applied sciences

MDPI

Article

Task Assignment of UAV Swarm Based on Wolf Pack Algorithm

Yingtong Lu, Yaofei Ma *, Jiangyun Wang and Liang Han

School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; luyingtong@buaa.edu.cn (Y.L.); wangjiangyun@buaa.edu.cn (J.W.); hanliang@buaa.edu.cn (L.H.)
* Correspondence: ma_yaofei@buaa.edu.cn

Received: 5 November 2020; Accepted: 21 November 2020; Published: 24 November 2020

Abstract: To perform air missions with an unmanned aerial vehicle (UAV) swarm is a significant trend in warfare. The task assignment among the UAV swarm is one of the key issues in such missions. This paper proposes PSO-GA-DWPA (discrete wolf pack algorithm with the principles of particle swarm optimization and genetic algorithm) to solve the task assignment of a UAV swarm with fast convergence speed. The PSO-GA-DWPA is confirmed with three different ground-attack scenarios by experiments. The comparative results show that the improved algorithm not only converges faster than the original WPA and PSO, but it also exhibits excellent search quality in high-dimensional space.

Keywords: UAV swarm; task assignment; Wolf Pack Algorithm (WPA); PSO-GA-DWPA; ground-attack

1. Introduction

In the last two decades, how to use unmanned aerial vehicles (UAVs) for various military missions has received growing attention. Over the years, UAVs have been used in military missions, such as ground attack missions (GAM), wide area search missions (WASM), suppression of enemy air defense (SEAD), and combat intelligence, surveillance, and reconnaissance (ISR).

At present, research on UAVs has changed from focus on a large full-function UAV to a swarm of small single-function stealth UAVs, because the cost of a small single-function UAV is much lower than that of a large full-function UAV. The basic performance of UAV swarm operations is to meet the requirement of fast reliable assignment and execution, which is very complex to implement because of the large amount of computation.

The task assignment of a UAV swarm is usually viewed as an optimization problem, in which the optimal solution is found by maximizing or minimizing an objective function with constraints. The classical optimization models include a mixed integer linear programming (MILP) [1] model, the traveling salesman problem (TSP) [2] model, the generalized assignment problem (GAP) [3] model, the vehicle routing problem (VRP) [4] model and the cooperative multiple task assignment problem (CMTAP) [5] model.

The methods to solve optimization models are divided into two categories. One category is to get an exact solution by mathematical programming, such as the Hungarian algorithm [6], the branch and bound search algorithm [7], dynamic programming [8], the exhaustive method [9], Newton's method [10] and the gradient method [11]. These methods can guarantee an optimal solution of a convex problem if there is a solution, but it is difficult to solve an NP-hard non-convex problem. Moreover, this kind of method is normally suitable for low-dimensional space.

Another category is evolutionary algorithms, which are inspired by the mechanism of nature's evolution of a biological population. These methods usually do not have polynomial time complexity, but can approach a global optimal or suboptimal solution with finite calculation cost. In the task

assignment of the UAV swarm, it is difficult to establish an accurate mathematical model and guarantee that the model is convex. Therefore, the evolutionary algorithms are the most common methods to solve this kind of problem. The classical evolutionary algorithms include the simulated annealing algorithm (SAA) [12], the tabu search algorithm (TS) [13], the genetic algorithm (GA) [14], the ant colony optimization algorithm (ACO) [15], and particle swarm optimization algorithm (PSO) [16].

Considering the UAV swarm operation in the future, task assignment is challenging due to its computational complexity caused by two reasons. The first is the complexity of the task assignment model, including determining the order of tasks, the constraints and the coupling between task assignment and trajectory optimization. The second is the large amount of computation and long solution time caused by the dimension of the variables. For example, in a scenario where 80 aircraft attack 100 targets, assuming that only a single attack task is performed and each target is attacked only once, the variables describing the UAVs assigned to all targets, reach 100 dimensions, which greatly increases the computation. Jia et al. [17] and Su [18] improved the variable's coding method where the constraint information was added so as to overcome the model complexity caused by constraints. The co-evolutionary ant colony algorithm was presented in [19] to decompose a high-dimensional variable into several low-dimensional variables. This method can effectively improve the speed of solving the algorithm itself, but it is necessary to fuse all UAVs' decisions to meet the constraints of task assignment, such as the order of tasks. In [20], by integrating PSO with the beetle antennae search (BAS) algorithm, which can overcome the defect that PSO is prone to fall into local convergence, the convergence speed and solution accuracy of the algorithm are improved. Both in [18,20], the fixed parameters of PSO are improved to be linear or dynamic so as to effectively accelerate the convergence speed. Braun [21] developed the island model, where several populations isolated on an island are optimized by GA until they degenerate. Then the degeneration is removed by refreshing the population on each island through individuals of other islands. This method effectively solves the problem of population degradation. To mitigate the parameter tuning on high-dimensional problems, which is a computationally expensive procedure, Tuani et al. [22] introduced an adaptive approach to a heterogeneous ant colony population that evolves the alpha and beta controlling parameters for ACO.

In addition, distributed task assignment methods were proposed to improve the computation efficiency to overcome the problem of large computation and long solution time. Matin et al. [23] proposed minimum distance greedy search (MDGS) to make the decision for a UAV to select the targets, when the arrivals are dynamic and appear uniformly in a known rectangular region. This approach is currently very effective in the field of dynamic task assignment for individual UAVs. Di et al. [24] used the distributed auction algorithm, in which each UAV decides their own task goals according to their own local information, to reduce the computational load of the system to a certain extent. Similarly, a market-based decentralized algorithm was proposed by Oh et al. [25] to realize online task assignment. Zhang et al. [26] used a contract network to rapidly update the task assignment scheme after pre-assignment based on PSO. Such methods did not necessarily find the optimal solution based on the global information, so the optimal assignment of resources cannot be realized.

Despite the great progress in improving efficiency, dimension explosion is still the primary concern of these methods. High-dimension combinatorial optimization is still a difficult task assignment problem. One possible direction is to establish a hybrid method to overcome the defects of each individual method, such as PSO-SA [27], PSO-GA [28,29], and DPSO-GT [30,31]. Simulation experiments have demonstrated the effectiveness of these methods, when there are under 30 dimensions. The other direction is to design a more efficient algorithm, especially for high- dimension problems. Wu et al. [32] proposed a new heuristic swarm intelligent algorithm named the wolf pack algorithm (WPA), based on a swarm of intelligent wolves. Simulation results showed that the WPA is especially suitable for solving high-dimension and multimodal function optimization problems. Subsequently, the algorithm was applied to some classical optimization problems and achieved good results [33,34]. However, whether the WPA with excellent performance in solving high-dimensional

continuous problems is suitable for discrete optimization problems such as task assignment needs further verification.

The main contribution of this paper is the application of WPA in high-dimension task assignment and the improvement on the WPA with the ideas of PSO and a GA to accelerate convergence speed. The remainder of this paper is organized as follows. Section 2 formalizes the UAV swarm task assignment problem with respect to performance requirements. The principles of the WPA and the improvement of four aspects of the WPA are described in Section 3. In Section 4, the PSO-GA-DWPA is compared with the WPA and PSO by simulation experiments in three task scenarios. Finally, Section 5 concludes the paper and briefly explores future work.

