756	0.08683	0.10191	Std deviation
DE	5.59849	1.85658	1.98665	2.47533	2.75406	2.92717	4.14844	4.58752	3.98125	3.69402	Best
	5.80682	1.99903	2.11889	2.65453	2.95975	3.25458	4.31031	4.76395	4.22699	3.92693	Worst
	5.72977	1.92201	2.05332	2.53862	2.85785	3.06828	4.22782	4.68243	4.08003	3.81012	Average
	0.05331	0.03472	0.02914	0.05005	0.05363	0.08813	0.04530	0.05005	0.05506	0.06165	Std deviation
FA	5.33925	1.69999	**1.88776**	2.40394	2.55200	2.48963	4.68407	4.57875	3.84686	3.51198	Best
	5.96082	1.97332	2.37488	3.09543	2.68846	3.02501	5.22605	5.16279	4.65075	4.40823	Worst
	5.69696	1.81607	2.09590	2.67013	2.56051	2.73860	4.98621	4.78427	4.42518	3.76076	Average
	0.16364	0.06355	0.19622	0.23080	0.02813	0.16406	0.11673	0.15727	0.18589	0.18211	Std deviation
GCMBO	5.78657	2.01830	2.39372	2.90994	3.14760	3.12060	4.54533	5.18538	4.54210	4.21660	Best
	6.30673	2.48651	3.22922	3.62314	4.24778	4.16561	5.70627	6.44745	5.75663	5.02060	Worst
	6.06380	2.21077	2.66618	3.19552	3.67160	3.49270	5.02587	5.63362	5.02665	4.63974	Average
	0.13623	0.12100	0.18939	0.19989	0.28268	0.23986	0.26947	0.30288	0.35624	0.22564	Std deviation
GWO	5.34573	1.73384	1.91593	2.31796	2.52626	2.34983	4.36972	4.55366	3.78445	3.52735	Best
	5.73215	2.09779	2.31262	3.11416	2.94557	3.13474	5.11318	5.39305	5.13422	4.56521	Worst
	5.56816	1.89774	2.05335	2.59894	2.70787	2.69087	4.69397	4.92003	4.18506	3.92667	Average
	0.09551	0.08944	0.14233	0.23897	0.11710	0.21214	0.21956	0.22447	0.34129	0.25907	Std deviation

Table 3. *Cont.*

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Case 7**	**Case 8**	**Case 9**	**Case 10**	
HS	5.35354	1.72596	1.94628	2.30948	2.56904	2.38619	4.09788	4.49091	3.78557	3.57972	Best
	5.54654	1.95107	2.03262	2.42990	2.73681	2.56015	4.20235	4.63989	3.96182	3.93696	Worst
	5.45640	1.80369	1.98097	2.35934	2.67809	2.48170	4.12535	4.53653	3.86109	3.74365	Average
	0.06811	0.06155	0.02289	0.02769	0.04163	**0.03772**	0.02180	0.04085	0.04647	0.10070	Std deviation
MSA	5.60787	1.83507	1.93295	2.36184	2.47755	2.45798	4.22888	4.54239	3.78452	3.86997	Best
	6.19402	2.12393	2.55960	3.59325	3.04907	3.55619	5.45793	5.54283	5.21095	4.43788	Worst
	5.83312	1.97181	2.16728	2.74959	2.68769	2.87685	4.86628	5.01900	4.38520	4.17755	Average
	0.14906	0.08224	0.19211	0.30554	0.11633	0.27158	0.31305	0.27074	0.43944	0.15929	Std deviation
PSO	5.98899	1.88923	2.07140	2.84528	2.70623	2.75534	4.63766	4.64379	4.04411	3.92172	Best
	6.47456	2.23412	2.49609	3.81686	3.13978	3.33932	5.53994	5.27142	4.95748	4.57736	Worst
	6.22912	2.00588	2.24133	3.19410	2.87739	3.08111	5.07385	4.93601	4.40160	4.22134	Average
	0.14031	0.07914	0.12277	0.22559	0.11863	0.14866	0.23081	0.16315	0.20275	0.15953	Std deviation
WOA	6.61783	2.39543	2.77779	4.11051	3.97166	3.43633	5.56872	5.94858	5.55131	5.47799	Best
	7.61292	3.10127	3.69120	4.91711	5.72925	4.62582	7.04486	7.96494	7.38154	6.84366	Worst
	7.17790	2.74974	3.23990	4.42680	4.67329	4.20823	6.41485	6.93741	6.33971	5.97577	Average
	0.26680	0.21759	0.26836	0.23448	0.41286	0.26723	0.32993	0.52289	0.42879	0.35668	Std deviation
SMO	**5.29633**	**1.67871**	**1.88776**	**2.24303**	**2.45917**	**2.29663**	**4.04452**	**4.42171**	**3.60465**	**3.46036**	**Best**
	5.31442	**1.70088**	**1.88782**	**2.24332**	**2.55200**	**2.39585**	**4.04452**	**4.42171**	**3.69517**	**3.47025**	**Worst**
	5.29896	**1.68123**	**1.88779**	**2.24308**	**2.46334**	**2.32247**	**4.04452**	**4.42171**	**3.66045**	**3.46117**	**Average**
	0.00504	**0.00606**	**0.00001**	**0.00006**	**0.01848**	**0.03772**	**0.00000**	**0.00000**	**0.03406**	**0.00207**	**Std deviation**

