applied sciences

Article

A Task Allocation Strategy of the UAV Swarm Based on Multi-Discrete Wolf Pack Algorithm

Shufang Xu [1,2,*], Linlin Li [1,2], Ziyun Zhou [1,2], Yingchi Mao [1,2] and Jianxin Huang [3]

1 School of Computer and Information, Hohai University, Nanjing 211100, China; 201307040034@hhu.edu.cn (L.L.); 1706020317@hhu.edu.cn (Z.Z.); yingchimao@hhu.edu.cn (Y.M.)
2 Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing 211100, China
3 Suma Technology Co., Ltd., Kunshan, Suzhou 215300, China; Huangjx@cancon.com.cn
* Correspondence: xushufang@hhu.edu.cn

Abstract: With the continuous development of artificial intelligence, swarm control and other technologies, the application of Unmanned Aerial Vehicles (UAVs) in the battlefield is more and more extensive, and the UAV swarm is increasingly playing a prominent role in the future of warfare. How tasks are assigned in the dynamic and complex battlefield environment is very important. This paper proposes a task assignment model and its objective function based on dynamic information convergence. In order to resolve this multidimensional function, the Wolf Pack Algorithm (WPA) is selected as the alternative optimization algorithm. This is because its functional optimization of high-dimensional complex problems is better than other intelligent algorithms, and the fact that it is more suitable for UAV swarm task allocation scenarios. Based on the traditional WPA algorithm, this paper proposes a Multi-discrete Wolf Pack Algorithm (MDWPA) to solve the UAV task assignment problem in a complex environment through the discretization of wandering, calling, sieging behavior, and new individual supplement. Through Orthogonal Experiment Design (OED) and analysis of variance, the results show that MDWPA performs with better accuracy, robustness, and convergence rate and can effectively solve the task assignment problem of UAVs in a complex dynamic environment.

Keywords: wolf pack algorithm; task allocation; UAV swarm; discrete

check for updates

Citation: Xu, S.; Li, L.; Zhou, Z.; Mao, Y.; Huang, J. A Task Allocation Strategy of the UAV Swarm Based on Multi-Discrete Wolf Pack Algorithm. *Appl. Sci.* **2022**, *12*, 1331. https://doi.org/10.3390/app12031331

Academic Editor: Giancarlo Mauri

Received: 21 December 2021
Accepted: 24 January 2022
Published: 26 January 2022

1. Introduction

Due to its high flexibility and wide adaptability, the UAV swarm has more and more extensive application potential and has received great attention at home and abroad [1]. With the continuous development of technologies such as artificial intelligence, swarm control, and collaborative interaction, the application of Unmanned Aerial Vehicles (UAVs) in the battlefield is more and more extensive, and the UAV swarm is increasingly playing a prominent role in the future of warfare [2]. Among them, the purpose of UAV task assignment is to find a distribution plan that optimizes the overall target effect, which is of great significance in the application scenario of UAV swarm combat. In general, it is a sophisticated combinatorial optimization problem with complex constraints in terms of task precedence and coordination, timing constraints, and flyable trajectories. In the real environment, it is difficult to get the optimal solution to this problem, especially for the uncertain complex dynamic environment [3]. In essence, the task allocation problem of the UAV swarm is also a large-scale, high-dimensional, and complex function optimization problem. So it is important to work out how to optimize this complex combinatorial optimization problem. At present, relevant scholars have done a lot of research on the task allocation problem of the UAV swarm.

UAV task assignment methods are mainly divided into centralized task assignment and distributed task assignment [1]. Among them, the distributed task assignment algorithm has the characteristics of good fault tolerance and strong scalability. However, due to the limited grasp of the overall information of the battlefield by each decision maker, it is often difficult to guarantee the quality of the solution. Given that, in the actual combat environment, more precise assignment plans are often required to achieve the best results from battlefield operations. Therefore, the distributed task allocation plan is not very applicable in the process of optimizing the best combat plan. Corresponding to the distributed task assignment method, is the centralized task assignment method. The control center has the global information and makes the optimal allocation plan on behalf of the overall interests and then notifies each UAV. The centralized task assignment method has the potential to generate the global optimal solution [4], which mainly includes optimization methods [5–8] and heuristic methods. The optimization algorithms can flexibly adjust the constraint conditions to solve the actual problem and have the theoretical optimal solution; however, the scale should not be too large, and it is generally used for the discrete task allocation of the UAV swarm.

The heuristic methods are independent of the mathematical performance of the problem, they do not have strict requirements on the initial value and can effectively deal with high-dimensional complex optimization problems, including dynamic lists [9], clustering algorithms [10], and smart algorithms. Among them, the characteristics of biological groups in decentralization, neighboring individual information interactions, and overall self-organization are consistent with the pursuit of the UAV swarm. Therefore, the application of intelligent algorithms in UAV swarm task assignment is relatively common. It mainly includes intelligent algorithms such as the particle swarm algorithm (PSO) [11,12], genetic algorithm (GA) [13,14], wolf pack algorithm (WPA) [15], and so on. PSO is easy to implement and has fewer control parameters. However, its problems are low search efficiency and fast premature birth [16], and as the dimensionality of the problem increases, the convergence speed of the algorithm will inevitably be affected. In recent years, there has been a lot of research on evolutionary algorithms to improve the search ability of the entire search stage. Although it has a good search performance, it cannot avoid sticking into local optima, thereby obtaining the inferior solutions [14]. Most of these algorithms have global optimization capabilities, which are very helpful for the small-scale optimization of complex functions. However, in the UAV swarm combat scenario with high real-time dynamics and complexity, the above algorithms have certain shortcomings for the optimization of high-dimensional complex functions.

Among them, the wolf pack algorithm is a new bionic intelligent algorithm proposed by Wu et al. [17] in 2013. With clever division of labor, wolves can complete a series of group activities such as complex cooperative hunting, feeding cubs, and territorial maintenance [18], showing high efficiency, flexibility, and dynamic adaptation. Compared with other algorithms, the wolf pack algorithm is a random probability search algorithm, which can quickly find the optimal solution with a high probability. WPA also has parallelism, which can search from multiple points at the same time without affecting each other, thus improving the efficiency of the algorithm. WPA has better global convergence and computational robustness and is especially suitable for solving complex functions with high dimensions and multiple peaks [19]. Therefore, the processing and adaptability of WPA has greater advantages in handling the task allocation of the UAV swarm in a dynamic and complex battlefield environment. However, the wolf pack algorithm has disadvantages such as slow convergence speed and the fact that it can easy to fall into local optimum. Many scholars have done a lot of research on how to optimize the wolf pack algorithm. The optimization of the wolf pack algorithm can be divided into three categories.

(1) Enhance the global search ability;

Literature [20] aims to improve the optimization quality, stability and convergence speed of the wolf pack algorithm. All wolves in the pack are allowed to compete as the leading wolf to improve the probability of finding the global optimal solution, thus enhancing the search ability; In order to promote information exchange between artificial wolves, improve wolves' grasp of global information and enhance their exploration ability, reference [21] introduces drift operators and wave operators into wolves' reconnaissance behavior and summation behavior. An adaptive dynamic adjustment factor strategy is proposed for besieging behavior. Literature [22] discusses an optimization strategy of WPA based on opposition learning, which not only enhances the local search ability of WPA under normal conditions, but also increases the population diversity under catastrophic conditions. In order to study the parameter optimization problem of complex nonlinear functions, an adaptive variable step size chaotic wolf optimization algorithm (CWOA) is proposed in literature [23]. Combining the adaptive variable step search strategy with the chaos optimization strategy, gives the algorithm high robustness and global search ability. In order to accelerate the convergence of WPA algorithm and enhance its search ability, the differential evolution (DE) elite-set strategy is adopted in literature [24]. In literature [25], on the premise of ensuring the solution accuracy, optimization time is reduced. A jump raid strategy is proposed, which directly airlifts half of the wolves to the lead wolf, which improves the performance of the raid process and speeds up the convergence speed of the algorithm.

(2) The balance between local search ability and global search ability;

In order to solve the disadvantages of WPA, the strategy of opposition learning is adopted to initialize the population in literature [26], so as to enhance the population diversity in the process of global search. At the same time, genetic algorithm is used to select the head wolf to avoid local optimization. This combined optimization strategy harmonizes the global exploration capability and local development capability well. Literature [27] proposes a quantum-excited wolf pack algorithm based on quantum coding: quantum rotation and quantum collapse. The first step makes the population move to the global optimal value, the second step helps to avoid individuals falling into the local optimal value. The global search capability and local search capability can be balanced.