2. Task Assignment Model

In this section, a background incorporating a UAV swarm that executes an attack mission in a large field is presented. Then, a mathematical formulation is established.

There are N_T static targets whose initial positions have been revealed, and N_V attacking UAVs planning to attack these targets with the minimum cost.

In actual task assignment for ground attack, the task assignment model is very complex; it includes many factors, such as fuel constraints, ammunition quantity, timing constraints between multiple tasks, UAV mobility, threats, and redundancy tolerance. The factors considered in different combat scenarios are different, so the task assignment model varies with the actual situation. This paper considers a more general task assignment model which is not limited to a specific scenario and a specific UAV type. The point is to verify whether the proposed PSO-GA-DWPA is suitable for solving high-dimensional discrete optimization problems.

To avoid the influence of uncertain factors on the analysis of simulation results, the following ideal assumptions are set up to simplify the formalization of the problem.

1. There are no obstacles and no-fly zones in the task scenario. A flight trajectory can be described as connections of straight lines.
2. It does not take into account the consumption of time spent preparing and firing the weapon. In other words, only the time the UAV swarm takes to reach the target positions is considered.
3. The UAV swarm maintains the same constant velocity and hence the flight time can be represented with the flight distance.
4. Each target can be attacked only once. There are more targets than UAVs in the swarm, which means each UAV will probably be assigned multiple targets.

The targets can be expressed as $T = \left\{T_1, T_2, \cdots, T_{N_T}\right\}$, and the attacking UAV swarm can be expressed as $V = \left\{V_1, V_2, \cdots, V_{N_V}\right\}$.

The task assignment plan of UAV $V_i \in V$ is

$$\begin{gathered} plan_i = \left\{stage_1^{(i)}, stage_2^{(i)}, \cdots, stage_{N_{mi}}^{(i)}\right\} \\ stage_k^{(i)} = (T_j, L_k^{(i)}), T_j \in T \end{gathered}, \tag{1}$$

where $stage_k^{(i)}$ is the stage k of UAV $V_i \in V$, N_{mi} is the task number assigned to UAV $V_i \in V$, T_j is the target of the UAV $V_i \in V$ at stage k, and $L_k^{(i)}$ is the distance to T_j of UAV $V_i \in V$ at stage k.

Without loss of generality, two cost indicators that must be considered in task assignment are adopted: the shortest total range of all UAVs (i.e., the lowest fuel cost) and the shortest time to complete all tasks.

1. Total range of all UAVs

$$J_1 = \sum_{i=1}^{N_V} \sum_{k=1}^{N_{mi}} L_k^{(i)}. \tag{2}$$

For all UAVs in the swarm, the average range is

$$J_1^* = \frac{\sum_{i=1}^{N_V} \sum_{k=1}^{N_{mi}} L_k^{(i)}}{N_V}. \tag{3}$$

2. Time to complete all tasks

$$J_2 = \max_{i \in V} Time_i, \tag{4}$$

where $Time_i$ is the time of UAV $V_i \in V$ to finish all of its tasks. With consideration of the same constant velocity for all UAVs, the time of UAV $V_i \in V$ to finish all of its tasks can be expressed as its total range. Therefore, Equation (4) can be represented as Equation (5).

$$J_2^* = \max_{i \in V} \sum_{k=1}^{N_{mi}} L_k^{(i)}. \tag{5}$$

The optimization goal is to minimize both the total range of all UAVs and the maximum range among all UAVs.

Thus, the cost function of task assignment is

$$\min J = \omega_1 J_1^* + \omega_2 J_2^* = \omega_1 \frac{\sum_{i=1}^{N_V} \sum_{k=1}^{N_{mi}} L_k^{(i)}}{N_V} + \omega_2 \max_{i \in V} \sum_{k=1}^{N_{mi}} L_k^{(i)}, \tag{6}$$

where ω_1 and ω_2 are weighting factors reflecting the importance of each performance criterion, decided by the commander, and $\omega_1 + \omega_2 = 1$

3. The Wolf Pack Algorithm

3.1. The Basics of the Wolf Pack Algorithm

The WPA adopts the bottom-up design principle and simulates wolves hunting cooperatively according to their division system of responsibilities as shown in Figure 1. Wolves are divided into a leader wolf, exploring wolves and fierce wolves, and they share information with each other. The leader wolf is closest to the prey because it is most sensitive to smell, so it is the guidance of the wolf pack. Exploring wolves are responsible for detecting the environment to find the position of the prey, due to their sensitivity to smell. Fierce wolves are responsible for getting close to catch the prey quickly.

The problem space is defined as an $N \times D$ Euclidean space, where N is the number of wolves in the wolf pack, and D is the number of each wolf's information dimension. The position of the wolf $i \in N$ is expressed as $X_i = \{x_{i1}, \cdots x_{id}, \cdots, x_{iD}\}$, where x_{id} is the value in $d^{th}(d \in D)$ dimension of wolf $i \in N$, the smell concentration of prey perceived by wolf is described as the cost function value. The distance between the wolf p and q is defined as the Manhattan distance.

$$L(p,q) = \sum_{d=1}^{D} \left| x_{pd} - x_{qd} \right|. \tag{7}$$

WPA is shown in Algorithm 1. The whole process of wolf pack hunting can be abstracted into 3 features:

- the selection of the leader wolf based on the winner-take-all rule;
- three cooperative behaviors including walking, calling and sieging;
- an update mechanism based on the strongest-survives law.

Algorithm 1: WPA

Data: initial position of UAVs:pos_V, position of targets:pos_T,iterations, population size:N, $step_a$, $step_b$,$step_c$, T_{max}, d_{near}, α, β
Result: solution X_{leader}, min cost J
Generate initial solution X;
for *iterations* **do**
  Select the leader wolf;
  /* walking behaviour: */
  Select S exploring wolves;
  while *walking times* $< T_{max}$ **do**
    for *exploring wolves* **do**
      walk to h directions as *Eq.*(8);
      if $J(X_{new}) < J(X)$ **then**
        update position:$X \leftarrow X_{new}$
      end
      if $J(X) < J(X_{leader})$ **then**
        replace leader wolf:$X_{leader} \leftarrow X$;
        break;
      end
    end
  end
  /* calling behaviour: */
  Select M fierce wolves;
  while *distance between fierce wolf and leader* $< d_{near}$ **do**
    for *fierce wolves* **do**
      get close to the leader wolf as *Eq.*(9) **if** $J(X_{new}) < J(X)$ **then**
        update position:$X \leftarrow X_{new}$;
        update distance between fierce wolf and leader as *Eq.*(7)
      end
      if $J(X) < J(X_{leader})$ **then**
        replace leader wolf:$X_{leader} \leftarrow X$;
        break;
      end
    end
  end
  /* sieging behaviour: */
  for *all wolves except the leader wolf* **do**
    get close to the leader wolf as *Eq.*(11);
    if $J(X_{new}) < J(X)$ **then**
      update position:$X \leftarrow X_{new}$
    end
  end
  /* eliminate and generate: */
  Select R weakest wolves;
  delete the R wolves;
  generate randomly R new wolves;
end

