applied sciences

Article

Multi-Population Parallel Wolf Pack Algorithm for Task Assignment of UAV Swarm

Yingtong Lu [1,2], Yaofei Ma [1,2,*] and Jiangyun Wang [1]

[1] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; luyingtong@buaa.edu.cn (Y.L.); wangjiangyun@buaa.edu.cn (J.W.)

[2] Science and Technology on Electro-Optic Control Laboratory, Luoyang 471000, China

* Correspondence: ma_yaofei@buaa.edu.cn

Abstract: The effectiveness of the Wolf Pack Algorithm (WPA) in high-dimensional discrete optimization problems has been verified in previous studies; however, it usually takes too long to obtain the best solution. This paper proposes the Multi-Population Parallel Wolf Pack Algorithm (MPPWPA), in which the size of the wolf population is reduced by dividing the population into multiple sub-populations that optimize independently at the same time. Using the approximate average division method, the population is divided into multiple equal mass sub-populations whose better individuals constitute an elite sub-population. Through the elite-mass population distribution, those better individuals are optimized twice by the elite sub-population and mass sub-populations, which can accelerate the convergence. In order to maintain the population diversity, population pretreatment is proposed. The sub-populations migrate according to a constant migration probability and the migration of sub-populations are equivalent to the re-division of the confluent population. Finally, the proposed algorithm is carried out in a synchronous parallel system. Through the simulation experiments on the task assignment of the UAV swarm in three scenarios whose dimensions of solution space are 8, 30 and 150, the MPPWPA is verified as being effective in improving the optimization performance.

Keywords: Wolf Pack Algorithm (WPA); multi-population optimization algorithms; Multi-Population Parallel Wolf Pack Algorithm (MPPWPA); approximate average division method; elite-mass population distribution; population pretreatment; synchronous parallel system; task assignment of UAV swarm

check for updates

Citation: Lu, Y.; Ma, Y.; Wang, J. Multi-Population Parallel Wolf Pack Algorithm for Task Assignment of UAV Swarm. *Appl. Sci.* **2021**, *11*, 11996. https://doi.org/10.3390/app112411996

Academic Editor: Paola Pellegrini

Received: 12 November 2021
Accepted: 13 December 2021
Published: 16 December 2021

1. Introduction

During the Nagorno-Karabakh conflict in 2020, it was shocking to witness UAV swarms being employed to attack the ground targets precisely and roundly. It is an inevitable trend that Unmanned Aerial Vehicle (UAV) swarms will play an important role in future war [1]. Consequently, the task assignment of UAV swarms has become a research hotspot in recent decades.

There are two core problems in task assignment. One is to establish a model for a specific operational process which has obtained abundant achievements after decades of extensive research, such as TSP (Travelling Salesman Problem), VRP (Vehicle Routing Problem), MILP (Mixed Integer Linear Programming), CMTAP (Cooperative Multiple Task Assignment Problem) and their extensions. The other is to design an appropriate optimization algorithm to solve the model. The quality of an optimization method determines the quality of a UAV's task sequence and then affects the overall combat effectiveness. Although the development of optimization algorithms is also thriving, there are few existing studies that solve the task assignment with high dimensionality (task assignment for 100 UAVs, for instance).

In this paper, we proposed a novel heuristic method called the multi-population parallel Wolf Pack Algorithm (MPPWPA), which is improved from the basic Wolf Pack

Algorithm (WPA) and is suitable for resolving the high-dimensional task assignment problems. The main contribution of this paper is to incorporate the multi-population optimization method and the parallel computing with WPA to reduce the convergence time. The remainder of this paper is organized as follows. Section 2 mainly describes the basic knowledge and corresponding literature review. MPPWPA is proposed in Section 3, where the communication structures in multi-population optimization and the parallel propulsion modes are described in detail. In Section 4, the performance of MPPWPA is tested and compared with PSO, GA, ABC and WPA in three task assignment scenarios for UAV swarms. Finally, Section 5 concludes the paper and briefly explores the outlook for future work.

2. Basic Knowledge and Literature Review

2.1. Task Assignment Model

The background of the mission is to attack static present targets on the ground using a UAV swarm consisting of isomorphic attack UAVs with limited weapons. The objective of the mission is to attack all targets in the shortest time or with the least total fuel consumption. When a UAV's weapons are insufficient to cope with one target, multiple UAVs need to be scheduled to attack the target at the same time. UAVs can control their speeds to achieve the salvo attack. Any UAV needs to avoid obstacles and no-fly zones as it flies to the target. In a specific mission, various constraints should be considered to assign the UAVs, such as the mobility constraints, the weapon quantity constraints, the fuel quantity (or flying range) constraints and the flyable zone constraints.

In this paper, a simplified task assignment model is described, which involves as little uncertain information as possible, so as to facilitate the performance comparison of optimization algorithms. The objective function is to minimize the costs of all tasks, including the cost of the total range and the cost of the time to complete all tasks [2].