(3) Optimization of high-dimensional complex problems;

In order to study the parameter optimization problem of complex nonlinear functions, an adaptive variable step size Chaotic Wolf Optimization Algorithm (CWOA) is proposed in literature [23]. Combining the adaptive variable step search strategy with chaos optimization strategy, the algorithm has high robustness and global search ability. For the optimization of high-dimensional functions, a new wolf pack algorithm is proposed in literature [28], which can accurately calculate the optimal value of high-dimensional functions. Firstly, the chaotic inverse initialization is designed to improve the quality of the initial solution. Secondly, interference factors are added in the search process to enhance the search ability of wolves, and the adaptive step size is designed to enhance the global search ability of wolves, which effectively prevents wolves from falling into local optimal. Literature [29] proposes a Multi-population Parallel Wolf Algorithm (MPPWPA) in order to solve the problem of too long convergence time of the wolf algorithm in the optimization of high-dimensional complex problems. By dividing the wolf population into multiple sub-populations and independently optimizing the sub-populations at the same time, the size of the wolf population is reduced. Literature [30] proposes a combination of the principle of particle swarm optimization and genetic algorithm to optimize the wolf pack algorithm and solve the task allocation problem of the UAV swarm with a faster convergence speed. The improved algorithm shows good search quality in high-dimensional space. Literature [31] applies the improved Wolf Pack Search (WPS) algorithm to calculate the quasi-optimal trajectory of rotorcraft UAV in a complex three-dimensional space (including real and false three-dimensional space) and to adopt a multi-objective cost function. In the process of path planning, some concepts of the genetic algorithm are applied to implement the WPS algorithm.

The optimization strategies can effectively improve the performance of WPA, but there are few literature studies on the wolf pack algorithm dealing with complex discrete problems, such as whether it could be better applied to discrete issues such as UAV task allocation.

In order to enable the wolf pack algorithm to better solve the discrete problem, this paper proposes a multiple discretized wolf pack algorithm (MDWPA). The contributions of this paper can be summarized as follows.

(1) In order to solve the shortcomings of traditional WPA in solving discrete problems, a multiple discrete optimization strategy is proposed to improve the traditional wolf pack algorithm. The improved WPA is compared with other intelligent algorithms in the simulation experiment, and its performance is far superior to other algorithms.
(2) Looking into the multi-UAV intelligence collection scenario in a dynamic battlefield environment, a corresponding multi-UAV task allocation model is established and, at the same time, resource consumption and mission revenue are considered. The objective function is constructed to solve the complex dynamic task assignment problem.

2. Task Assignment Model

In combat scenarios, in order to better attack enemy targets, UAVs are often required to collect enemy information and other intelligence in the enemy area, and after mastering comprehensive intelligence, they are required to perform tasks such as precision strikes on the enemy area. At the initial moment, the UAV departs from the base, is required to successfully detect the enemy target and obtain useful information, and execute the task sequence in turn before returning to the base. All UAVs are not initially dispatched, some UAVs will be kept at the base for emergencies.

2.1. UAV and Task Modeling

This article models the above actual combat scenarios, which can be expressed as a four-tuple $< U, T, M, C >$, where U is the set of UAVs, T is the set of targets in the combat area, and M is the set of tasks on the target. C is the set of constraints in the problem model. Based on the literature [32], this paper further considers the problem of the cooperative task allocation of UAVs in actual combat area scenarios, further considers multiple constraints and optimization objectives, and constructs a cooperative multi-task assignment model for UAVs in combat scenarios. Table 1 shows the attributes of three objects in the model.

Table 1. Object attributes and symbol interpretation.

	Attributes
UAV U (For any UAV u_i)	The total number of UAVs: N_u Initial position (the base) $P_i = (x_i, y_i)$ $(i = 0)$ Flying speed: v_i The maximum travel distance of UAV: d_i^{max} The maximum travel time of UAV: τ_i^{max}
Target T (For any target T_j)	The number of targets: N_t Position coordinates: $P_j = \left(x_j, y_j\right)$ Reconnaissance time required for completion of missions: I_j

The definition of the UAV system has N_u reconnaissance UAVs. In this paper, for any UAV $u_i(i \in [1, N_u])$, given its constant flight speed v_i, maximum navigation distance d_i^{max} and maximum flight time τ_i^{max}.

Due to the dynamic update of information in the battlefield environment, the intelligence information of each target point may be updated after reconnaissance. Therefore, each target point T_j $(j \in [1, N_t])$ involved in this article contains at least one reconnaissance mission. Among them, define N_t as the number of targets. I_j is the reconnaissance time required to complete the intelligence collection work at the target point T_j.

On the premise of ensuring the validity of the model, in order to simplify the model and reduce the difficulty of solving the model, this article puts forward the following reasonable assumptions about the model:

(1) The number and approximate location coordinates of all targets are known to the command center of UAV base;

(2) The UAV flies at a constant speed, and the flying heights of the UAVs are not consistent in order to prevent collisions between the UAVs;
(3) The same task can be completed by multiple UAVs, and it is assumed that the UAVs working in coordination reach the target point at the same time, and there is no waiting time for each other.

Based on the above symbolic interpretation and model assumptions, the scheme of cooperative multi-task allocation for UAVs can be expressed as a vector, as shown in Table 2. The vector has three rows, the first row represents the target sequence, the second row represents the task sequence on the target (mainly multiple intelligence searches, taking two searches as an example), and the third row represents the result of task assignment.

Table 2. The vector of cooperative task allocation for UAVs.

1	T_j
2	M_K
3	X_i^j

In Table 2, M_K represents the Kth mission of target T_j and it is assigned to u_i, so the third row X_i^j $(i \in [1, N_u])$ is 1. For other $i'(i' \neq i,\ i' \in [1, N_u])$, $X_{i'}^j$ is 0, which means that T_j is not assigned to $u_{i'}$.

2.2. Objective Function

In order to make the plan of task allocation more scientific and reasonable, this paper constructs a task allocation model under multiple constraints such as limited resources. The objective function of the constructed model mainly considers the minimization of resource consumption and the optimization of the task completion effect. Therefore, we construct a cost function and an evaluation function of task completion effect to quantify resource consumption and task completion effects respectively.

2.2.1. Cost Function

Most studies only consider the traveled distance or time in the evaluation of resource consumption, which makes the obtained task assignment result one-sided. To address this defect, we jointly consider the impact of distance and time on resource consumption. Following this idea, the distance cost and time cost of the UAV system is constructed as follows.

a. Distance Cost

The distance cost of the UAV system is denoted by d_{total}, which can be achieved by

$$d_{total} = \sum_{i=1}^{N_u} \sum_{j=1}^{N_t} \left(d_{jk\hat{j}}(i) x_{ijk} \right) \tag{1}$$

where $d_{jk\hat{j}}(i)$ represents the flight distance of the UAV u_i from the target point t_j to the target point $t_{\hat{j}}$; where x_{ijk} is a binary variable that represents the k execution of target t_j by UAV u_i, and the specific expression is shown as follows.

$$x_{ijk} = \begin{cases} 1,\ if\ M_k\ on\ t_j\ is\ assigned\ to\ u_i \\ 0,\ others \end{cases} \tag{2}$$

If there are no obstacles in the flight path from the task point t_j to the next task point $t_{\hat{j}}$, the distance $d_{jk\hat{j}}$ can be easily obtained according to the calculation Formula (3).

$$d_{jk\hat{j}}(i) = \sqrt{\left(x_{t_j}^k(u_i) - x_{t_{\hat{j}}}^{k'}(u_i) \right)^2 + \left(y_{t_j}^k(u_i) - y_{t_{\hat{j}}}^{k'}(u_i) \right)^2} \tag{3}$$

where $[x^{k}_{t_j}(u_i),\ y^{k}_{t_j}(u_i)]$ and $[x^{k'}_{t_{\hat{j}}}(u_i), y^{k'}_{t_{\hat{j}}}(u_i)]$ represent the position coordinates of the UAV u_i in executing the task M_k of the target t_j and the task $M_{k'}$ of the target $t_{\hat{j}}$, respectively.

b. Time Cost

In order to ensure that the tasks are completed as quickly as possible, the model also considers the completion time of all tasks as another optimization index, where the shortest completion time of all tasks can ensure that the entire task can be completed quickly. The time cost of the UAV system to perform the task is represented by τ_{total}.

$$\tau_{total} = max\left(\tau^{e}_{ijk} + \tau^{f}_{ijk}\right) \tag{4}$$

where τ^{e}_{ijk} and τ^{f}_{ijk} represents the time that u_i execute M_k on t_j and the flight time of the UAV u_i from t_j to the next target point $t_{\hat{j}}$, respectively. The τ_{total} is the maximum sum of τ^{e}_{ijk} and τ^{f}_{ijk}. τ^{e}_{ijk} is a predetermined parameter, and the calculation formula of τ^{f}_{ijk} is shown in Formula (5).

$$\tau^{f}_{ijk} = \frac{d_{jk,\hat{j}}(i)}{v_i} \tag{5}$$

This paper defines a cost function to quantify the resource consumption of the UAV system. Distance and time are the main indicators to measure resource consumption. Therefore, the distance cost and time cost of the UAV system should be the main components of the cost function. However, the evaluation indicators of distance and time are in different numerical dimensions, the direct addition of these two indicators affects the fairness of the cost function. Therefore, these two indicators should be standardized to eliminate dimensional effects. Based on the above analysis, the cost function of the UAV system can be obtained as followed.

$$f_{cost} = \frac{d_{total}}{\sum_{i=1}^{N_u} d_i^{max}} + \frac{\tau_{total}}{\underset{i\in[1,N_u]}{max}\ \tau_i^{max}} \tag{6}$$

The first and second terms of Formula (6) measure resource consumption in terms of distance and time, respectively. In order to eliminate the dimensional influence between distance and time indicators, divide the different indicators by their respective upper bounds (the maximum flight distance of the UAV system $\sum_{i=1}^{N_u} d_i^{max}$ and the endurance time of the UAV system $\underset{i\in[1,N_u]}{max}\ \tau_i^{max}$) for normalization.