4.5. Performance Comparison on Large-Scale Problems

In this section, the performance of each algorithm on ten large-scale cases (D = 60) are investigated. Figures 9–11 show the paths performed by different algorithms on Case 8 in large-scale. We can observe that as the dimension increases, all the algorithms perform much worse except SMO. As depicted in Figures 9–11, we can notice that three algorithms including BA, FA, and SMO can provide the smooth paths. Compared with the paths in Figures 3–5 and Figures 6–8, it can be seen that the performance decreases significantly with the increasing dimension. Most algorithms cannot generate the successful paths since the search ability decreases with the increasing dimension. The best, worst, average objective values and the standard deviation of each algorithm on each case are tabulated in Table 4. From Table 4, we can conclude the following observations. (a) SMO provides the best path among all those twelve algorithms while the path provided by WOA has the worst performance. Moreover, the path provided by SMO has scarcely changed when compared with the cases in small-scale and medium-scale. (b) Although FA can present the acceptable paths, FA made different steps from SMO. That's the reason why the objective value of FA is larger than that of SMO. (c) Based on the std deviation, it reveals that SMO can provide the most stable paths over multiple independent runs among all the algorithms. Moreover, the convergence rate of Case 8 under the large-scale problems is presented in Figure 12. We can observe that most algorithms can obtain the best results within less than 150 fitness evaluations. SMO can perform the best paths. However, ABC, DE and WOA cannot reach the convergence point due to the scant number of fitness evaluations of the large scale. It can be concluded that SMO has superiority over other compared algorithms with good robustness.

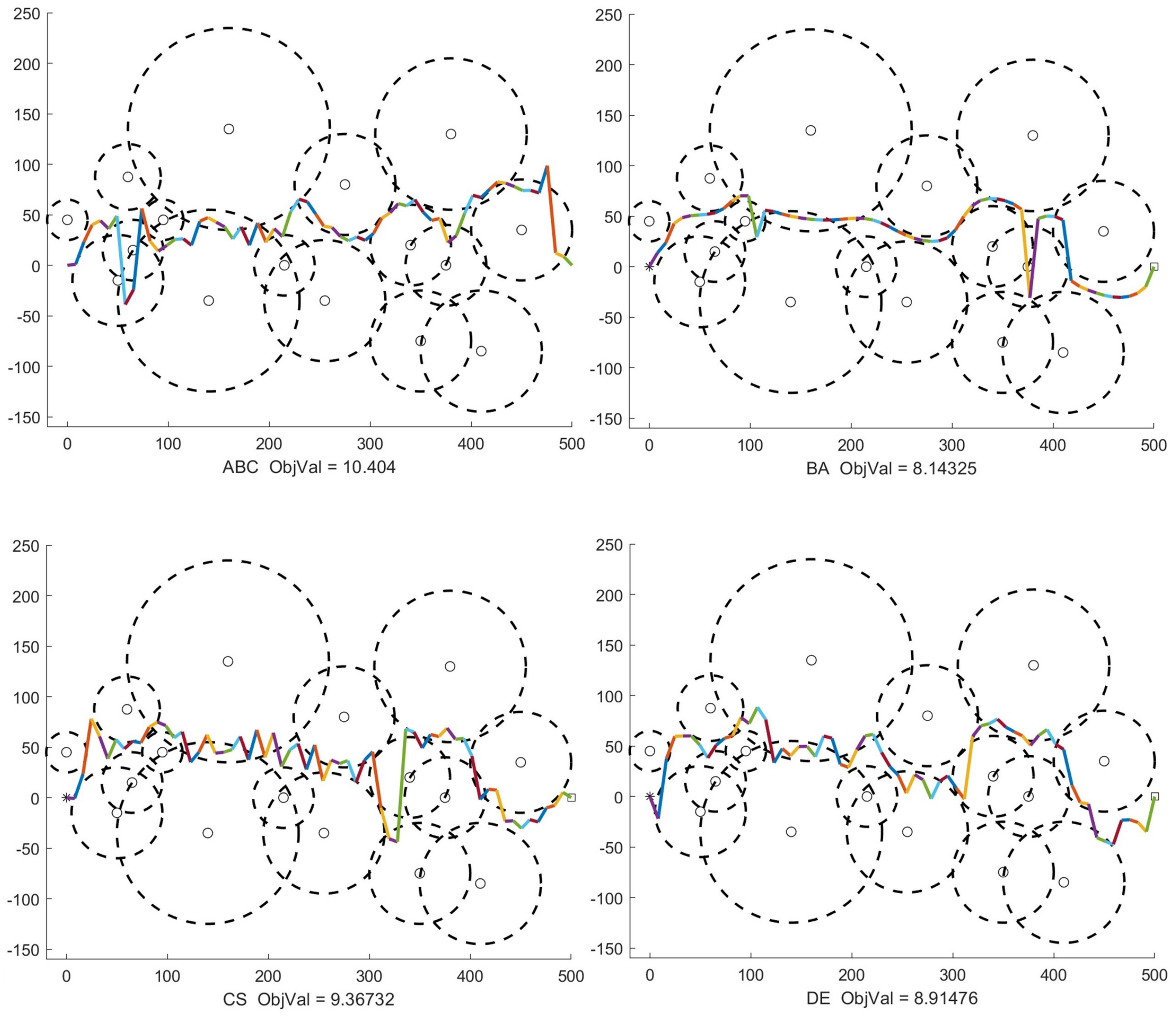

Figure 9. The comparison performance of different swarm intelligence algorithms including ABC, BA, CS and DE for Case 8 with $D = 60, N = 120$.