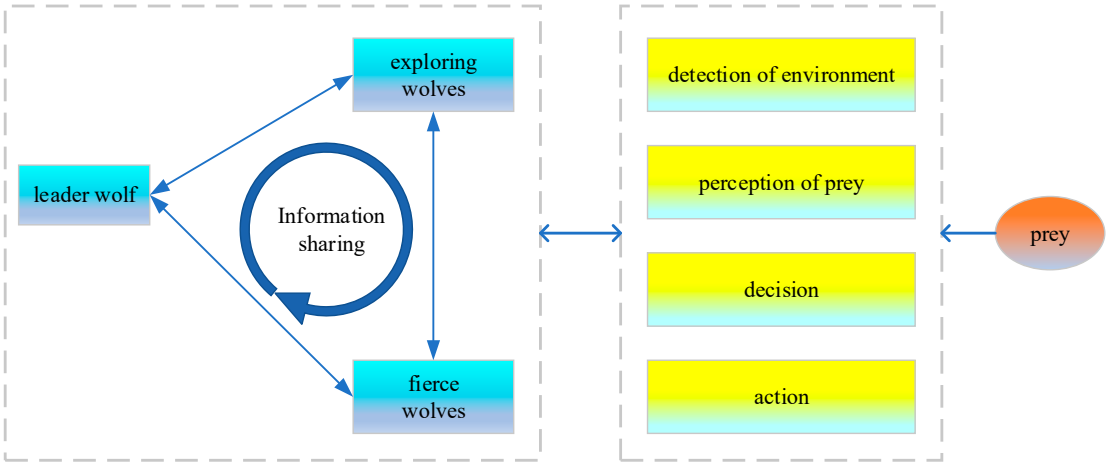


Figure 1. Hunting process of the wolf pack.

3.1.1. The Selection of Leader Wolf Based on Winner-Take-All Rule

The wolf with the minimum cost can be the leader wolf at any time. Initially, the wolf with the minimum cost is chosen as the leader wolf. In the subsequent stages, if the cost of some wolf is less than that of the current leader wolf, that wolf will be the new leader. If multiple wolves are the best at the same time, pick one at random to be the leader wolf. The leader wolf goes directly to the next iteration without executing the following cooperative behaviors, until it is replaced by a new leader wolf.

3.1.2. Three Cooperative Behaviors

1. Walking behavior. S suboptimal wolves are selected as exploring wolves to perform the walking behavior. S is a random integer picked from $\left[\frac{N}{\alpha+1}, \frac{N}{\alpha}\right]$, where α is the scale factor of exploring wolves. Starting from the current position, exploring wolf $i \in N$ makes one step forward towards h directions. On account of individual differences, h generally takes a random integer within a limited range. The new position to the $b^{th}(b \in h)$ direction is obtained by

$$X_i^b = X_i + \sin(2\pi \times \frac{b}{h}) \times step_a, \tag{8}$$

where $step_a$ is the length of walking step. Exploring wolf $i \in N$ will return to its initial position after detecting each direction, then choose the best one to update its position. All exploring wolves will keep walking until the satisfy one of the following conditions.

- As long as one exploring wolf's new position is better than that of the leader wolf, this exploring wolf will be the new leader wolf and the wolf pack will move to the calling behavior.
- When walking times reach the maximum $T_{\max}$, the wolf pack move to the calling behavior.

2. Calling behavior. The leader wolf calls for $M(M = N - S - 1)$ nearest wolves as fierce wolves to get close to its position rapidly with a large step. The new position of fierce wolf $i \in N$ is obtained by

$$X_i^{k+1} = X_i^k + step_b \times \frac{X_{leader}^k - X_i^k}{\left|X_{leader}^k - X_i^k\right|}, \tag{9}$$

where $step_b$ is the length of raid step. Fierce wolves will continue to get close to the leader wolf until one of the following conditions is satisfied.

- As long as one fierce wolf's new position is better than that of the leader wolf, this fierce wolf will be the new leader and the wolf pack moves to the sieging behavior.

- The Manhattan distance between a fierce wolf and the leader wolf is less than the threshold distance d_{near}. d_{near} is estimated according to

$$d_{near} = \frac{1}{D \times \omega} \times \sum_{d=1}^{D} |\max_d - \min_d|, \tag{10}$$

where ω is the factor representing the threshold distance and $[\min_d, \max_d]$ is the variable's domain in the d^{th} dimension.

3. Sieging behavior. The leader wolf, whose position is treated as the prey's position, guides all other wolves to siege the prey with a small step. For iteration k, the position of the prey is X^k_{leader}; then the new position of wolf $i \in N$ is updated according to

$$X^{k+1}_i = X^k_i + \lambda \cdot step_c \cdot \left| X^k_{leader} - X^k_i \right|, \tag{11}$$

where λ is a random real number in $[-1, 1]$, and $step_c$ is the length of the siege step. For any wolf, if its new position is better than the current position, its position will be updated, otherwise its current position will be kept. The wolf whose position is the best will be chosen as the leader wolf.

3.1.3. Update Mechanism Based on Strong-Survive Law

Prey is distributed according to the principle of strong to weak, which will cause the weakest wolves to starve without food, that is to say, the wolves too far away from the prey will be eliminated. In the algorithm, the R weakest wolves will be eliminated from the population and extra new R wolves will be generated randomly. The larger R is, the more wolves will be generated, which is conducive to maintaining the diversity of individuals in the population. However, if R is too large, the algorithm tends to perform a random search. On the contrary, if R is too small, it is not conducive to maintaining the diversity of individuals in the population, and the ability of the algorithm to open up a new solution space is weakened. Since the size and number of prey are different with each capture, the number of wolves starving to death varies. R is a random integer in $\left[\frac{N}{2\beta}, \frac{N}{\beta}\right]$, where β is the scale factor of the wolf population update.

3.2. The Proposed PSO-GA-DWPA

The WPA is suitable for continuous problems where the variable in each dimension changes continuously. However, the task assignment problem is a discrete problem in which the variable in each dimension is an integer belonging to a set. As such, based on the rules of the WPA, it is necessary to improve the WPA to match the integer discrete character of the task assignment. In addition, in order to improve the convergence speed and solution accuracy, PSO and a GA are introduced into WPA. The details of the PSO-GA-DWPA are as follows and the algorithm is shown in Algorithm 2.

3.2.1. Integer Matrix Coding

Integer matrix coding is adopted to express the assignment schema

$$X_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{iN_T} \\ y_{i1} & y_{i2} & \cdots & y_{ij} & \cdots & y_{iN_T} \end{bmatrix}, \tag{12}$$

where X_i is the position of the wolf $i \in N$, N is the population number of wolves, the dimensions of variable is N_T because there are more targets than UAVs and each target can only be attacked once. The 1st row of Equation (12) is the index of the UAVs which perform attacking tasks where $x_{ij} \in V$, and elements in this row can be repeated because a UAV can attack multiple targets. The 2ed row of Equation (12) is the index of targets, where $y_{ij} \in T$. For the same UAV, the order in which the target number appears is the attack order. For example, if there are 5 UAVs attacking 8 targets

and the coding matrix is shown as $X_i = \begin{bmatrix} 1 & 3 & 2 & 2 & 1 & 1 & 3 & 3 \\ 3 & 5 & 1 & 4 & 2 & 8 & 7 & 6 \end{bmatrix}$, then the task assignment scheme is as follows: $plan_1 = \left\{(3, L_1^{(1)}), (2, L_2^{(1)}), (8, L_2^{(1)})\right\}$, $plan_2 = \left\{(1, L_1^{(2)}), (4, L_2^{(2)})\right\}$, and $plan_3 = \left\{(5, L_1^{(3)}), (7, L_2^{(3)}), (6, L_2^{(3)})\right\}$.