2.2.2. Task Revenue Function

In a battle scenario, mastering the enemy's important intelligence is the key to victory. Assuming that, at the initial moment, the UAV only knows the approximate location of the target. When the UAV detects the target, the accurate location information of the target $P_j = \left(x_j, y_j\right), j \in [1, N_t]$ and the initial value of the target $value\left(t_j\right)$ can be obtained. This article uses the initial value of the target to express the importance of intelligence.

In actual scenarios, the initial value of the task tends to decrease as the waiting time increases, so this paper considers that the effect of task execution will decrease with the increase of time. That is to say, the function value is inversely proportional to the time. To eliminate the dimensional influence between costs and benefits, we also normalize the benefits by dividing the actual revenue by the sum of the initial values of all the tasks. The specific operation is shown in Formula (7).

$$f_{rew} = \frac{\sum_{i=1}^{N_u}\sum_{j=1}^{N_t}\left(x_{ijq}\cdot value\left(t_j\right)\cdot\varphi(t)\right)}{\sum_{j=1}^{N_t} value\left(t_j\right)} \tag{7}$$

where x_{ijq} is a binary value, indicating whether UAV u_i is assigned to a task t_j. Only when μ_q is an attack task, the value of x_{ijq} is 1; $value(t_j)$ is the importance of the target at the initial moment, and $\varphi(t) \in [0,1]$ decreases with time, indicating that the value obtained by destroying the target varies with time, and the calculation formula of $\varphi(t)$ is as follows.

$$\varphi(t) = e^{-\rho t} \tag{8}$$

where ρ represents the rate of decline in revenue, the larger the ρ, the faster the revenue from destroying the target decreases over time, and vice versa.

However, in a real scenario, enemy information is dynamic and constantly updated. UAVs may be required to perform repeated or multiple intelligence searches at the same target point to ensure effectiveness and accuracy. But in a dynamic mission scenario, the optimal solution is to cover as many target points as possible in a short time. Therefore, UAV should try to select unsearched target points, rather than repeatedly search the same mission point. Therefore, we adjust the function of task revenue according to the actual situation. When UAVs conduct repeated searches of the same target, the information they find will largely overlap, so the value of the mission should be halved.

The change of the function of task value is shown in Formula (9).

$$value(t_j) = (\frac{1}{2})^{k-1} * value(t_j) \tag{9}$$

where k is the number of UAV searches for intelligence on task point t_j.

2.3. Task Allocation Model

Comprehensive consideration of the UAV body constraints, task requirements, cost consumption, effects of task completion and other factors, a task allocation model under multiple constraints is established.

$$maxf = \alpha \frac{f_{rew}}{f_{cost}} \tag{10}$$

$$s.t. \sum_{i=1}^{N_u} \sum_{j=1}^{N_t} \sum_{k=1}^{3} x_{ijk} = 3 \tag{11}$$

$$\forall i \in [1, N_u], d_i \leq d_i^{max}, \tau_i \leq \tau_i^{max} \tag{12}$$

$$\sum_{i=1}^{N_u} \tau_{ijk}^{e} X_i^j \geqslant I_j \tag{13}$$

where α is the binary flag, which can be expressed as follows.

$$\alpha = \begin{cases} 1,\ if(9) \sim (14) are\ all\ satisfied \\ 0, others \end{cases} \tag{14}$$

Formula (11) is used to ensure that each task must be completed; in Formula (12), d_i and τ_i, respectively, represent the total flight distance and total time consumed by the UAV u_i to complete all assigned tasks, neither of which can exceed the prescribed upper limit.

Formula (13) is to ensure that each task is successfully completed, and the resources consumed by the UAV performing the task need to meet the conditions required for the completion of the task.

In the formulated task allocation model, the cost function and the function of task benefit evaluation are used as the numerator and denominator of the objective function, respectively. By maximizing the objective function proposed, the contradictory variables of task revenue and resource consumption are optimized at the same time. Meanwhile, constraints such as limited resources and task priority are introduced to improve the applicability of the model. The introduction of multiple constraints makes the established model more accurate and further improves the mission execution efficiency of the UAV system.

3. The Wolf Pack Algorithm

3.1. Traditional Wolf Pack Algorithm

The wolf pack algorithm is a relatively new swarm intelligence algorithm based on the wolf pack behavior mechanism [17]. The entire wolf pack algorithm mainly contains two mechanisms, namely, the head wolf generation mechanism, and the wolf pack update mechanism, and three behaviors, namely, wandering behaviors, summoning behaviors, and siege behaviors. The following gives a brief description of the two mechanisms and three behaviors in the wolf pack algorithm to prepare for the specific analysis of the improved wolf pack algorithm.

3.1.1. The Head Wolf Generation Mechanism

The head wolf generation mechanism means that, according to the "victor is king" rule, when a more powerful leader appears in the wolves to prey on prey, a more capable wolf will become a new head wolf. The prey odor concentration perceived by the artificial wolf is $Y = f(X)$, where Y is the value of the objective function. That is, when the fitness of the ith wolf in the pack Y_i (prey concentration at its location) is greater than the fitness Y_{lead} of the head wolf, this wolf becomes the new head wolf.

3.1.2. Wolf Pack Update Mechanism

The wolf pack update mechanism refers to removing the R wolf pack individuals with the worst value of objective function, and then adding R wolf pack individuals at the same time.

$$R = randi(\left[\frac{N}{2 \cdot \beta}, \frac{N}{\beta}\right]) \tag{15}$$

among them, β is the population update scale factor; N is the number of wolves.

3.1.3. Wandering Behavior

Except for the head wolf, the most elite $Snum$ wolves are called the detection wolf. If α is the scale factor of the detection wolf, the number of the detection wolf $Snum$ is a random number between $\left[\frac{N}{1+\alpha}, \frac{N}{\alpha}\right]$. The detective wolf walks one step in the surrounding h directions with its walking step $step_a$, and respectively senses the odor concentration of the location. Every time it travels in one direction, the odor concentration is sensed and recorded, and then it returns to the original coordinates. If there is a direction where the odor concentration is greater than the original coordinate, then it moves to the direction with the largest odor concentration. After detective wolf i advances in the $p(p = 1, 2, \ldots, h)$ direction, its position in the dth dimension is shown as follows.

$$Z_{id}^{p} = Z_{id} + \lambda \cdot step_a^d \tag{16}$$

among them, Z_{id} is the coordinates of the ith wolf in the d-dimensional space; λ is a random number in the range $[-1, 1]$. Compare the updated fitness of the wolf detection with the fitness of the head wolf. If $Y_i > Y_{lead}$, then update $Y_{lead} = Y_i$, and detective wolf i will replace the head wolf and re-initiate the call; if $Y_i < Y_{lead}$, continue the wandering behavior.

3.1.4. Summoning Behavior

As the wolves enter the summoning phase, the fierce wolves summoned by the head wolf rush towards the coordinates of the head wolf with a larger step length $step_b$, the position change of the fierce wolf i in the d-dimensional space in the $k+1$ iteration is shown in Formula (17).

$$Z_{id}^{k+1} = Z_{id}^{k} + step_b^d \cdot \left(Z_{lead}{}_d^k - Z_{id}^k\right) / \left|Z_{lead}{}_d^k - Z_{id}^k\right| \tag{17}$$

among them, Z_{id}^k is the current position of the wolf; $Z_{lead}{}_d^k$ is the coordinate of the kth generation wolf in the d-dimensional space. In the process of running, if the adaptability of the wolf i is greater than that of the head wolf, the wolf i becomes a new generation of the head wolf, the position of the head wolf is updated, and the calling behavior is initiated; if the adaptability of the wolf i is less than the wolf, and the distance d_{is} between wolf i and the head wolf is less than d_{near}, they switch to siege behavior; supposing the value range of the dth variable to be optimized is $[min_d, max_d]$, then the judgment distance d_{near} can be determined by the Formula (18).

$$d_{near} = 1/(D \cdot \omega) \cdot \sum_{d=1}^{D} |max_d - min_d| \tag{18}$$

among them, ω is the distance determination factor, and its different values will affect the convergence speed of the algorithm. max_d and min_d are the upper and lower bounds, respectively.

3.1.5. Sieging Behavior

When the wolves are close enough to the prey, the fierce wolves initiate a siege on the prey. At this time, the updated position Z_{id}^{k+1} of the wolf can be expressed by Formula (19).

$$Z_{id}^{k+1} = Z_{id}^{k} + \lambda \cdot step_c^d \cdot \left| Z_{lead}{}_d^k - Z_{id}^k \right| \tag{19}$$

among them, λ is a random number in the range $[-1, 1]$; $step_c^d$ is the siege step length. If the fitness of the wolf i after the siege behavior is greater than the fitness of the original position, its position will be updated; otherwise, the wolf will keep the original position unchanged.