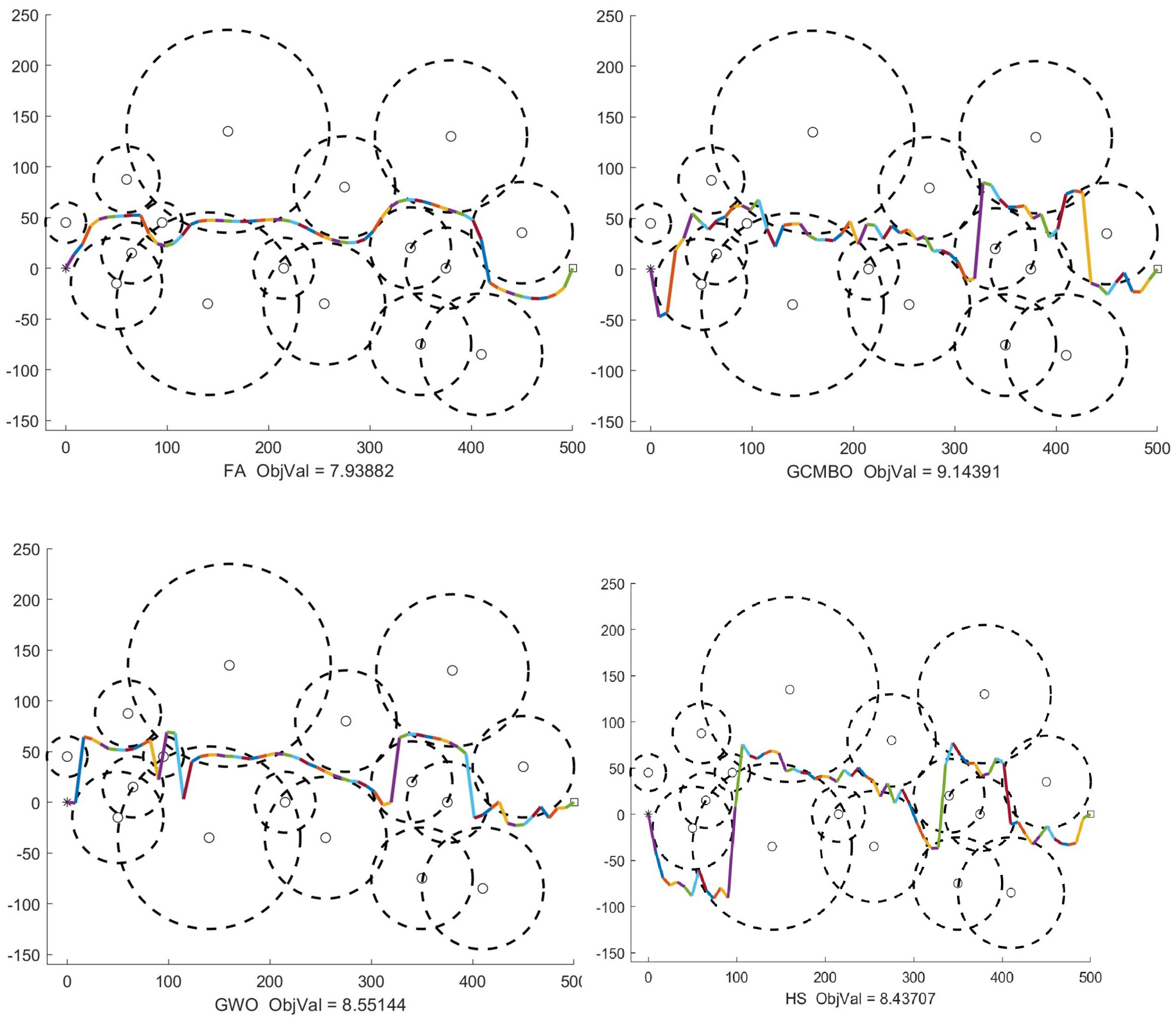

Figure 10. The comparison performance of different swarm intelligence algorithms FA, GCMBO, GWO and HS for Case 8 with $D = 60, N = 120$.