3.2.2. Improvement on Walking Behavior

The walking behavior of wolves is essentially an active exploration of the unknown environment. This process determines the extent of the current best solution approaches to the global optimal solution. In the WPA, each exploring wolf scouts the prey from h directions divided equally by 360 degrees as Equation (8), 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. To scout the prey more effectively, PSO is introduced in this paper to make the walking process completed from as few directions as possible, under the guidance of the global extremum and the individual extremum.

Each exploring wolf is represented as one particle in the PSO. The implementation of PSO is divided into three steps [20]: tracking individual extremum, tracking global extremum and individual variation.

1. Tracking individual extremum

This part of the formula is expressed in the discrete domain as Equation (13). This represents a copy operation with probability c_1. X_i^k represents the position of wolf $i \in S$ in iteration k. P_i^k is the optimal solution of wolf $i \in S$ according to the current scouted directions, representing the individual extremum of wolf $i \in S$ in iteration k. Note that φ_i^k is a temporary variable.

$$\varphi_i^k = c_1 \otimes F_1(X_i^k, P_i^k) = \begin{cases} F_1(X_i^k, P_i^k), & \varepsilon = rand() < c_1 \\ X_i^k, & otherwise \end{cases}. \tag{13}$$

During the operation in Figure 2, a random number $\varepsilon \in [0, 1]$ is generated firstly. If $\varepsilon < c_1$, where c_1 is the threshold to determine crossover, two random integers $a, b \in [1, N_T]$ are generated. The columns between the index range $[a, b]$ of X_i^k will be exchanged with the counterpart in the individual extremum P_i^k. The result of this operation is φ_i^k. On the other hand, if $\varepsilon \geq c_1$, $\varphi_i^k = X_i^k$.

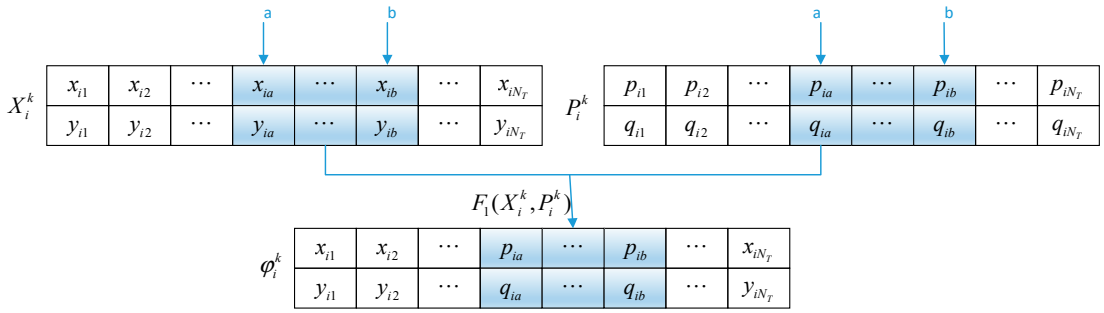


Figure 2. Tracking individual extremum.

2. Tracking global extremum

Equation (14) represents a copy operation with probability c_2. P_g^k is the position of the leader wolf, representing the global extremum in iteration k. λ_i^k is a temporary variable.

$$\lambda_i^k = c_2 \otimes F_1(\varphi_i^k, P_g^k) = \begin{cases} F_1(\varphi_i^k, P_g^k), & \varepsilon = rand() < c_2 \\ \varphi_i^k, & otherwise \end{cases}. \tag{14}$$

During the operation in Figure 3, a random number $\varepsilon \in [0,1]$ is generated firstly. If $\varepsilon < c_2$, where c_2 is the threshold to determine crossover, two random integers $a, b \in [1, N_T]$ are generated. The columns between the index range $[a,b]$ of φ_i^k will be exchanged with the counterpart in the global extremum P_g^k. The result of this operation is λ_i^k. On the other hand, if $\varepsilon \geq c_2$, $\lambda_i^k = \varphi_i^k$.

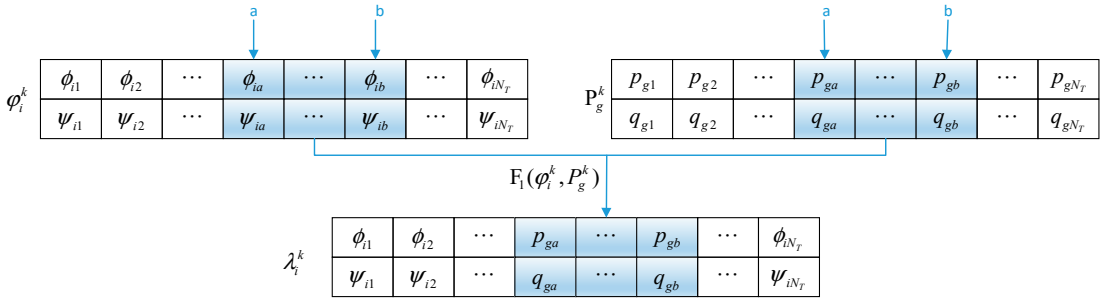


Figure 3. Tracking global extremum.

3. Individual variation

This part of the formula is described in the discrete domain as Equation (15), where $step_a$ is the walking step, which determines how many columns of λ_i^k will mutate. W_i^k is a temporary variable representing the result of the walking behavior.

$$W_i^k = F_2(\lambda_i^k, \Gamma(step_a)). \tag{15}$$

During the operation in Figure 4, function $\Gamma(step_a)$ randomly selects $step_a$ columns in the λ_i^k. The values of the 1^{st} row in these columns are randomly selected from $[1, N_V]$, while the values of the second row are randomly exchanged.

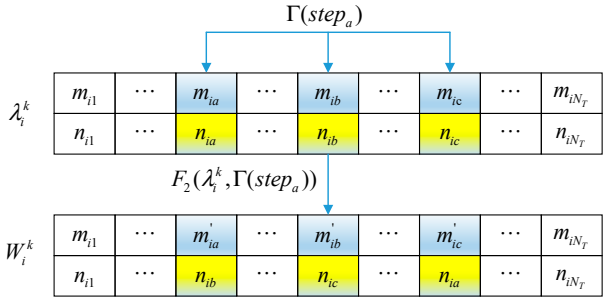


Figure 4. Individual variation.

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. Inspired by gene segment duplication in the GA, each fierce wolf duplicates a piece of the leader's position to replace its own to realize the fast approach of the leader wolf.

This part of the formula is expressed in the discrete domain as Equation (16). P_g^k is the position of the leader wolf, representing the global extremum in iteration k, $step_b$ is the raid step, and C_i^k ($i \in M$) is a temporary variable representing the result of the calling behavior.

$$C_i^k = F_1(X_i^k, P_g^k, step_b). \tag{16}$$

During the operation in Figure 5, a continuous segment of length $step_b$ is randomly intercepted from the wolf $i \in M$, then, is replaced by the corresponding segment of the leader wolf.