To formalize the problem, the following assumptions are set up.

1. There is no consideration of obstacles and no-fly zones, so the range between the UAV and target can be expressed by their straight-line distance.
2. UAVs fly at fixed altitudes with the same constant velocity. Thus, the flight time can be equivalent to the fight range.
3. The UAV drops the weapon directly above the target, so the task position is the projection of the target position on the flight level. In other words, the UAV's fight range flying to the task position is the straight-line distance between the UAV's position and the target position in the horizontal plane, regardless of the altitude.
4. The time consumption involved in preparing and firing the weapon is not taken into account. In other words, the time cost to complete a task includes only the flight time.
5. Each target can be attacked only once, while any UAV can attack multiple targets.
6. Targets' initial positions are revealed.

The UAV swarm consists of N_V attack UAVs, which can be expressed as $\mathbf{V} = \{V_i | i = 1,,2,\cdots,N_V\}$, and N_T targets, which can be expressed as $\mathbf{T} = \{T_j | j = 1,,2,\cdots,N_T\}$.

The task sequence of UAV $V_i \in \mathbf{V}$ is

$$\begin{aligned} plan_i &= \left\{ stage_i^{(k)} \middle| k = 1,2,\cdots,N_{si} \right\} \\ stage_i^{(k)} &= \left(T_n, Range_{m,n}^i\right), T_n \in \mathbf{T} \end{aligned} \tag{1}$$

where $stage_i^{(k)}$ is the stage k of UAV V_i, N_{si} is the task number assigned to UAV V_i, T_n is the target of UAV V_i in the k^{th} stage, and $Range_{m,n}^i$ is the distance between T_m and T_j for UAV V_i, where m $= 0,1,2,\cdots N_T$ and n $= 1,2,\cdots N_T$, T_0 is a virtual target representing the initial position of a UAV.

This task assignment problem is similar to the Vehicle Routing Problem (VRP) in which UAVs equate to vehicles, targets equate to customers and the initial position of each UAV equates to a depot. VRP [3] is a reasonably well-studied problem with extensive related derivative problems, such as the Capacitated Vehicle Routing Problem (CVRP) [4], the Vehicle Routing Problem with Time Window (VRPTW) [5], the Split Delivery Vehicle Routing Problem (SDVRP) [6], the Dynamic Vehicle Routing Problem (DVRP) [7], the Vehicle Routing Problem with Simultaneous Delivery Pickup (VRPSDP) [8] and so on. At present, the existing VPR models can basically meet the requirements for the modeling of task assignment problems.

In this paper, we introduce the basic VRP to build the task assignment model. The decision variable of task assignment is $x^i_{m,n} \in \{0, 1\}$, representing whether UAV V_i attacks target T_n after T_m; $x^i_{m,n} = 1$ represents an attack, other values signify no attack.

The total range of all UAVs is

$$J_1 = \sum_{i=1}^{N_V} \sum_{m=0}^{N_T} \sum_{n=1}^{N_T} x^i_{m,n} \cdot Range^i_{m,n} \tag{2}$$

The time to complete all tasks refers to the maximum time taken by any UAV to complete its own task sequence, and is expressed as

$$J_2 = \max_{i \in N_V} Time_i \tag{3}$$

where $Time_i$ is the time of UAV V_i to finish its all tasks and can be equivalent to the total range of UAV V_i, so the time to complete all tasks can be represented as

$$J_2^* = \max_{i \in N_V} \sum_{m=0}^{N_T} \sum_{n=1}^{N_T} x^i_{m,n} \cdot Range^i_{m,n} \tag{4}$$

To keep the values of the two costs within the same order of magnitude, the total range of all UAVs is equivalent to the average range, represented as

$$J_1^* = \frac{\sum\limits_{i=1}^{N_V} \sum\limits_{m=0}^{N_T} \sum\limits_{n=1}^{N_T} x^i_{m,n} \cdot Range^i_{m,n}}{N_V} \tag{5}$$

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

$$\min J = \omega_1 J_1^* + \omega_2 J_2^* = \omega_1 \frac{\sum\limits_{i=1}^{N_V} \sum\limits_{m=0}^{N_T} \sum\limits_{n=1}^{N_T} x^i_{m,n} \cdot Range^i_{m,n}}{N_V} + \omega_2 \max_{i \in N_V} \sum_{m=0}^{N_T} \sum_{n=1}^{N_T} x^i_{m,n} \cdot Range^i_{m,n} \tag{6}$$

$$\begin{aligned} s.t. \quad (1) \quad & \sum_{i=1}^{N_V} \sum_{m=0}^{N_T} x^i_{m,n} = 1 \\ (2) \quad & \sum_{i=1}^{N_V} \sum_{m=0}^{N_T} \sum_{n=1}^{N_T} x^i_{m,n} = N_T \end{aligned} \tag{7}$$

where ω_1 and ω_2 are weighting factors reflecting the importance of each performance criterion, decided by the commander, and $\omega_1 + \omega_2 = 1$. Equation (7) indicates constraints: each target can be attacked only once, and all targets must be attacked.