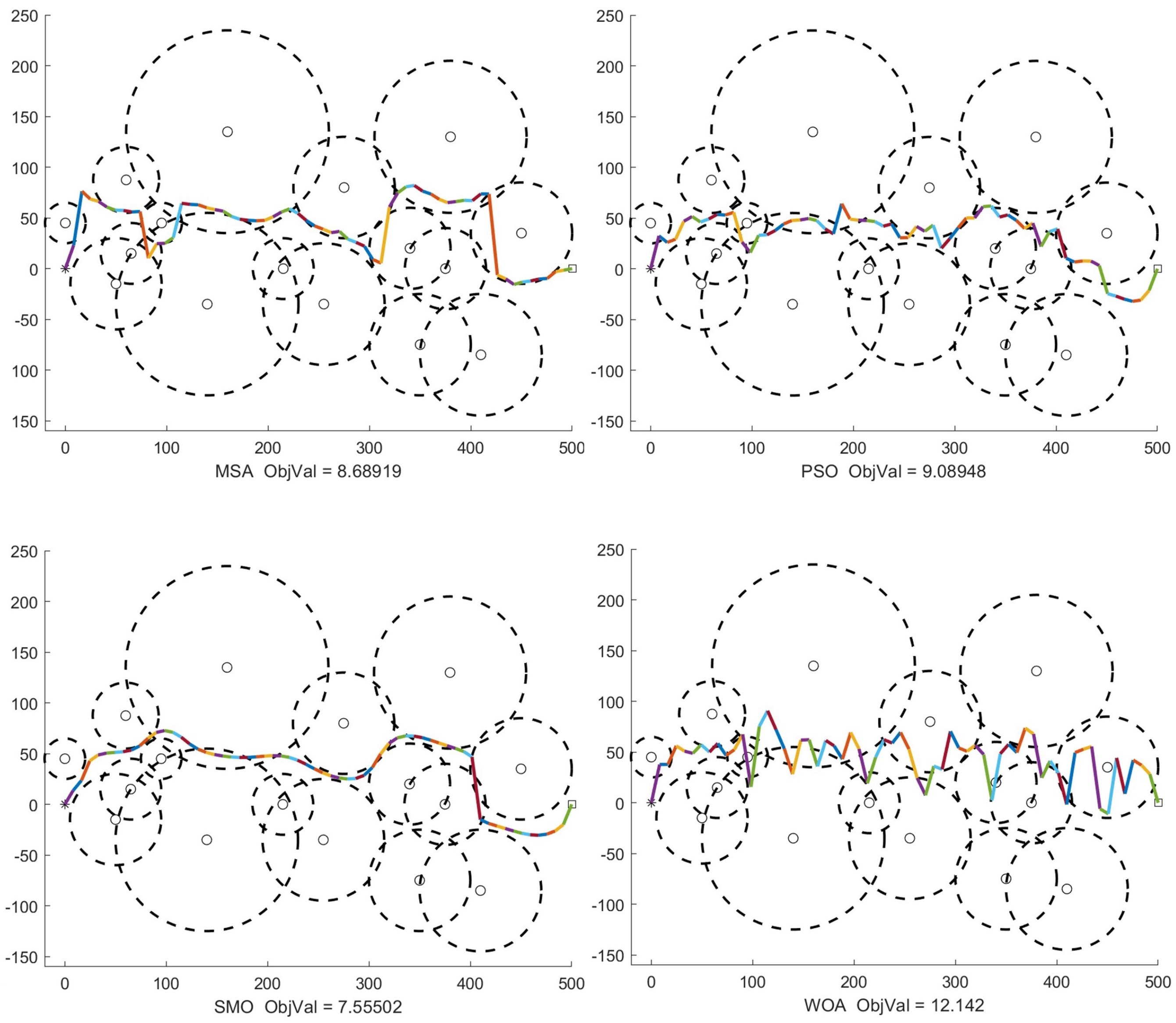

Figure 11. The comparison performance of different swarm intelligence algorithms MSA, PSO, SMO and WOA for Case 8 with $D = 60, N = 120$.

Table 4. Performance comparison of different swarm intelligence algorithms for large-scale UCAV path-planning. The best, worst, average and standard deviation over 25 runs of each algorithm for ten cases are provided. The best values are in bold.

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**	**Case 6**	**Case 7**	**Case 8**	**Case 9**	**Case 10**	
ABC	11.28948	3.52370	4.62416	5.27874	6.57170	6.29494	9.21089	10.40403	8.69186	8.05864	Best
	12.45483	4.23379	5.74213	7.18993	8.43326	7.31052	11.21872	11.79650	11.61452	10.06662	Worst
	11.82839	3.88755	5.20526	6.61427	7.70273	6.85397	10.35821	11.20326	10.48508	9.33420	Average
	0.26937	0.19687	0.28801	0.40836	0.53493	0.30889	0.51347	0.37926	0.64202	0.43498	Std deviation
BA	10.07523	2.36412	3.03078	3.56815	4.98585	4.56774	7.93291	8.14325	7.42097	6.66584	Best
	11.34526	3.25754	4.03706	5.66683	7.15262	6.17443	9.93842	10.89560	10.30115	9.39240	Worst
	10.84208	2.86006	3.49477	4.62838	5.88189	5.38978	9.25137	9.66495	8.97254	7.80731	Average
	0.30431	0.21337	0.28616	0.56681	0.49289	0.40968	0.47881	0.63872	0.71529	0.62308	Std deviation
CS	11.02470	3.27712	4.05966	5.11979	6.08151	5.68708	8.80291	9.36732	8.60296	7.74665	Best
	11.97790	3.75498	4.62767	5.84504	6.84316	6.76414	9.59481	10.38349	9.52930	8.67339	Worst
	11.56665	3.48136	4.32548	5.50343	6.41659	6.35895	9.22561	9.85457	9.02512	8.33623	Average
	0.23127	0.11770	0.15375	0.19957	0.20786	0.25384	0.21309	0.26566	0.25167	0.24913	Std deviation
DE	10.81923	3.05663	3.65372	4.48336	5.54172	6.41216	8.09821	8.91476	7.88400	7.28692	Best
	11.41790	3.39874	4.03394	5.17994	6.05136	6.83525	8.57801	9.62494	8.58606	8.06260	Worst
	11.17581	3.26770	3.84818	4.83894	5.80331	6.62057	8.35366	9.24848	8.23071	7.66489	Average
	0.12184	0.07760	0.09269	0.13693	0.12447	0.13182	0.12208	0.19508	0.15882	0.18647	Std deviation
FA	9.74002	2.35975	2.68280	4.03736	3.92864	3.73500	8.34560	7.93882	7.17874	6.13161	Best
	10.93901	3.00926	3.65493	5.25249	4.06381	4.81286	9.54978	8.82559	8.26742	6.50968	Worst
	10.27844	2.55646	3.31374	4.56729	3.94498	4.39317	8.81935	8.34753	7.64827	6.20375	Average
	0.28866	0.16495	0.25383	0.28831	0.03587	0.27751	0.24707	0.26402	0.25245	0.08006	Std deviation
GCMBO	10.75231	3.06132	3.86643	5.03095	6.12203	5.80826	8.74677	9.14391	8.46155	7.92789	Best
	11.98840	3.82188	5.00522	6.57505	7.82037	6.81993	10.11920	11.09870	10.22534	9.64340	Worst
	11.22760	3.47342	4.55553	5.67516	6.86026	6.34602	9.21887	10.39141	9.27781	8.66641	Average
	0.26693	0.19764	0.29868	0.42072	0.43076	0.28311	0.33034	0.46456	0.55259	0.40073	Std deviation
GWO	9.78014	2.47045	2.89594	3.53668	3.97649	3.78132	8.09615	8.55144	7.10976	6.40047	Best
	10.95061	2.98815	4.12009	5.12438	5.22502	5.29693	9.59289	10.66260	8.88337	7.86799	Worst
	10.29231	2.77026	3.45772	4.15567	4.45868	4.44264	8.78866	9.48210	8.00236	7.11720	Average
	0.26782	0.13709	0.33983	0.40610	0.27546	0.35927	0.35898	0.55005	0.55676	0.36556	Std deviation