Assuming that the value range of the dth variable to be optimized is $[min_d, max_d]$, then in the wolf pack algorithm process, three steps are involved: walking step size $step_a$, rush step size $step_b$, siege step size $step_c$, the step size in the d-dimensional space has the following relationship.

$$step_a^d = \frac{step_b^d}{2} = 2 \cdot step_c^d = |max_d - min_d|/S \tag{20}$$

where S is the step size factor and represents the fineness of the artificial wolf searching for the optimal solution in the solution space.

The optimization process of the WPA algorithm is divided into wandering, calling, and sieging. In the WPA, the exploring wolves select H directions and explore them. There is no communication between them and so there will be repetition in exploring space, so there is a lack of necessary information communication between artificial wolves, and the algorithm is not global enough. At present, it is difficult to find the global optimal solution for the wolf pack algorithm. Therefore, the continuous exploration of swarm intelligence optimization algorithm has become a hot topic for many scholars.

3.2. Proposed Algorithm

The wolf pack algorithm is mainly composed of three behavior mechanisms: wandering, summoning and sieging. Through these three behaviors, the position of each wolf is gradually optimized and the global optimum is finally found. The traditional wolf pack algorithm has strong convergence and global search ability and is not easy to fall into invalid search. WPA usually has good performance for continuous problems, but is not good at discrete problem optimization. The task assignment problem of UAV is a discrete problem, in which the variables of each dimension are integers belonging to a set. Therefore, based on the WPA rules, it is necessary to improve the WPA to match the integer discrete characteristics of task assignment. In order to improve solution accuracy and efficiency of the algorithm, this paper uses the multi-step discrete optimization strategy to improve the WPA, which mainly includes individual coding, wandering strategy, calling strategy, siege strategy, and recruitment of new individuals. The details of the MDWPA are as follows and the algorithm is shown in Algorithm 1.

3.2.1. Individual Coding and Initialization

In the improved algorithm, a hybrid coding method based on task assignment and path planning is adopted for mixed variables. The position vectors of individuals contain discrete variables representing the results of task assignment. In order to generate a feasible task allocation plan more effectively, according to the characteristics of UAV task allocation problem, the algorithm adopts a mixed variable coding method based on task allocation and path planning and designs an appropriate constraint processing method to initialize the position vector of wolf individuals according to the coding method. At the same time, new individuals are generated by a recombination method based on structure learning to further improve the quality of the population. In addition, in the process of population renewal, in order to obtain more excellent individuals, the improved algorithm adopts the mechanism of coevolution, that is, a coevolutionary population and the original population compete to generate new individuals.

At present, the existing coding methods can only describe the final assignment scheme but cannot represent the executing sequence of tasks, that is, the path planning result of the task performed by UAV. Therefore, in order to represent both the result of task assignment and the result of UAV's path planning during task execution, a hybrid variable coding method based on task assignment and path planning is proposed in this paper. This method codes according to the specific situation of the UAV in the execution of the task. The code of each individual contains two parts, that is, $P = (MN, UN)$, where the first part MN is a discrete variable, recording the task number, and the second part UN is a discrete variable, which records the number of UAVs performing corresponding tasks.

Figure 1 shows the code of an individual corresponding to the model of Section 2. It contains 6 targets, there are 3 UAVs, numbered 1–3. The two lines of the position vector of the individual wolf are used to represent the task assignment result. For example, the targets T1, T4 and T6 are assigned to the UAV U_1 for execution. At this time, this is the corresponding task assignment plan vector: $X_1^k = 1\ (k \in \{1\ 6\})$.

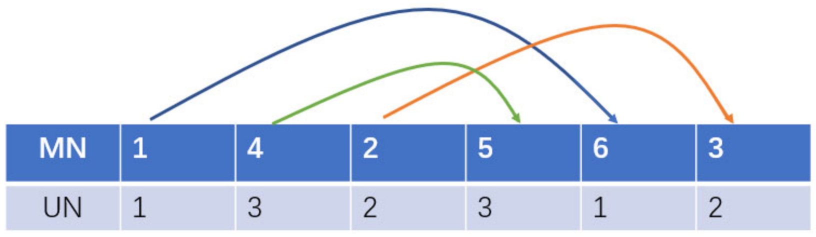


Figure 1. The hybrid coding method.

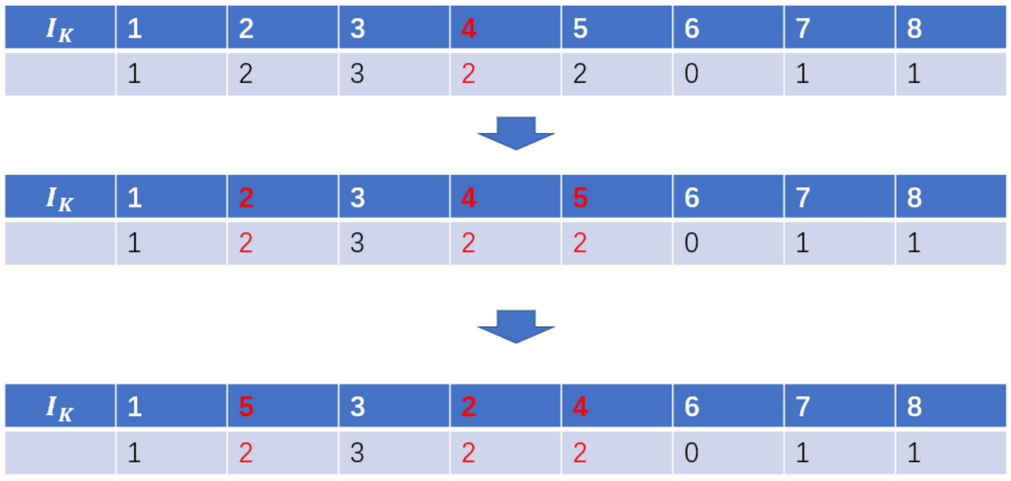
In addition, in order to show the results of the UAV's path planning, the order in which the task numbers appear in the position vector represents the order in which the UAV performs tasks. The blue curve in Figure 1 indicates that UAV U_1 performs tasks in sequence T_1,T_6. The orange curve indicates that the execution tasks of the UAV U_2 are T_2, T_3. The green curve indicates that the execution tasks of the UAV U_3 are T_4 and T_5, in turn. Because the coding method uses a hybrid variable coding method based on task allocation and path planning, this variable can not only efficiently convert the position vector of the individual wolf pack into a task allocation plan of the model, but can also fully consider the path uncertainty during the execution of the UAV task and use the order of tasks in the position vector to explicitly show the order in which UAVs perform tasks. This coding method can generate a definite task allocation plan and the path planning results of the UAV when the individual is initialized.

In the actual initialization operation in the UAV task allocation, we first initialize the task number in the first row. In order to provide a global orientation to the algorithm, randomly scramble $[0, 1, \ldots, W]$ and assign it to the first row (W is the last mission number), and then the UAV number in the second line is initialized by random assignment.

3.2.2. Improvement on Walking Behavior

The walking behavior of wolves is essentially an active exploration of the unknown environment, which determines the global search capability of the algorithm. In the WPA, each exploring wolf scouts the prey from h directions divided equally by 360 degrees, which is represented by Formula (16), and adjusts the coverage of the scout by changing the size of h. The bigger h is, the larger the coverage of the scout will be, but the speed of the scout will be relatively slower.

In order to better solve the convergence speed of the optimal solution in discrete problems, this paper proposes that the updating method of Formula (16) is no longer used when each exploring wolf finds the optimal forward direction. Instead, it states that all exploring wolves, in turn, select the position variables in each column, read the number of the UAV performing that task, find all the task numbers assigned to that UAV, and then randomly rank those task numbers and compare them to the original individual. If the change is better, keep the updated plan, otherwise leave it unchanged. The specific operation steps are shown in Figure 2.



I_K	1	2	3	4	5	6	7	8
	1	2	3	2	2	0	1	1

I_K	1	2	3	4	5	6	7	8
	1	2	3	2	2	0	1	1

I_K	1	5	3	2	4	6	7	8
	1	2	3	2	2	0	1	1

Figure 2. The walking strategy.

3.2.3. Improvement on Calling Behavior

The calling behavior is an essential process for fierce wolves to converge to the current optimal solution which is the position of the leader wolf. This behavior determines the convergence rate of the algorithm. The traditional wolf pack algorithm's calling behavior, when the fierce wolves approach the leader wolf in step length, will lead to the convergence speed of the algorithm being too slow. Inspired by replication of gene fragment in genetic algorithm, each of the fierce wolf copy part of the leader's position to replace its own position, so as to achieve fast proximity to the leader wolf and achieve fast convergence of the algorithm.

The position of the fierce wolf should be updated by learning the position of the leader wolf. The specific operations are as follows: Firstly, keep the order of the tasks in the first line of the position variable of the fierce wolf unchanged; finally, update the UAV task assignment scheme in the second line by learning the UAV task assignment scheme of the leader wolf. The specific steps are shown in Figure 3.