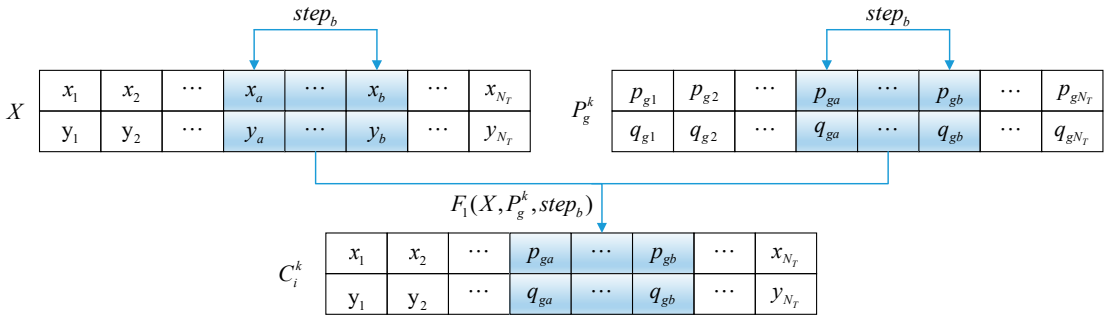


Figure 5. Duplicating the leader wolf's position segment in the calling behavior.

3.2.4. Improvement on Sieging Behavior

This part of the formula is expressed in the discrete domain as Equation (17), where $step_c$ is siege step, and X_i^{k+1} is the result of the siege behavior.

$$X_i^{k+1} = F_2(X, \Gamma(step_c)) = \begin{cases} F_2(W_i^k, \Gamma(step_c)), & i \in S \\ F_2(C_i^k, \Gamma(step_c)), & i \in M \end{cases}. \tag{17}$$

During the operation in Figure 6, Function $\Gamma(step_c)$ randomly selects $step_c$ columns in X. The values of the first row in these columns are randomly selected from $[1, N_V]$, while the values of the second row are randomly exchanged.

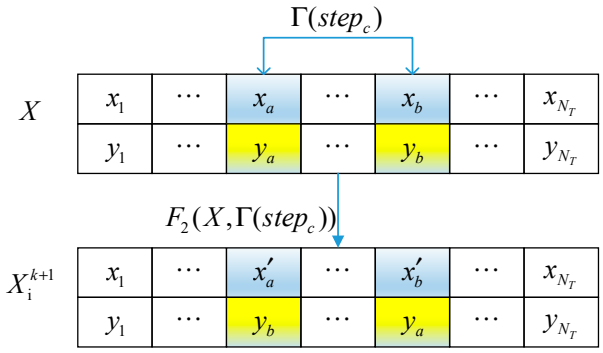


Figure 6. Individual variation in siege behavior.

3.2.5. Improvement in Wolf Population Update

In WPA, R weakest wolves are eliminated and replaced by new wolves generated randomly just as the initialization. Experiments show that new wolves generated by this method do not have competitiveness, and they are finally eliminated and meaningless in the algorithm. In the PSO-GA-DWPA, a new wolf is generated by using the method of small variation of the leader wolf. The process is similar to individual variation in walking behavior.

This is expressed in the discrete domain as Equation (18), where $step_d$ is variation step, and X_i^{k+1} is the position of new wolf.

$$X_i^{k+1} = F_2(P_g^k, \Gamma(step_d)). \tag{18}$$

During the operation in Figure 7, Function $\Gamma(step_d)$ randomly selects $step_d$ columns in P_g^k. The values of the first row in these columns are randomly selected from $[1, N_V]$, while the values of the second row are randomly exchanged.

Algorithm 2: PSO-GA-DWPA

Data: initial position of UAVs:pos_V, position of targets:pos_T,iterations, population size:N, $step_a$, $step_b$,$step_c$, $step_d$, T_{max}, d_{near}, α, β
Result: solution X_{leader}, min cost J
Generate initial solution X;
for *iterations* **do**
  Select the leader wolf;
  `/* walking behaviour:` */
  Select S exploring wolves;
  while *walking times* $< T_{max}$ **do**
    for *exploring wolves* **do**
      tracking individual extremum as *Eq.*(13);
      tracking global extremum as *Eq.*(14);
      individual variation as *Eq.*(15);
      if $J(X_{new}) < J(X)$ **then**
        update position:$X \leftarrow X_{new}$
      end
      if $J(X) < J(X_{leader})$ **then**
        replace leader wolf:$X_{leader} \leftarrow X$;
        break;
      end
    end
  end
  `/* calling behaviour:` */
  Select M fierce wolves;
  while *distance between fierce wolf and leader* $< d_{near}$ **do**
    for *fierce wolves* **do**
      get close to the leader wolf as *Eq.*(16) **if** $J(X_{new}) < J(X)$ **then**
        update position:$X \leftarrow X_{new}$;
        update distance between fierce wolf and leader as *Eq.*(7)
      end
      if $J(X) < J(X_{leader})$ **then**
        replace leader wolf:$X_{leader} \leftarrow X$;
        break;
      end
    end
  end
  `/* sieging behaviour:` */
  for *all wolves except the leader wolf* **do**
    get close to the leader wolf as *Eq.*(17);
    if $J(X_{new}) < J(X)$ **then**
      update position:$X \leftarrow X_{new}$
    end
  end
  `/* eliminate and generate:` */
  Select R weakest wolves;
  delete the R wolves;
  for R *wolves* **do**
    $X \leftarrow X_{leader}$;
    individual variation as *Eq.*(18):$X \leftarrow X_{new}$
  end
end

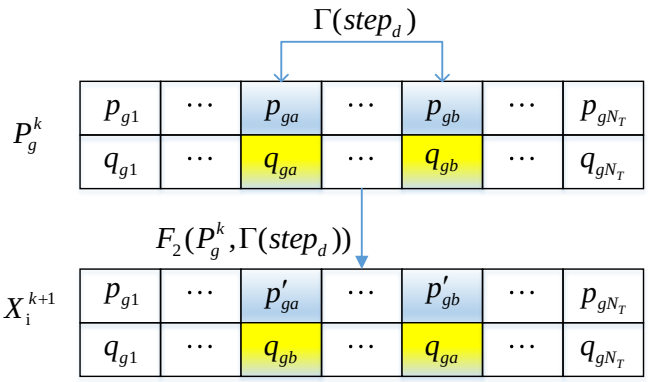


Figure 7. Individual variation in the update stage.

In this way, the new wolf can not only be competitive, but can also conduct local optimization at the position of the leader wolf to avoid local convergence.

4. Experiments of Task Assignment for UAV Swarm Using PSO-GA-DWPA

The performance of the PSO-GA-DWPA is analyzed in this section using simulation according to the assumptions in Section 2. By setting three scenarios with different numbers of UAVs and targets, we verified the applicable scope and efficiency of the algorithm. The parameters used for the simulation and the PSO-GA-DWPA are summarized in Table 1.

Table 1. Simulation parameters.