Table 4. *Cont.*

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	
HS	9.82004	2.64062	3.32102	4.00540	4.88691	4.03190	7.48906	8.37265	6.83698	6.75239	Best
	10.34328	2.94563	3.69148	4.17452	5.32263	4.61613	7.83730	8.71449	7.54559	7.42808	Worst
	10.13713	2.79976	3.47388	4.08885	5.09346	4.36115	7.65542	8.54144	7.17944	7.06739	Average
	0.13847	0.08645	0.08576	0.05438	0.12405	0.12310	0.10100	0.07568	0.16236	0.15036	Std deviation
MSA	10.32344	2.87248	2.96156	4.22400	4.16115	4.57433	8.64102	8.68919	6.70786	7.00489	Best
	11.86800	3.90167	4.12512	5.86799	5.36838	5.89082	10.12286	10.38032	9.10212	8.67775	Worst
	10.94214	3.18777	3.66284	4.91878	4.67980	5.17434	9.40588	9.43296	7.89441	7.82829	Average
	0.38142	0.25169	0.29124	0.53616	0.33062	0.33019	0.40917	0.45564	0.62966	0.52474	Std deviation
PSO	11.50878	3.03016	3.37450	5.62018	4.83580	5.43883	9.65874	9.08948	7.99124	7.12936	Best
	12.60180	3.99445	4.32213	6.85294	6.22815	6.66681	11.35977	10.23278	9.50469	8.94442	Worst
	12.04514	3.51718	3.87036	6.14793	5.39061	5.97327	10.45275	9.68565	8.88687	8.07835	Average
	0.31066	0.23999	0.26719	0.30718	0.30680	0.29597	0.45541	0.31336	0.39625	0.41303	Std deviation
WOA	13.07540	4.35528	5.29230	7.27906	8.05213	6.75522	11.90830	12.14199	11.27253	11.16476	Best
	14.99135	5.95417	7.10852	9.70345	10.45468	9.73888	14.05014	15.23174	14.75277	13.14390	Worst
	14.15923	5.32378	6.13517	8.50375	9.09167	8.24215	12.90899	14.05722	12.75595	11.96014	Average
	0.45608	0.34832	0.44914	0.59170	0.71228	0.75644	0.62665	0.73750	0.76187	0.46090	Std deviation
SMO	**9.18336**	**2.23828**	**2.68036**	**3.16685**	**3.70875**	**3.36677**	**6.88067**	**7.55502**	**5.98846**	**5.54869**	Best
	9.23077	**2.25006**	**2.68237**	**3.16958**	**3.93383**	**3.64994**	**6.88070**	**7.55514**	**5.99026**	**5.59005**	Worst
	9.19360	**2.24292**	**2.68106**	**3.16777**	**3.72058**	**3.43767**	**6.88068**	**7.55506**	**5.98890**	**5.55364**	Average
	0.01229	**0.00351**	**0.00051**	**0.00070**	**0.04451**	**0.10010**	**0.00001**	**0.00003**	**0.00037**	**0.00895**	Std deviation