Given that the calling step of the improved strategy is a direct learning of the head wolf, the distance judgment operation of the traditional wolf pack algorithm is not used here, and the elite strategy is used instead, that is, if the fierce wolf is worse than the original, it will not be updated.

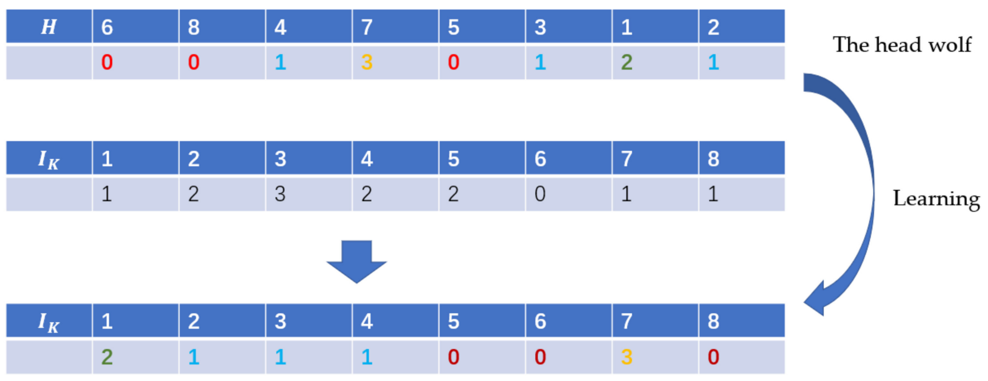


Figure 3. The calling strategy.

3.2.4. Improvement on Sieging Behavior

The main purpose of the improved siege strategy is to enhance the algorithm's global search capabilities. The operations are as follows.

In order to make the algorithm have better convergence effect and optimization ability in the later stage of the iteration, this article will adopt two strategies to improve the siege behavior. First, traverse all individuals with a probability of 0.5, randomly select a column of random integers between $[n/(2 \times \vartheta),\ n/\vartheta]$ for each individual, ϑ is a scale factor, and re-assign the UAVs number under these columns randomly. Or perform a mutation operation with a probability of 0.5. That is, an individual with a higher fitness value in the population is randomly selected, a few columns are randomly selected from the position information of the individual variable, and the two elements of the column are randomly allocated to obtain a new individual. After improvement, if the effect is better, update it, otherwise it will remain unchanged. The specific steps are shown in the Figures 4 and 5, respectively.

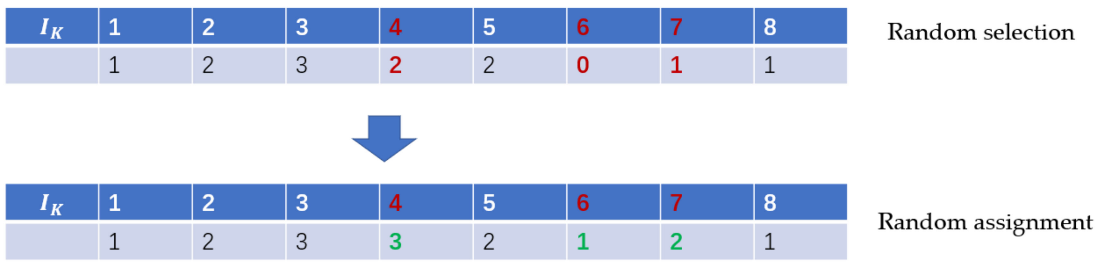


Figure 4. The siege strategy using random assignment.

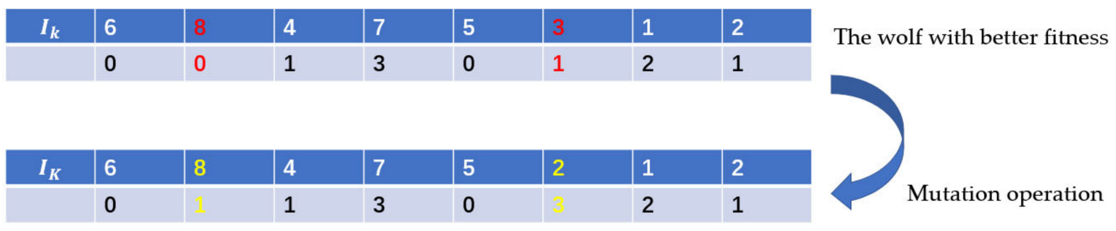


Figure 5. The siege strategy using mutation operations.

3.2.5. Replenishment of New Individual

The diversity of the wolf population algorithm in the later stage of the population is maintained by the "survival of the strong" mechanism, which is not effective for complex problems. The reason is that it only relies on the initialization mechanism to supplement new individuals, and it is not competitive with other individuals that have been screened later. However, the traditional optimization strategy only uses the position information of the copy of the wolf, and only a slight exchange strategy. In the later stage of the algorithm iteration, the new individuals generated by this method do not have enough diversity, which can easily cause the algorithm to enter local convergence. Therefore, this article proposes a new individual generation strategy.

Among them, individual wolves perform exchange operations with a probability of 0.5, that is, they randomly find two tasks performed by different UAVs in the position variables of the head wolves, and then exchange the numbers of the UAV that perform these two tasks. Or they perform mutation operations with a probability of 0.5. That is, an individual with a higher fitness value in the population is randomly selected, a few columns are randomly selected from the position information of the individual variable, and the two elements of the column are randomly allocated to obtain a new individual. The specific steps are shown in Figures 6 and 7, respectively. If the mutant individuals of the head wolf are used for individual supplementation, in the later stage of the iteration, the similarity between each individual and the head wolf becomes high, which makes it easy for the population to fall into the local optimum. Therefore, this paper adopts the probability of 0.5 to use mutant individuals with higher fitness values to supplement new individuals. This method maintains the diversity of the population and prevents the algorithm from falling into a local optimum.

Algorithm 1: MDWPA

```
Input: initial position of UAVs: posV, position of targets: posT, iterations, population size: N, Tmax
Output: solution xbest
Generate initial solution X;
for iterations do
    Select the leader wolf Lx;
    /* walking behavior (See Section 3.2.2) */
Select S exploring wolves;
walking times = 0
    while walking times < $T_{max}$ do:
        for exploring wolves $X_{ew}$ do
            [
            individual variation according to Section 3.2.2 and Figure 2;
]
        if $F(X_{ew}') > F(X_{ew})$ then
            update position: $X_{ew} \leftarrow X_{ew}'$
        end
        if $F(X_{ew}') > F(X_{leader})$ then
            update position: $X_{leader} \leftarrow X_{ew}'$
            break;
        end
    end
/* calling behavior (See Section 3.2.3) */
Select M fierce wolves;
    for fierce wolves $X_{fw}$ do
        [
        get close to the leader wolf according to Section 3.2.3;
        ]
        if $F(X_{fw}') > F(X_{fw})$ then
            update position: $X_{fw} \leftarrow X_{fw}'$
        end
        if $F(X_{fw}') > F(X_{leader})$ then
            replace leader wolf: $X_{leader} \leftarrow X_{fw}'$
            break;
        end
end
```

Algorithm 1: *Cont.*

```
/* sieging behavior (See Section 3.2.4) */
    for all wolves except the leader wolf $X_w$ do
        [
        $r$ = rand (0,1)
        if $r < 0.5$ then
            individual variation according to Section 3.2.4 and Figure 4;
    else
            get close to the wolf with better fitness according to Section 3.2.4 and Figure 5;
        end
]
        if $F(X_w') > F(X_w)$ then
            update position: $X_w \leftarrow X_w'$
        end
end
/* generating new individuals (See Section 3.2.5) */
Select R weakest wolves $X_N$
    for $X_N$ do
        [
        $r$ = rand (0,1)
        if $r < 0.5$ then
get close to the leader wolf according to Section 3.2.5 and Figure 6;
        else
            get close to the wolf with better fitness according to Section 3.2.5 and Figure 7;
        end
]
        if $F(X_N') > F(X_N)$ then
            update position: $X_N \leftarrow X_N'$
        end
end
Select optimal individual xbest in the population.
end
```

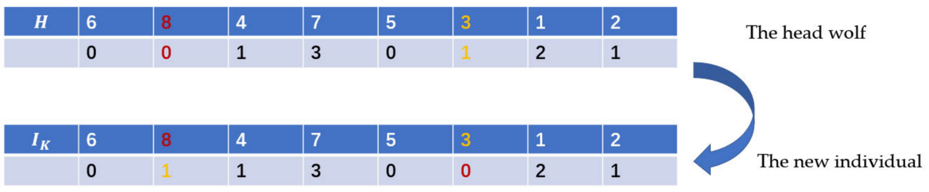


Figure 6. Create new individuals by replicating the head wolf.

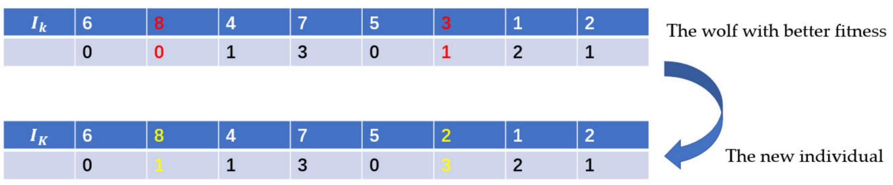


Figure 7. Generate new individuals in a mutated way.