Parameters	Scenario 1	Scenario 2	Scenario 3
iterations	200	400	1000
population size	100	100	100
$step_a$	2	2	2
$step_b$	4	14	70
$step_c$	1	2	2
$step_d$	2	2	2
$T_{\max}$	10	10	10
d_{near}	2	14	70
α	4	4	4
β	5	5	5

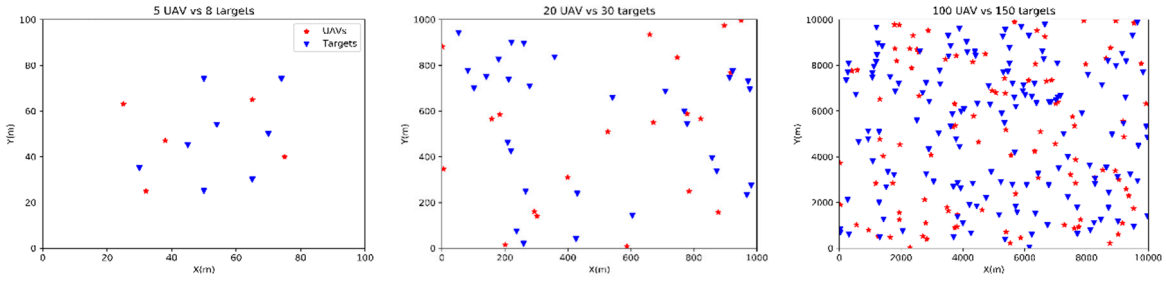
4.1. Monte Carlo Simulation in Different Scenarios

A Monte Carlo study, consisting of 50 runs, is used in this section to compare the performance of the PSO-GA-DWPA, WPA and PSO algorithms with respect to the cost function of Equation (6) where $\omega_1 = \omega_2 = 0.5$. Three scenarios are examined: 5 UAVs on 8 targets, 20 UAVs on 30 targets and 100 UAVs on 150 targets. The initial locations of the UAVs and targets were generated randomly as shown in Figure 8.

The algorithms are coded in Python, and all simulations are run on a computer with a 2.50-GHz Intel Core i5 CPU and 4 GB of RAM.

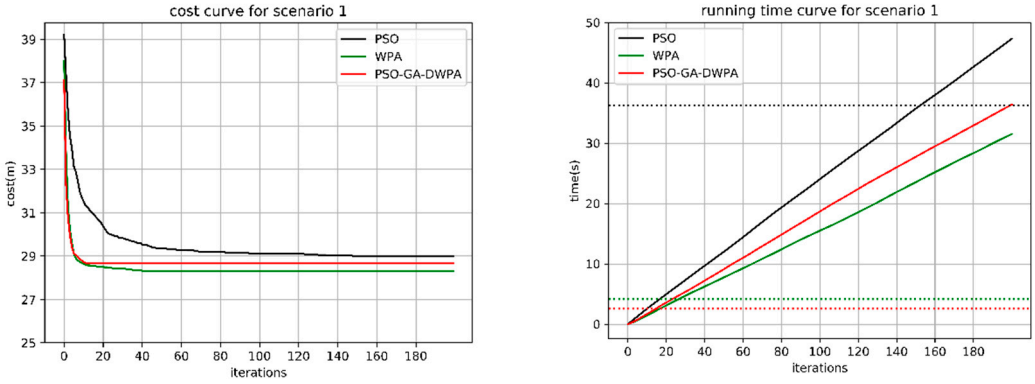
First, a small size scenario of 5 UAVs on 8 targets is analyzed. In Figures 9 and 10, the cost curve and the running time curve (the intersection of the solid line and dotted line indicates the convergence time) are plotted to compare the convergence performances of the three methods. Figure 9 shows that the optimization capabilities of the three algorithms are similar, but the convergence speed of the WPA and the PSO-GA-DWPA is better than that of PSO. Also, PSO is time-consuming and WPA and PSO-GA-DWPA is time-saving. Since the calculation process of the PSO-GA-DWPA is more complicated than that of the WPA, running time for each iteration of the PSO-GA-DWPA is slightly

longer than that of the WPA. However, the PSO-GA-DWPA needs less number of iterations to converge, so the convergence time is shorter than that of the WPA.



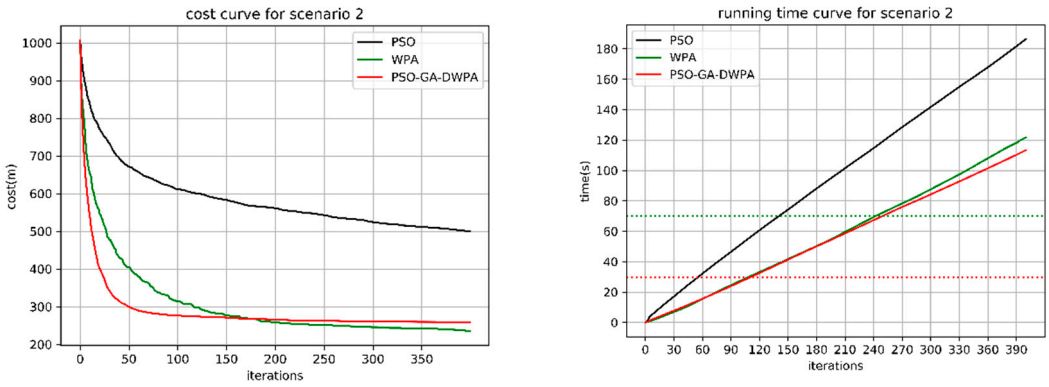
(**a**) (**b**) (**c**)

Figure 8. Initial positions of the UAV swarm and targets. (**a**) Scenario 1: 5 UAVs and 8 targets are randomly distributed in a space of 100 × 100 m. (**b**) Scenario 2: 20 UAVs and 30 targets are randomly distributed in a space of 1000 × 1000 m. (**c**) Scenario 3: 100 UAVs and 150 targets are randomly distributed in a space of 10,000 × 10,000 m.



(**a**) (**b**)

Figure 9. Simulation results in scenario 1: (**a**) the mean cost of the Monte Carlo runs, and (**b**) the mean convergence time of the Monte Carlo runs.



(**a**) (**b**)

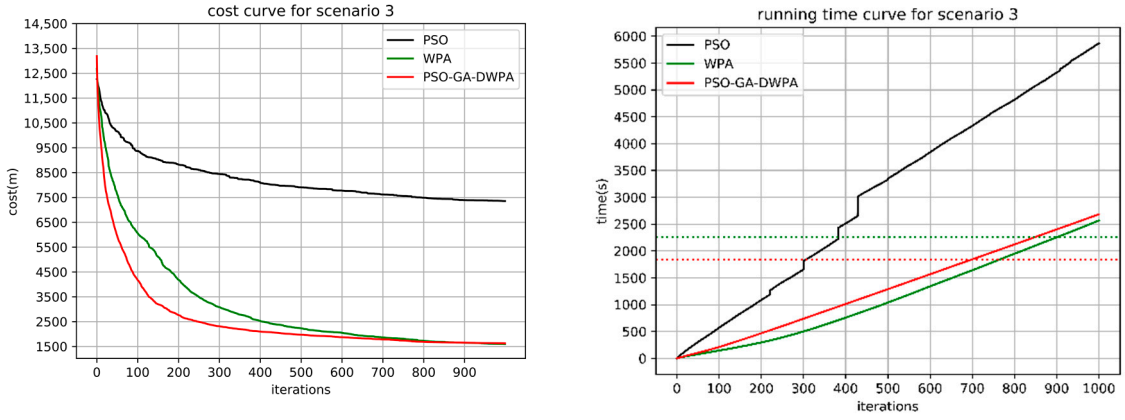
Figure 10. Simulation results in Scenario 2: (**a**) mean cost of the Monte Carlo runs, and (**b**) the mean convergence time of the Monte Carlo runs.