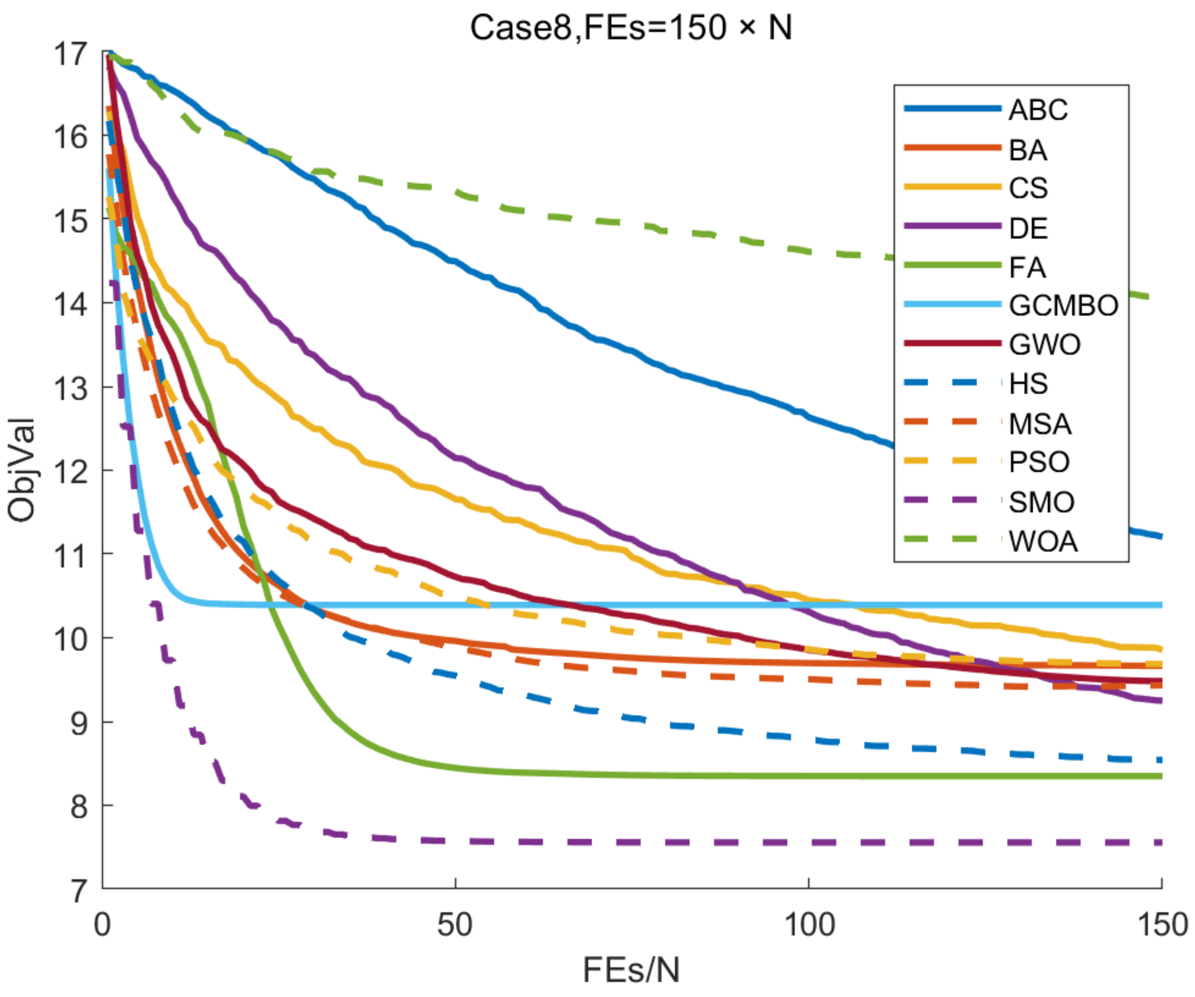

Figure 12. The convergence rate of different SI algorithm for Case 8 with large-scale. The horizontal axis denotes the number of fitness evaluations divided by the size of the population N, which is the iteration times. The vertical axle is the average objective value over 25 runs.

4.6. Stability of the Number of Fitness Evaluations

Generally, the number of iterations is often used for the criteria of termination. However, the number of fitness evaluations depends on the unique feature of each algorithm. The different procedure of each algorithm causes the different number of fitness evaluations in an iteration. Therefore, it is common to use the number of fitness evaluations as the termination threshold instead of the number of iterations [22]. Selecting a proper number of fitness evaluations is a challenging task. If the number of fitness evaluations is not enough, the optimal results can not be obtained. Otherwise, it would be wasted if the number of fitness evaluations is too big because the best result has been found very early.

Tables 2–4 show that SMO obtains the best paths among all the algorithms for UCAV path-planning problem. To discuss the stability of FEs for SMO, a convergence analysis based on the number of fitness evaluations is conducted on Case 8 under the large-scale problems. As depicted in Figure 13, we can observe that the best result has been obtained at $50 \times N$ times, which is far less than $150 \times N$. This indicates that SMO has great optimizing capabilities for the UCAV path-planning problems.