4. Simulation

4.1. Simulation Setup

The simulation is performed on a 2.30 GHz Intel i5 processor and Python 3.8.0 on a computer with 4 GB memory.

Experimental setting parameters are shown in Table 3.

Table 3. Experimental parameter settings.

Parameter	Value
Number of wolves: N	100
Dimension of the search space:	50/100/150
The maximum number of iterations: k_{max}	200/400/600
Distance judgement factor: ω	800
Step factor: S	1000
Detective Wolf Ratio Factor: α	4
Update scale factor: β	6
Maximum number of walks: T_{max}	10

The setting of related parameters of the wolf pack algorithm has been analyzed in the literature [17], and the parameter settings in Table 3 can optimize the performance of the algorithm.

4.2. Parameter Analysis of MDWPA

In order to obtain the best performance of MDWPA, the optimization strategy of each stage is worth further study. There are four optimization steps in MDWPA, namely, discrete optimization of wandering behavior, discrete optimization of calling behavior, discrete optimization of besieging behavior, and discrete optimization strategy of new individual supplement. In order to verify the effect of the above four optimization steps and optimal performance, this paper uses the orthogonal experimental design (OED) method [33], with the objective function value and the average distances as the evaluation index, 16 kinds of algorithm have been designed, which run independently under the UAV task assignment model 200 times in order to verify which combination of discrete optimization operations would achieve the best optimization performance of MDWPA. In order to avoid contingency, we conduct 30 independent repeated experiments for each scheme, and record the mean, max, and variance of the objective function optimization. The settings of each parameter in the algorithm are shown in Table 3, and the relevant parameter settings of the algorithm are set according to literature [17], in which a large number of experiments prove that the data in Table 3 is the optimal parameter combination.

In Table 4, P1 represents the discrete optimization strategy of wandering behavior. P2 represents the discrete optimization strategy of calling behavior. P3 represents the discrete optimization strategy of siege behavior. P4 represents the new individual recruitment strategy. Through the above 16 different algorithm combinations, simulation experiments are conducted to determine which phase of the optimization strategy has the greatest impact on the algorithm performance. As can be seen from the results in Table 4, the combined optimization strategy in the first and fourth stages has the greatest impact on algorithm performance. It can be seen from the results that MDWPA has certain advantages over the other 15 algorithms in terms of optimal value and average value of objective function.

4.3. Performance of MDWPA

In order to fully analyze the performance and advantages of MDWPA, this paper takes the objective function in Section 2 as the optimization object. By comparing and analyzing the experimental data, the performance of MDWPA algorithm is evaluated from efficiency and stability.

4.3.1. Efficiency

In the UAV task allocation model described in Section 2, for each target's military mission, the UAV needs to start from the base and rush to the coordinates of the target in turn, the coordinate's unit is kilometers. The UAV needs to spend a certain period of time at the target to complete the intelligence search task (the unit of time is seconds). Each instance contains five UAVs and 25 military mission targets, which are randomly set in a fixed position. The attributes of each UAV include flight speed and initial coordinates. The attributes of each target are the starting coordinates, initial value, and the execution time. Table 5 is the attributes of targets in Example 1; Table 6 is the attributes of UAVs in Example 1.

Table 4. The design scheme of orthogonal experiment design.

No.	Algorithm	Average	Max	Std
1	WPA	0.094311828	0.098515562	0.003123479
2	WPA + P1	0.094406883	0.099737827	0.003602062
3	WPA + P2	0.100099429	0.104996064	0.003406232
4	WPA + P3	0.255877633	0.277344851	0.012403991
5	WPA + P4	0.089991202	0.099891448	0.003966736
6	WPA + P1 + P2	0.101346873	0.104414414	0.002610191
7	WPA + P1 + P3	0.128543916	0.133154261	0.002759905
8	WPA + P1 + P4	0.266737265	0.292252801	0.017772005
9	WPA + P2 + P3	0.259615261	0.291802999	0.012754875
10	WPA + P2 + P4	0.091592871	0.099638071	0.003605891
11	WPA + P3 + P4	0.252197181	0.270349845	0.011637504
12	WPA + P1 + P2 + P3	0.263325287	0.295271752	0.013132305
13	WPA + P1 + P2 + P4	0.121476444	0.125332704	0.003436636
14	WPA + P1 + P3 + P4	0.263325611	0.294600559	0.018923418
15	WPA + P2 + P3 + P4	0.255140928	0.276746983	0.016519906
16	WPA + P1 + P2 + P3 + P4	0.275775987	0.315469438	0.011828138

Table 5. The attributes of targets in Example 1.

No.	Coordinate	Initial Value	Execution Time (s)	No.	Coordinates	Initial Value	Execution Time (s)
0	(13,41)	80	8	13	(90,22)	92	15
1	(61,83)	90	11	14	(95,20)	95	12
2	(79,12)	40	3	15	(81,86)	93	12
3	(41,98)	90	14	16	(16,65)	68	7
4	(23,65)	95	22	17	(85,51)	82	9
5	(12,91)	79	12	18	(23,72)	62	16
6	(68,54)	73	17	19	(70,83)	60	18
7	(82,50)	84	18	20	(23,30)	90	4
8	(20,50)	81	3	21	(65,30)	60	16
9	(92,48)	69	18	22	(87,77)	94	11
10	(69,59)	71	10	23	(93,18)	83	12
11	(59,10)	70	15	24	(19,19)	90	8
12	(12,51)	62	20				

Table 6. The attributes of UAVs in Example 1.

No.	Initial Coordinate	Velocity (km/s)
0	(0,0)	0.1
1	(0,0)	0.12
2	(0,0)	0.16
3	(0,0)	0.11
4	(0,0)	0.1

In order to prove the efficiency of the MDWPA algorithm in solving the UAV task allocation model, a variety of optimization algorithms are selected to compare with the MDWPA. Some of the algorithms involved are for continuous problems and others are for discrete problems. Therefore, it is difficult to use continuous benchmark functions for unified performance evaluations. In this paper, we select the most widely used intelligent algorithms, PSO [34], GA [35], WPA, PSO-GA-WPA [31] (For brevity, PSO-GA-WPA is abbreviated as GPWPA), CJADE [36] and EBOCMAR [37].

For the fairness of the experiment, the population size is set to 100. In addition, the learning factors c1 and c2 in the particle swarm algorithm are set to 1; the crossover probability in the genetic algorithm is set to 0.8, and the mutation probability is set to 0.2. In order to ensure the accuracy of the results of the simulation experiment, each algorithm will be run 200 times on each instance individually, and the iterative effect analysis curve of each algorithm will be obtained. Figure 6 is the comparison of iterative curves of algorithms on test cases 1.

In Figure 8, the blue, red, orange, purple, green, pink, and brown curves represent the iterative curves of MDWPA, GA, GPWPA, PSO, WPA, CJADE and EBOCMAR, respectively. The purpose of the comparison with the traditional WPA and PSO is to illustrate our efforts on solving discrete problems with traditional biological optimization algorithms. But unfortunately, direct reference to these continuous algorithms is not ideal for simulation results. Comparing MDWPA with the three algorithms for discrete problems, GA, CJADE and EBOCMAR, our proposed MDWPA has advantages on convergence speed and accuracy. GA is a traditional optimization algorithm for discrete problems, and the improved optimization algorithm CJADE and EBOCMAR are newer and can solve discrete problems very well. It is clear from the figure that the GPWPA algorithm has a fast convergence rate in the initial stage, but with the increase of the number of iterations, the algorithm falls into a local convergence state. Although the genetic algorithm has good optimization ability, the convergence speed of genetic algorithm becomes slow after 75 iterations, resulting in local convergence problem. Among them, CJADE can only solve discrete problems, into both continuous and discrete processing. However, the results are not as good as the improved wolf pack algorithm in dealing with the high-dimensional complex task assignment of UAV. As other algorithms deal with continuous problems, they are not effective in dealing with the discrete problem of task allocation for UAVs. Obviously, the MDWPA algorithm has a better convergence speed and assignment effect than other algorithms when dealing with the UAV task assignment model. This is mainly related to the new individual recruitment strategy. With a probability of 0.5, the mutation of the head wolf individual is used for the recruitment of new individuals, and the mutation of the individual with a higher fitness value is used for the recruitment of new individuals. In this way, the diversity of population is kept and the algorithm is prevented from falling into local optimum.

This paper uses variance analysis and crunches other data to compare and analyze the performance of MDWPA and six other algorithms. We carried out the experiments in the first scenario of the model, and the parameters of each algorithm were set as they were in Table 3. A total of 30 independent repeated experimental samples were taken for each algorithm. The algorithm performance was analyzed by function mean, function standard deviation, function maximum, and mean time of each algorithm. The experimental results are shown in Table 7.