Compared with the recognized fast and accurate PSO algorithm, the advantage of the WPA and the proposed PSO-GA-WPA in a low-dimensional solution space (such as the 8-dimensional solution space in scenario 1), is not obvious. The experiments in Scenario 1 prove the availability of the WPA and the PSO-GA-WPA for task assignment problem. The higher the dimension of the solution space is,

the advantage of the proposed algorithm is more obvious. Compared with the WPA, the PSO-GA-WPA has the advantage of a faster convergence speed. Although their convergence costs are similar, the PSO-GA-WPA can obtain a convergent solution in a shorter time.

Second, we set the size as 20 UAVs on 30 targets, in other words, the dimension of the optimization problem is 30. As shown in Figure 10, it is obvious that the optimization performances of the WPA and the PSO-GA-DWPA are much better than that of the PSO. It can be seen that the WPA and the PSO-GA-DWPA can obtain a better solution in a shorter time. PSO-GA-DWPA has the same optimization performance as WPA, however, its convergence speed is better than that of the WPA.

Finally, the dimension of the optimization problem is set to be 150, that is, the scenario is 100 UAVs on 150 targets. The convergence performance and running time of different measures are shown in Figure 11. With the increase of dimension, PSO has become ineffective, (i) the optimal solution cannot be obtained; and (ii) the calculation speed is too slow, and the time required for 1000 iterations is nearly 2.5 times that of the other two algorithms. Running time for each iteration of the PSO-GA-DWPA is slightly longer than that of the WPA, but its convergence speed in the early stage is obviously faster, so its convergence time is shorter than that of WPA.



(**a**) (**b**)

Figure 11. Simulation results in scenario 3: (**a**) the mean cost of the Monte Carlo runs, and (**b**) the mean convergence time of the Monte Carlo runs.

The comparison of the three methods in 3 scenarios is shown in Table 2, which indicates visually that the overall performance of the PSO-GA-DWPA is better than that of the other two methods. According to Figures 9–11 and Table 2, compared with the PSO, which is recognized as fast and accurate, the advantage of the WPA and the proposed PSO-GA-WPA in the low-dimensional solution space (such as the 8-dimensional solution space in Scenario 1) is not obvious. The experiments in Scenario 1 prove the availability of the WPA and the PSO-GA-WPA for task assignment problem. The higher the dimension of the solution space is, the advantage of the proposed algorithm is more obvious. Compared with the WPA, the PSO-GA-WPA has the advantage of a faster convergence speed. Although their convergence costs are similar, the PSO-GA-WPA can obtain a convergent solution in a shorter time.

Table 2. Comparison of simulation results.

Scenario	Algorithm	Cost Function				Convergence Time			
		Maximum	Minimum	Mean	Standard Deviation	Maximum	Minimum	Mean	Standard Deviation
5 vs. 8	PSO	31.69	28.72	29.49	0.78	46.77	16.23	36.31	7.8
	WPA	29.7	28.72	28.92	0.37	7.47	2.64	4.22	1.43
	PSO-GA-WPA	30.98	28.72	29.23	0.52	4.36	1.45	2.53	0.17
20 vs. 30	PSO	582.86	412.4	511.84	37	-	-	-	-
	WPA	279.15	239.23	261.95	12.65	100.55	64.89	70.29	10.17
	PSO-GA-WPA	269.84	235.28	253.6	10.2	51	23.67	29.56	4.77
100 vs. 150	PSO	8586	7929	8212	225	-	-	-	-
	WPA	1908	1650	1783	80	2508	2160	2255	102
	PSO-GA-WPA	1880	1480	1606	78	1894	1784	1844	27

4.2. The Real-Time Analysis of the PSO-GA-DWPA

In practical application, the situation of the battlefield is rapidly changing; as a result, the time of task assignment is limited. When the optimization problem scale is small, the global optimal solution or sub-optimal solution can be obtained within the specified time. However, if the dimension of the optimization problem is high, it is impossible to get the optimal solution in the specified time. In this case, an algorithm with a faster convergence speed is required in the early stage to obtain a relatively reasonable solution.

As shown in Figure 9b, to get the convergent solution of the optimization problem with 8 dimensions, the WPA needs 5 s and the PSO-GA-DWPA needs 2 s, which meets the requirement of real time. While as shown in Figures 10b and 11b, the WPA needs 70 s and the PSO-GA-DWPA needs 30 s when the dimension of the problem is 30, even worse, the WPA needs 2500 s and the PSO-GA-DWPA needs 1800 s when the dimension of the problem goes up to 150. Hence in practice, a compromise must be made between the optimal solution and the convergence time.

In the 150-dimensional scenario, it is required to make the decision of task assignment within a specified time. Fifty experiments with three methods were conducted to compare the adaptability of the three methods. The experimental results are shown in Figure 12. As can be seen from the results in the figure, the PSO-GA-DWPA obtains a better solution within the specified time. If we need to obtain a better but not optimal solution in finite time, the PSO-GA-DWPA is a better choice.

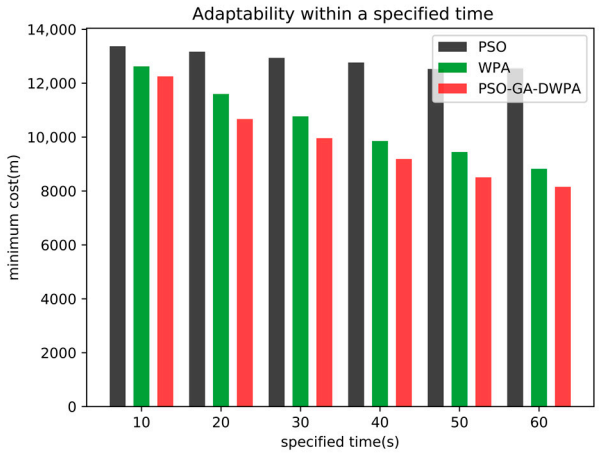


Figure 12. Mean of the Monte Carlo runs: 100 vs. 150 scenario, minimum cost within specified time.

5. Conclusions

Aiming at the problem of the UAV swarm task assignment, this paper introduces a novel swarm intelligence algorithm, the wolf pack algorithm (WPA), which is particularly suitable for high-dimensional continuous optimization problems. In order to apply this method to the discrete optimization problem and improve the convergence speed, this paper proposes the PSO-GA-DWPA

based on matrix coding and the principles of particle swarm optimization and genetic algorithm. According to a general task assignment model and simulation experiments, this method proved to be applicable for high-dimensional task assignment.

The simulation results show that in the case of a small dimension such as Scenario 1, the PSO-GA-DWPA has almost the same optimization ability as the PSO. However, when the dimension increases to 30 in Scenario 2, the PSO-GA-DWPA is obviously superior to PSO both in terms of convergence speed and solution accuracy. When the dimension increases to 150 in Scenario 3, the PSO fails, while the PSO-GA-DWPA can still obtain an effective solution. Therefore, this method can serve as a useful reference for the UAV swarm task assignment.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L., Y.M. and J.W.; formal analysis, Y.L.; investigation, Y.L.; resources, Y.L., Y.M., J.W. and L.H.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L.; visualization, Y.L.; supervision, Y.M., J.W. and L.H.; project administration, Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the anonymous reviewers for their constructive comments and suggestions, which improved the quality of the paper.