2.2. Optimization Methods

Optimization methods for the task assignment problem of UAV swarms are generally studied in three categories.

The first category treats task assignment as a programming problem, such as the Hungarian algorithm [9], the branch and bound search algorithm [10], dynamic programming [11], the exhaustive method [12], Newton's method [13], the gradient method [14] and so on. These methods can obtain the optimal solution if there is a solution, but it is difficult to solve non-convex NP-hard problems. Generally, these methods need to abstract the problem to establish a mathematical model, so high mathematical ability is required, especially for large-scale problems. The methods that are easy to implement, such as the exhaustive method and the branch and bound search algorithm, have high time and space complexity. When the problem size increases, the difficulty of solving the problem increases sharply, and the time consumption increases exponentially [15].

The second category includes mainly the distributed optimization methods, such as the auction algorithm [16], the market-based decentralized algorithm [17], the contract network [18], etc. These methods utilize a market-based decision strategy as the mechanism for decentralized task selection and can naturally converge to a conflict-free solution. Luc Brunet et al. [19] introduced consensus to the auction algorithm and proposed a consensus-based auction algorithm (CBAA) and a consensus-based bundle algorithm (CBBA) to improve the efficiency of conflict resolution. Yu X et al. [20] applied the CBAA to the task assignment of complex space crafts and found that the algorithm can guarantee successful assignment. These methods have excellent performance in small-scale distributed systems, but with the increase in the numbers of individuals in communication networks, the computation and time required for conflict resolution increase greatly. Hence, these methods are generally applicable to the coordination tasks in small-scale communication networks. In addition, these methods need build objective functions named bidding functions for individuals, and the bidding function directly determines the success and quality of the solution.

The last category employs mainly bio-based heuristic algorithms, such as the Ant Colony Optimization Algorithm (ACO) [21], the Genetic Algorithm (GA) [22,23], the Particle Swarm Optimization Algorithm (PSO) [24], etc. Such methods are inspired by the mechanism of natural evolution of biological populations and can approach optimal or sub-optimal solutions with finite calculation costs. Due to the characteristics of simplicity, robustness, parallelism, wide applicability and low structural requirements for the problem model, they are widely applied to various optimization problems [25]. In this paper, we focus on the application of heuristic algorithms in the task assignment of UAV swarms. The traditional classical heuristic algorithms still occupy the main position of optimization at present, and scholars have been committed to overcoming their defects of premature convergence and have put forward a large number of variants to adapt to various problems. In addition to the study of classical heuristic algorithms, many scholars are committed to designing new algorithms, such as the Fish Swarm Algorithm (FSA) [26], Bacterial Foraging Optimization (BFO) [27], the Shuffled Frog Leaping Algorithm (SFLA) [28], the Artificial Bee Colony (ABC) algorithm [29], the Wolf Pack Algorithm (WPA) [30] and so on. ABC has attracted much attention due to its simple optimization mechanism and the ability to move beyond a local optimal. In [29], compared with PSO and GA, it was found to perform well in solving five high dimensional numerical benchmark functions including the Griewank, Rastrigin, Rosenbrock, Ackley and Schwefel functions. Similarly, WPA also showed outstanding performance in high-dimensional and multimodal continuous functions compared with PSO, GA and FSA in [30], and this was verified by 15 complex benchmark continuous functions such as Easom, Matyas, Trid6, Sumsquares, Spere, Booth, Bohachevsky1, Eggcrate, Schaffer, Six Hump Camel Back, Bohachevsky3, Bridge, Rastrigin, Quadric and Ackley. This paper mainly studies the application and improvement of the WPA to task assignment in UAV swarms. Because of the WPA's outstanding performance in high-dimensional continuous problems, we apply it to solve the task assignment problem.

2.3. The Basics of WPA

The WPA simulates the cooperative hunting of wolves according to their system of dividing responsibilities. The prey represents the optimal solution and each wolf represents a candidate solution. The wolf population consists of a leader wolf who is the current best solution, a small number of exploring wolves who explore the solution space in a specific way and a large number of fierce wolves who explore the solution space referring to the leader wolf. According to the laws of nature, the leader wolf is constantly replaced and the worst wolves are removed and replaced by new wolves that are generated randomly.

The WPA has many control parameters, as listed in Table 1.

Table 1. Parameters of WPA and PSO-GA-DWPA.