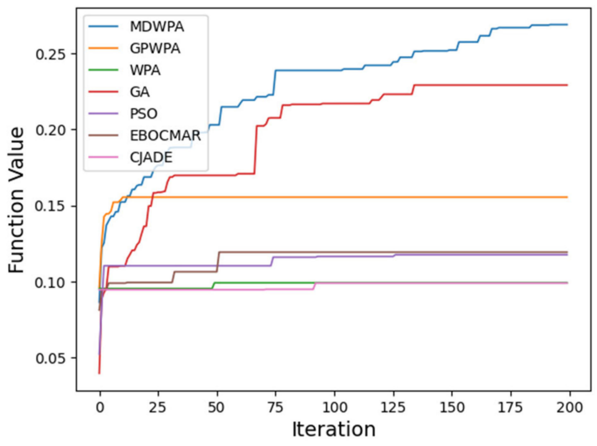


Figure 8. Comparison of iterative curves of algorithms on test cases 1.

Table 7. The experimental results of variance analysis.

Algorithm	Average	Std	Max	Time (Avg)
GPWPA	0.162847772	0.007316736	0.178401271	5900.154593
CJADE	0.101951109	0.006901708	0.115765696	649.6378847
EBOCMAR	0.11767223	0.006358509	0.138806691	161.0406503
GA	0.218305764	0.024635596	0.282582955	61.89460519
MDWPA	0.263548569	0.01590524	0.290838572	2827.711819
PSO	0.122822677	0.007341032	0.146461445	72.66523946
WPA	0.093725006	0.003100226	0.101597478	397.2063249

By analyzing the data in Table 7, we can see that CJADE, EBOCMAR, PSO and WPA do not have good effects when processing the model in this paper. The function values generally converge at a low level and fall into the local optimal solution. GPWPA improves the function search accuracy in a huge cost of running time, but it is still not ideal. GA has good search accuracy and short running time, but the search accuracy is not good compared with MDWPA proposed in this paper, and it is still locally optimal. In addition, although the MDWPA in this paper has a longer running time, it is also within a reasonable range, and the time complexity does not increase compared with the original WPA, especially for the task assignment scenarios with low real-time requirements.

In summary, the MDWPA has better results in dealing with the model of UAV task assignment, whether it is compared with other intelligent algorithms or compared with the existing improved wolf pack algorithm.

4.3.2. Stability

In order to further verify the stability of the proposed algorithm in complex scenarios. We redesigned two examples to increase the number of UAVs and military targets, respectively. And further compare and analyze with other algorithms.

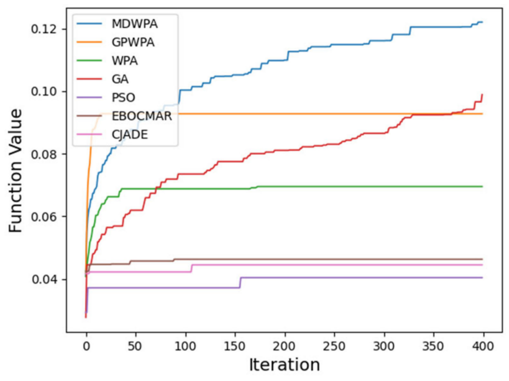
Example 2 includes 10 UAVs and 50 military targets; each time it runs, the maximum number of iterations is set to 400; the search space dimension is set to 100; Example 3 includes 15 UAVs and 75 military targets; the maximum number of iterations is set to 600; the search space dimension is set to 150; Figure 9a,b show the experimental results of Example 2 and Example 3, respectively.

In the Figure 9, the blue, red, orange, purple, green, pink, and brown curves represent the iterative curves of MDWPA, GA, GPWPA, PSO, WPA, CJADE, and EBOCMAR, respectively. It can be clearly seen from the Figure 9a,b that, with the increase of iterations and task complexity, GPWPA performs better than the genetic algorithm compared with Example 1, which also proves the advantage of the wolf pack algorithm in dealing with complex problems. At the end of iteration, GPWPA still fell into local convergence, but it is obvious from the figure that MDWPA not only has good optimization effect, but also still has high search ability at the end of iteration.

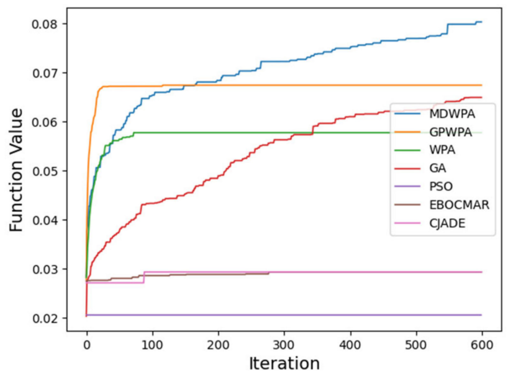
It is worth noting that as the number of UAVs and targets increases, the complexity and dimension of the problem also increase. Therefore, GA is inferior to the original WPA algorithm in Figure 9, which also confirms that the wolf pack algorithm has more advantages when it comes to solving high-dimensional complex problems than other optimization algorithms.

4.3.3. Comprehensive Analysis of MDWPA Performance

In the previous two subsections, the efficiency and stability of the proposed algorithm are compared, and it can be seen that MDWPA has obvious performance advantages in the UAV task assignment model. However, in order to more fully prove the practicability of the algorithm, the average flight distance is further used as an evaluation index, and the MDWPA algorithm is compared and analyzed with other intelligent algorithms. Figure 10 shows the average flight distance of the UAV of each algorithm in the three example scenarios.



(**a**)



(**b**)

Figure 9. Comparison of iterative curves of algorithms on test cases. (**a**) is the comparison of iterative curves of algorithms on Example 2; (**b**) is the comparison of iterative curves of algorithms on Example 3.

In Figure 10, the abscissa is the five intelligent algorithms, and the ordinate is the average flight distance of the UAV. The red (set1), blue (set2), and green rectangles (set3) represent the aforementioned Example 1, Example 2, and Example 3, respectively. Each intelligent algorithm has three bar graphs to represent the average flight distance of the UAV under the three mission scenarios. It can be clearly seen from Figure 10 that the MDWPA has obvious advantages compared to the other four intelligent algorithms. Regardless of the mission scenario, its average flying distance is the shortest, which proves that in the actual scenario, it can always find the best and shortest flight path for the UAV.

The above experimental results show that the efficiency and stability of the MDWPA in solving the UAV task allocation problem has considerable advantages compared with other algorithms. And it solves the shortcomings of the traditional wolf pack algorithm that can easily fall into the local optimum. Therefore, the improved wolf pack algorithm is suitable for large-scale and complex UAV task assignment and has relatively large performance advantages.

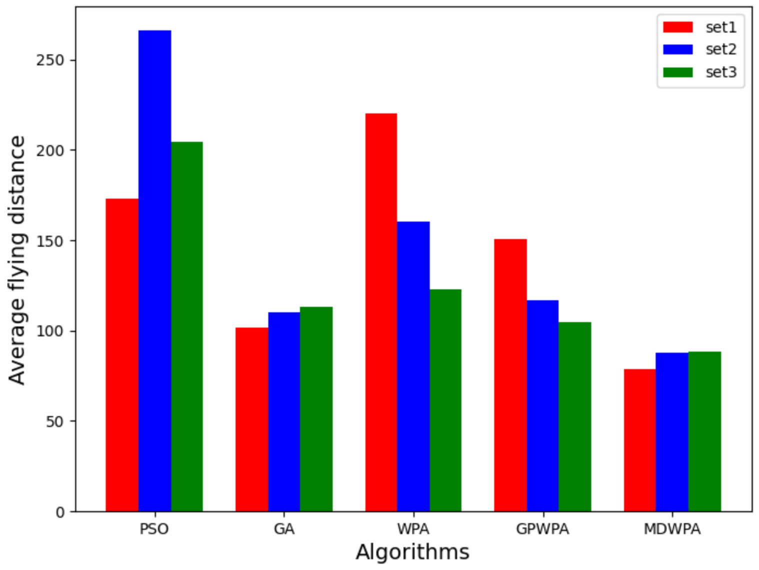


Figure 10. The average flight distance of the UAV in the three example scenarios.

5. Conclusions

This paper mainly studies the swarm intelligence algorithm based on the behavior mechanism of wolves and its application in the task assignment problem of UAV swarms. Firstly, a UAV task allocation model under the actual battlefield environment is established, and two evaluation indicators of resource consumption and task execution effect will be considered at the same time. The paper then summarizes and analyzes the principles and steps of the original WPA. After gaining a full understanding of the theoretical basis of the WPA, by analyzing the shortcomings of the wolf pack algorithm, in order to better solve the discrete problem of task allocation, a better performance multi-discrete wolf pack algorithm is proposed: MDWPA.

Through the simulation experiments, it was found that compared with WPA, PSO, GA GPWPA, and other algorithms, MDWPA has a greater performance advantage in terms of solving the problem of complex UAV task allocation.

In the future, more practical situations will be considered to further improve the existing model, such as building a model for a situation where too many tasks cause UAVs to complete all tasks due to limited resources. The next step of the research work will combine these new technical theoretical methods to improve existing algorithms to further improve the solution performance of the UAV cooperative multi-task assignment.