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

References

1. Peters, J.R.; Surana, A.; Bullo, F. Robust Scheduling and Routing for Collaborative Human/Unmanned Aerial Vehicle Surveillance Missions. *J. Aerosp. Inf. Syst.* **2018**, *15*, 1–19. [CrossRef]
2. Ran, H.; Zhou, R.; Dong, Z. Cooperative guidance of multi aircraft in beyond-visual-range air combat. *J. Beijing Univ. Aeronaut. Astronaut.* **2014**, *40*, 1457–1462.
3. Cattrysse, D.; van Wassenhove, L. A survey of algorithms for the generalized assignment problem. *Eur. J. Oper. Res.* **1992**, *60*, 260–272. [CrossRef]
4. Drexl, M. Synchronization in vehicle routing-a survey of VRPs with multiple synchronization constraints. *Transp. Sci.* **2012**, *46*, 297–316. [CrossRef]
5. Shima, T.; Rasmussen, S.J.; Sparks, A.G.; Passino, K.M. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Comput. Oper. Res.* **2006**, *33*, 3252–3269. [CrossRef]
6. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
7. Shima, T.; Rasmussen, S.J. Tree search algorithm for assigning cooperating UAVs to multiple tasks. *Int. J. Robust Nonlinear Control* **2008**, *18*, 135–153.
8. Alighanbari, M.; How, J. Cooperative task assignment of unmanned aerial vehicles in adversarial environments. In Proceedings of the American Control Conference, Portland, OR, USA, 8–10 June 2005; pp. 4661–4666.
9. Ling, X. The approximate optimal solution of the traveling salesman problem is obtained by the optimal exhaustive method. *Comput. Appl. Res.* **1998**, *15*, 82–83.
10. Lipson, J.D. Newton's method: A great algebraic algorithm. In Proceedings of the Third ACM Symposium on Symbolic & Algebraic Computation, Yorktown Heights, NY, USA, 10 August 1976; pp. 260–270.
11. Ji, S.; Ye, J. An accelerated gradient method for trace norm minimization. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 457–464.
12. Wang, C.; Mu, D.; Zhao, F.; Sutherland, J.W. A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup-delivery and time windows. *Comput. Ind. Eng.* **2015**, *83*, 111–122. [CrossRef]
13. Diaz, J.A.; Fernandez, E. A Tabu search heuristic for the generalized assignment problem. *Eur. J. Oper. Res.* **2011**, *132*, 22–38. [CrossRef]
14. Shima, T.; Rasmussen, S.J.; Sparks, A.G. UAV cooperative multiple task assignments using genetic algorithms. In Proceedings of the American Control Conference, Portland, OR, USA, 8–10 June 2005; pp. 2989–2994.
15. Boveiri, H.R. An incremental ant colony optimization based approach to task assignment to processors for multiprocessor scheduling. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 498–510. [CrossRef]

16. Sujit, P.B.; George, J.M.; Beard, R. Multiple UAV task allocation using particle swarm optimization. In Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI, USA, 18–21 August 2008; pp. 1–9.
17. Jia, Z.; Yu, J.; Ai, X.; Xu, X.; Yang, D. Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm. *Aerosp. Sci. Technol.* **2018**, *76*, 112–125. [CrossRef]
18. Liang, G.; Kang, Y.; Xing, Z.; Yin, G. UAV Cooperative Multi-task Assignment Based on Discrete Particle Swarm Optimization Algorithm. *Comput. Simul.* **2018**, *35*, 22–28.
19. Su, F. Research on Distributed Online Cooperative Mission Planning for Multiple Unmanned Combat Aerial Vehicles in Dynamic Environment. PhD Thesis, National University of Defense Technology, Changsha, China, 2013.
20. Wu, X. Coordination Tasks Pre-Allocation and Redistribution Studies in UAVs. Master's Thesis, Nanchang Hangkong University, Nanchang, China, 2018.
21. Braun, H. On solving travelling salesman problems by genetic algorithms. In *International Conference on Parallel Problem Solving from Nature*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 129–133.
22. Tuani, A.F.; Keedwell, E.; Collett, M. Heterogenous Adaptive Ant Colony Optimization with 3-opt local search for the Travelling Salesman Problem. *Appl. Soft Comput.* **2020**, *106720*, 1–14. [CrossRef]
23. Matin, H.N.; Yekkehkhany, A.; Nagi, R. Probabilistic Analysis of UAV Routing with Dynamically Arriving Targets. In Proceedings of the 22th International Conference on Information Fusion, Ottawa, ON, Canada, 2–5 July 2019; pp. 1–8.
24. Di, B.; Rui, Z.; Jiang, W. Distributed coordinated heterogeneous task allocation for unmanned aerial vehicles. *Control Decis.* **2013**, *28*, 274–278.
25. Oh, G.; Kim, Y.; Ahn, J.; Choi, H.-L. Market-Based Task Assignment for Cooperative Timing Missions in Dynamic Environments. *J. Intell. Robot. Syst.* **2017**, *87*, 97–123. [CrossRef]
26. Zhang, J.; Wang, Y.; Li, F.; Zhou, T. Dynamic Task Assignment Problem of Multi-agent. *Electron. Technol. Softw. Eng.* **2018**, *18*, 255–258.
27. Li, Y.; Dong, Y. Weapon-target assignment based on simulated annealing and discrete particle swarm optimization in cooperative air combat. *Acta Aeronaut. Et Astronaut. Sin.* **2010**, *31*, 626–631.
28. Kumar, N.; Vidyarthi, D.P. A novel hybrid PSO-GA metaheuristic for scheduling of DAG with communication on multiprocessor systems. *Eng. Comput.* **2016**, *32*, 35–47. [CrossRef]
29. López LF, M.; Blas, N.G.; Albert, A.A. Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations. *Soft Comput.* **2018**, *22*, 2567–2582. [CrossRef]
30. Guo, T.; Michalewicz, Z. Inver-over operator for the TSP. In *International Conference on Parallel Problem Solving from Nature*; Springer: Heidelberg, Germany, 1998; pp. 803–812.
31. Yan, J.; Li, X.; Liu, B. Cooperative task allocation of multi-UAVs with mixed DPSO-GT algorithm. *J. Natl. Univ. Def. Technol.* **2015**, *37*, 165–171.
32. Wu, H.; Zhang, F.; Wu, L. New Swarm Intelligence Algorithm-Wolf Pack Algorithm. *Syst. Eng. Electron.* **2013**, *35*, 2430–2437.
33. Liu, Y.; Li, W.; Wu, H.; Song, W. Track Planning for Unmanned Aerial Vehicles Based on Wolf Pack Algorithm. *J. Syst. Simul.* **2015**, *27*, 1838–1843.
34. Zhang, L.; Lei, Z.; Liu, S.; Zhou, J. Three-Dimensional Underwater Path Planning Based on Modified Wolf Pack Algorithm. *IEEE Access* **2017**, *5*, 22783–22795. [CrossRef]