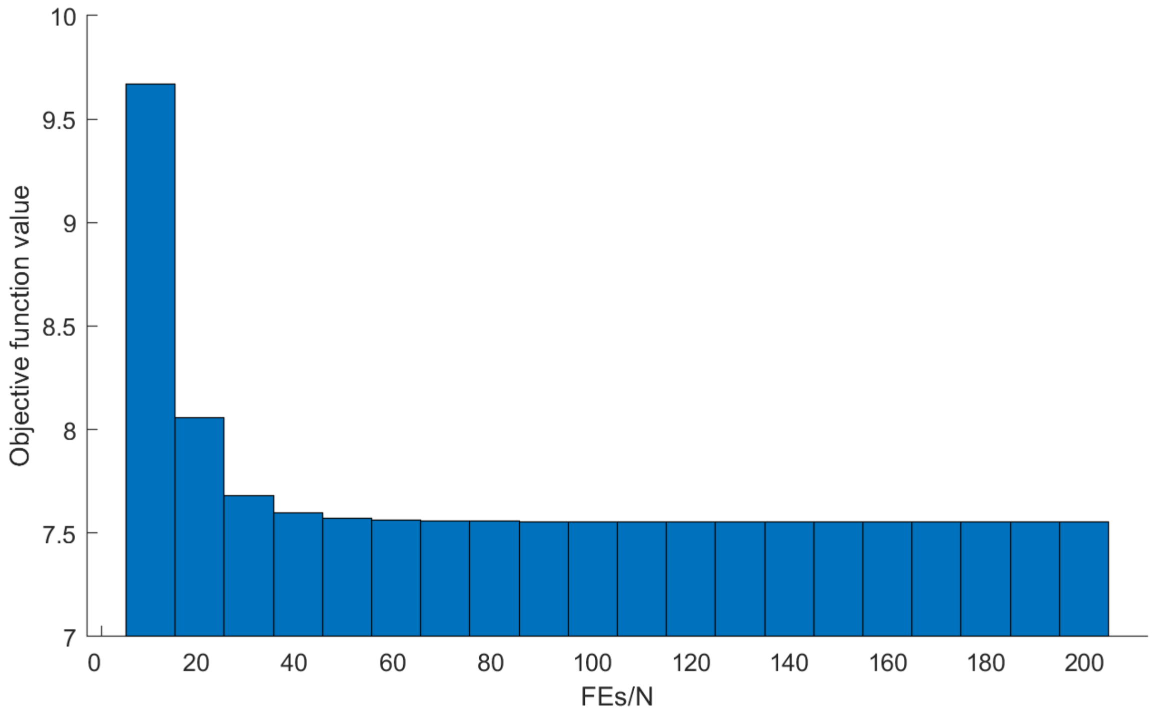

Figure 13. Convergence behavior of SMO. The horizontal axis denotes the number of fitness evaluations divided by *N*, which is the iteration times, and the vertical axis denotes the average objective value.

5. Conclusions

This study provides a comparison of twelve SI algorithms for the UCAV path-planning problem. We survey twelve algorithms from relevant studies and apply them to UCAV path-planning. In the experiment, thirty UCAV path-planning cases are employed to verify the performance of each algorithm. The experimental results demonstrate that SMO is superior to other algorithms from different perspectives. The main contributions of this study are as follows:

- Twelve SI algorithms including ABC, BA, CS, DE, FA, GCMBO, GWO, HS, MSA, PSO, SMO, and WOA, are applied to UCAV path-planning problem by reviewing a number of papers.
- Thirty cases in different scales including ten small-scale cases with $D = 20$, ten medium-scale cases with $D = 30$, and ten large-scale cases with $D = 60$ are designed to demonstrate the effectiveness of each algorithm.
- A comprehensive analysis of each algorithm to address the UCAV path-planning problem is conducted from the perspective of path quality, stability, and convergence. The experimental results demonstrate that SMO exhibits better performance in discovering the safe path than the other SI algorithms.

For the UCAV path-planning problems in different scales, SMO performs the best in average while WOA delivers an obvious inferiority in obtaining good average results. In particular, SMO can provide the best average results over 25 independent runs on Case 2 in small-scale, medium-scale, and large-scale. The std deviation of SMO on each case is equal or very close to zero, which demonstrates the strong robustness of SMO. However, SMO still might get trapped into local optimum in some cases. There are two main reasons for this phenomenon: only the terminal point of each step is considered in the path-planning model, resulting in a phenomenon that a path that goes through the center of the threat has a lower objective value than that detours the threat. Another reason is that it is not necessary to generate a high-dimensional solution depending on the reality.

In the future, SMO-based algorithms should be developed based on the analysis of SMO by exploring other special strategies to enhance the efficiency. They can be applied to other route planning problems. Moreover, we will focus on designing effective methods to construct UCAV path-planning models in three-dimension.

Author Contributions: Methodology, H.Z.; supervision, X.L.; validation, Y.W.; visualization, Y.W.; writing—original draft, H.Z.; writing—review & editing, Z.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Natural Science Foundation of China under Grant No. 62076109, and funded by the Natural Science Foundation of Jilin Province under Grant No. 20190103006JH. Health and Medical Research Fund, the Food and Health Bureau, the Government of the Hong Kong Special Administrative Region [07181426], and the funding from Hong Kong Institute for Data Science (HKIDS) at City University of Hong Kong. The work described in this paper was partially supported by two grants from City University of Hong Kong (CityU 11202219, CityU 11203520), National Natural Science Foundation of China under Grant No. 32000464.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