Author Contributions: Conceptualization, S.X., L.L., J.H. and Y.M.; methodology, S.X., L.L. and Z.Z.; validation, L.L. and Z.Z.; formal analysis, L.L., Z.Z. and S.X.; investigation, S.X. and L.L.; writing—original draft preparation, L.L., Z.Z. and S.X.; writing—review and editing, L.L.; supervision, S.X.; project administration, S.X. and Y.M.; funding acquisition, Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Gusu Innovation Leading Talents Special Project (the 2020 Kunshan Zu Chong Key Tack Project), grant number ZXL2020210.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within in the article.

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

References

1. Jia, G.; Wang, J. A Review on Mission Planning Methods of UAV Cluster. *Syst. Eng. Electron.* **2021**, *43*, 99–111.
2. Zhong, W.; Li, X.; Chang, H.; Liang, F. Anti-UAV cluster air defense deployment model based on nested PSO algorithm. *Electro-Opt. Control.* **2021**, *28*, 1–7.
3. Chen, Y.; Yang, D.; Yu, J. Multi-UAV Task Assignment With Parameter and Time-Sensitive Uncertainties Using Modified Two-Part Wolf Pack Search Algorithm. *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 2853–2872. [CrossRef]
4. Wei, R.; Wu, Z. Research on Real-time Task Assignment method of UAV Cluster. *J. Syst. Simul.* **2021**, *33*, 1574–1581.
5. Alidaee, B.; Wang, H.; Landram, F. A Note on Integer Programming Formulations of the Real-Time Optimal Scheduling and Flight Path Selection of UAVs. *IEEE Trans. Control. Syst. Technol.* **2009**, *17*, 839–843. [CrossRef]
6. Lim, G.J.; Kim, S.; Cho, J.; Gong, Y.; Khodaei, A. Multi-UAV Pre-Positioning and Routing for Power Network Damage Assessment. *IEEE Trans. Smart Grid* **2018**, *9*, 3643–3651. [CrossRef]
7. Bisis, R.S.; Pal, A.; Werho, T.; Vittal, V. A Graph Theoretic Approach to Power System Vulnerability Identification. *IEEE Trans. Power Syst.* **2021**, *36*, 923–935. [CrossRef]
8. Tootooni, M.S.; Rao, P.K.; Chou, C.; Kong, Z.J. A Spectral Graph Theoretic Approach for Monitoring Multivariate Time Series Data From Complex Dynamical Processes. *IEEE Trans. Autom. Sci. Eng.* **2018**, *15*, 127–144. [CrossRef]
9. Wang, H.; Zhang, L.; Shi, C.; Che, F.; Zhang, P. Modeling of Equipment Support Task Assignment and Solution of DLS-BCIWBA Algorithm. *Syst. Eng. Electron.* **2018**, *40*, 1979–1985.
10. Almannaa, M.H.; Elhenawy, M.; Rakha, H.A. A Novel Supervised Clustering Algorithm for Transportation System Applications. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 222–232. [CrossRef]
11. Wang, Z.J.; Zhan, Z.H.; Kwong, S.; Jin, H.; Zhang, J. Adaptive Granularity Learning Distributed Particle Swarm Optimization for Large-Scale Optimization. *IEEE Trans. Cybern.* **2021**, *51*, 1175–1188. [CrossRef] [PubMed]
12. Li, L.; Chang, L.; Gu, T.; Sheng, W.; Wang, W. On the Norm of Dominant Difference for Many-Objective Particle Swarm Optimization. *IEEE Trans. Cybern.* **2021**, *51*, 2055–2067. [CrossRef] [PubMed]
13. Huanca, D.H.; Pareja, L.A.G. Chu and Beasley Genetic Algorithm to Solve the Transmission Network Expansion Planning Problem Considering Active Power Losses. *IEEE Lat. Am. Trans.* **2021**, *19*, 1967–1975. [CrossRef]
14. Pradhan, D.; Wang, S.; Ali, S.; Yue, T.; Liaaen, M. CBGA-ES+: A Cluster-Based Genetic Algorithm with Non-Dominated Elitist Selection for Supporting Multi-Objective Test Optimization. *IEEE Trans. Softw. Eng.* **2021**, *47*, 86–107. [CrossRef]

15. Davari, S.A.; Nekoukar, V.; Garcia, C.; Rodriguez, J. Online Weighting Factor Optimization by Simplified Simulated Annealing for Finite Set Predictive Control. *IEEE Trans. Ind. Inform.* **2021**, *17*, 31–40. [CrossRef]
16. Duan, H.B.; Qiao, P.X. Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning Int. *J. Intell. Comput. Cybern* **2014**, *07*, 24–37. [CrossRef]
17. Wu, H.; Zhang, F.; Wu, L. A new swarm intelligence algorithm-Wolf pack algorithm. *Syst. Eng. Electron.* **2013**, *35*, 2430–2438.
18. Wu, H.; Xiao, R. New research on swarm intelligence: Role-matching Wolf Division of Labor. *J. Intell. Syst.* **2021**, *16*, 125–133.
19. Xu, J.; Kang, X.; Fan, Z.; Zhang, Z.; Li, Y.; Dong, X.; Gao, X. Design of Highly Uniform Magnetic Field Coils With Wolf Pack Algorithm. *IEEE Sens. J.* **2021**, *21*, 4412–4424. [CrossRef]
20. Wang, D.; Ban, X.; Ji, L.; Guan, X.; Liu, K.; Qian, X. An Adaptive Shrinking Grid Search Chaotic Wolf Optimization Algorithm Using Standard Deviation Updating Amount. *Comput. Intell. Neurosci.* **2020**, *2020*, 1–15. [CrossRef]
21. Cao, Q.K.; Yang, K.W.; Ren, X.Y. Vehicle routing optimization with multiple fuzzy time windows based on improved wolf pack algorithm. *Adv. Prod. Eng. Manag.* **2017**, *12*, 401–411. [CrossRef]
22. Li, H.; Wu, H. An oppositional wolf pack algorithm for parameter identification of the chaotic systems. *Optik* **2016**, *127*, 9853–9864. [CrossRef]
23. Zhu, Y.; Jiang, W.; Kong, X.; Quan, L.; Zhang, Y. A chaos wolf optimization algorithm with self-adaptive variable step-size. *Aip Adv.* **2017**, *7*, 105024. [CrossRef]
24. Chen, X.; Tang, C.; Wang, J.; Zhang, L.; Meng, Q. Improved Wolf Pack Algorithm Based on Differential Evolution Elite Set. *IEICE Trans. Inf. Syst.* **2018**, *101*, 1946–1949. [CrossRef]
25. Wang, D.; Qian, X.; Liu, K.; Ban, X.; Guan, X. An Adaptive Distributed Size Wolf Pack Optimization Algorithm Using Strategy of Jumping for Raid(September 2018). *IEEE Access* **2018**, *6*, 65260–65274. [CrossRef]
26. Chen, X.; Cheng, F.; Liu, C.; Cheng, L.; Mao, Y. An improved Wolf pack algorithm for optimization problems: Design and evaluation. *PLoS ONE* **2021**, *16*, e0254239. [CrossRef]
27. Gao, Y.; Zhang, F.; Zhao, Y.; Li, C. Quantum-inspired wolf pack algorithm to solve the 0-1 knapsack problem. *Math. Probl. Eng.* **2018**, *2018*, 1–10. [CrossRef]
28. Zhu, Q.; Wu, H.; Li, N.; Hu, J. A Chaotic Disturbance Wolf Pack Algorithm for Solving Ultrahigh-Dimensional Complex Functions. *Complexity* **2021**, *2021*, 1–15. [CrossRef]
29. Lu, Y.; Ma, Y.; Wang, J. Multi-Population Parallel Wolf Pack Algorithm for Task Assignment of UAV Swarm. *Appl. Sci.* **2021**, *11*, 11996. [CrossRef]
30. Lu, Y.; Ma, Y.; Wang, J.; Han, L. Task assignment of UAV swarm based on Wolf Pack algorithm. *Appl. Sci.* **2020**, *10*, 8335. [CrossRef]
31. YongBo, C.; YueSong, M.; JianQiao, Y.; XiaoLong, S.; Nuo, X. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing* **2017**, *266*, 445–457. [CrossRef]
32. Wang, F.; Zhang, H.; Han, M.; Xing, L. Co-evolution Based Mixed-variable Multi-objective Particle Swarm Optimization for UAV Cooperative Multi-task Allocation Problem. *Chin. J. Comput.* **2021**, *44*, 1967–1983.
33. Gao, S.; Zhou, M.; Wang, Y.; Cheng, J.; Yachi, H.; Wang, J. Dendritic Neuron Model With Effective Learning Algorithms for Classification, Approximation, and Prediction. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 601–614. [CrossRef]
34. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. MHS'95. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
35. Grefenstette, J.J. Optimization of Control Parameters for Genetic Algorithms. *IEEE Trans. Syst. Man Cybern.* **1986**, *16*, 122–128. [CrossRef]
36. Gao, S.; Yu, Y.; Wang, Y.; Wang, J.; Cheng, J.; Zhou, M. Chaotic Local Search-Based Differential Evolution Algorithms for Optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 3954–3967. [CrossRef]
37. Kumar, A.; Misra, R.K.; Singh, D. Improving the local search capability of Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat Phase. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1835–1842.