References

1. Kabamba, P.T.; Meerkov, S.M.; Zeitz, F.H., III. Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking. *J. Guid. Control Dyn.* **2006**, *29*, 279–288. [CrossRef]
2. Sud, A.; Andersen, E.; Curtis, S.; Lin, M.; Manocha, D. Real-time path planning for virtual agents in dynamic environments. In Proceedings of the 2007 IEEE Virtual Reality Conference, Charlotte, NC, USA, 10–14 March 2007; pp. 91–98.
3. Xin, H.; Chen, Q.; Wang, Y.; Jia, G.; Hou, Z. An Optimal Path Planning Method for UCAV in Terminal of Target Strike. In Proceedings of the 2019 Chinese Control and Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 3672–3676.
4. You, S.; Gao, L.; Diao, M. Real-time path planning based on the situation space of UCAVS in a dynamic environment. *Microgravity Sci. Technol.* **2018**, *30*, 899–910. [CrossRef]
5. Chen, H.X.; Nan, Y.; Yang, Y. A two-stage method for UCAV TF/TA path planning based on approximate dynamic programming. *Math. Probl. Eng.* **2018**, *2018*, 1092092. [CrossRef]
6. Wei, Z.; Huang, C.; Han, T.; Dong, K.; Li, Y. UCAVs online collaborative path planning method based on dynamic task allocation. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 872–877.
7. Pehlivanoglu, Y.V. A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. *Aerosp. Sci. Technol.* **2012**, *16*, 47–55. [CrossRef]
8. Zhang, S.; Zhou, Y.; Li, Z.; Pan, W. Grey wolf optimizer for unmanned combat aerial vehicle path planning. *Adv. Eng. Softw.* **2016**, *99*, 121–136. [CrossRef]
9. Wang, G.G.; Chu, H.E.; Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.* **2016**, *49*, 231–238. [CrossRef]
10. Yi, J.H.; Lu, M.; Zhao, X.J. Quantum inspired monarch butterfly optimisation for UCAV path planning navigation problem. *Int. J. Bio-Inspired Comput.* **2020**, *15*, 75–89. [CrossRef]
11. Pan, J.S.; Liu, N.; Chu, S.C. A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning. *IEEE Access* **2020**, *8*, 17691–17712. [CrossRef]
12. Huang, H.; Zhuo, T. Multi-model cooperative task assignment and path planning of multiple UCAV formation. *Multimed. Tools Appl.* **2019**, *78*, 415–436. [CrossRef]
13. Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Three dimensional path planning using Grey wolf optimizer for UAVs. *Appl. Intell.* **2019**, *49*, 2201–2217. [CrossRef]
14. Paszkiel, S.; Sikora, M. The Use of Brain-Computer Interface to Control Unmanned Aerial Vehicle. In Proceedings of the Conference on Automation, Warsaw, Poland, 27–29 March 2019; pp. 583–598.
15. Parpinelli, R.S.; Lopes, H.S. New inspirations in swarm intelligence: A survey. *Int. J. Bio-Inspired Comput.* **2011**, *3*, 1–16. [CrossRef]
16. Yang, X.S.; Cui, Z.; Xiao, R.; Gandomi, A.H.; Karamanoglu, M. *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*; Elsevier: Waltham, MA, USA, 2013.
17. Blum, C.; Li, X. Swarm intelligence in optimization. In *Swarm Intelligence*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 43–85.
18. Chakraborty, A.; Kar, A.K. Swarm intelligence: A review of algorithms. In *Nature-Inspired Computing and Optimization*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 475–494.
19. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
20. Bansal, J.C.; Sharma, H.; Jadon, S.S.; Clerc, M. Spider monkey optimization algorithm for numerical optimization. *Memetic Comput.* **2014**, *6*, 31–47. [CrossRef]
21. Ma, H.; Ye, S.; Simon, D.; Fei, M. Conceptual and numerical comparisons of swarm intelligence optimization algorithms. *Soft Comput.* **2017**, *21*, 3081–3100. [CrossRef]

22. Li, X.; Wong, K.C. A comparative study for identifying the chromosome-wide spatial clusters from high-throughput chromatin conformation capture data. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2017**, *15*, 774–787. [CrossRef]
23. Gunavathi, C.; Premalatha, K. A comparative analysis of swarm intelligence techniques for feature selection in cancer classification. *Sci. World J.* **2014**, *2014*, 693831. [CrossRef] [PubMed]
24. Parpinelli, R.S.; Teodoro, F.R.; Lopes, H.S. A comparison of swarm intelligence algorithms for structural engineering optimization. *Int. J. Numer. Methods Eng.* **2012**, *91*, 666–684. [CrossRef]
25. Li, B.; Gong, L.; Zhao, C. Unmanned combat aerial vehicles path planning using a novel probability density model based on Artificial Bee Colony algorithm. In Proceedings of the 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), Beijing, China, 9–11 June 2013; pp. 620–625. [CrossRef]
26. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.
27. 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; pp. 210–214.
28. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
29. Yang, X.S. Firefly algorithm, Levy flights and global optimization. In *Research and Development in Intelligent Systems XXVI*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 209–218.
30. Wang, G.G.; Zhao, X.; Deb, S. A novel monarch butterfly optimization with greedy strategy and self-adaptive. In Proceedings of the 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI), Hong Kong, China, 23–24 November 2015; pp. 45–50.
31. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
32. Omran, M.G.; Mahdavi, M. Global-best harmony search. *Appl. Math. Comput.* **2008**, *198*, 643–656. [CrossRef]
33. Wang, G.G. Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. *Memetic Comput.* **2018**, *10*, 151–164. [CrossRef]
34. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
35. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [CrossRef]