Parameters	Meaning	Parameters	Meaning
$I_{\max}$	Maximum iterations	d_{near}	Threshold distance for calling behavior
N	Size of population	α	Scale factor of exploring wolves
$step_a$	Walking step	β	Scale factor of wolf population update
$step_b$	Raid step	$h_{\min}$	Minimum walking directions
$step_c$	Siege step	$h_{\max}$	Maximum walking directions
$T_{\max}$	Maximum cycles of walking behavior		

2.3.1. Integer Matrix Coding

According to the task assignment model in this paper, the position of each wolf is expressed by two-dimensional matrix coding as

$$X_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{iN_T} \\ y_{i1} & y_{i2} & \cdots & y_{ij} & \cdots & y_{iN_T} \end{bmatrix} \tag{8}$$

where X_i denotes the position of the wolf $i \in N$, N is the population size of the wolves, the dimension of the variable is N_T because all targets must and can only be attacked once. The first row of the code is the index of UAVs ($x_{ij} \in N_V$) and elements in this row can be repeated because a UAV can attack multiple targets. The second row of the code is the index of targets ($y_{ij} \in N_T$) and the elements in this row are different. For any UAV, the order of a target's index is its task sequence. For instance, there are 3 UAVs attacking 5 targets and the coding matrix is $X_i = \begin{bmatrix} 1 & 1 & 2 & 1 & 2 \\ 3 & 2 & 5 & 4 & 1 \end{bmatrix}$, then the task assignment scheme is as follows: $plan_1 = \left\{ \left(3, Range^{(1)}_{1,3}\right), \left(2, Range^{(2)}_{1,2}\right), \left(4, Range^{(3)}_{1,4}\right) \right\}$,$plan_2 = \left\{ \left(5, Range^{(1)}_{2,5}\right), \left(1, Range^{(2)}_{2,1}\right) \right\}$, $plan_3 = NULL$.

2.3.2. The Optimization Mechanism of the WPA

The optimization process of the WPA is shown in Figure 1. There are three searching processes for the optimal solution in one iteration.

Step 1: Walking behavior. Except the leader wolf, S ($S = randint\left[\frac{N}{\alpha+1}, \frac{N}{\alpha}\right]$) best wolves become exploring wolves and explore the solution space by individual mutation, as Equation (9) and Figure 2 show. Then, the new position of wolf i will be the one that determines the objective value minimum, as expressed in Equation (10). Any exploring wolf better than the leader wolf will become the new leader wolf, then go to the next Step, otherwise Step 1 will be repeated. For the walking behavior, this step is repeated T_{max} times at most.

$$X^p_{i,temp} = F_1(X_i, step_a), p = \{1, 2, \cdots, h\} \tag{9}$$

$$X_{i,new} = \begin{cases} \underset{X^p_{i,temp}, p\in h}{\operatorname{argmin}} J\left(X^p_{i,temp}\right), & if \min J\left(X^p_{i,temp}\right) < J(X_i) \\ X_i, & else \end{cases} \tag{10}$$

where h ($h = randint(h_{min}, h_{max})$) means the number of variants of X_i and $J(\cdot)$ is the objective function.

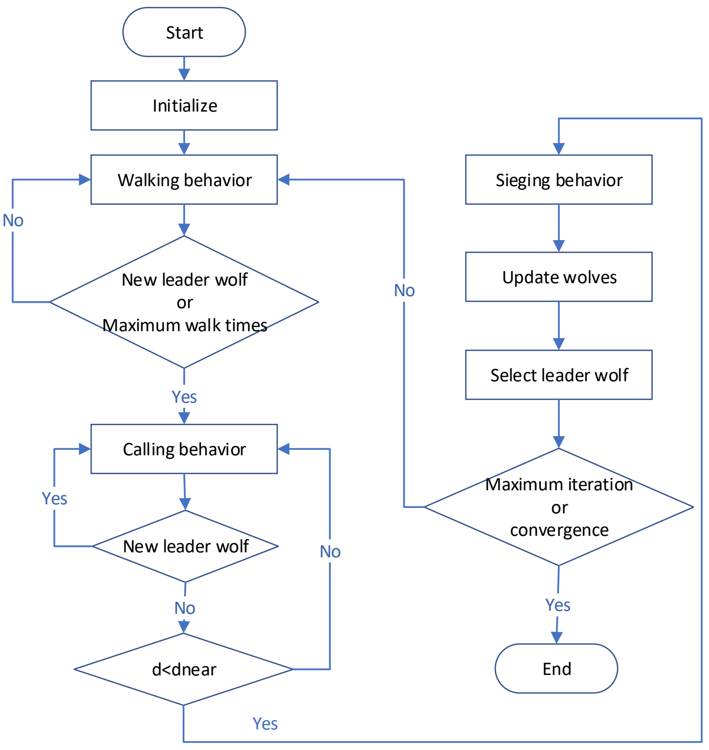


Figure 1. The optimization process of the WPA.

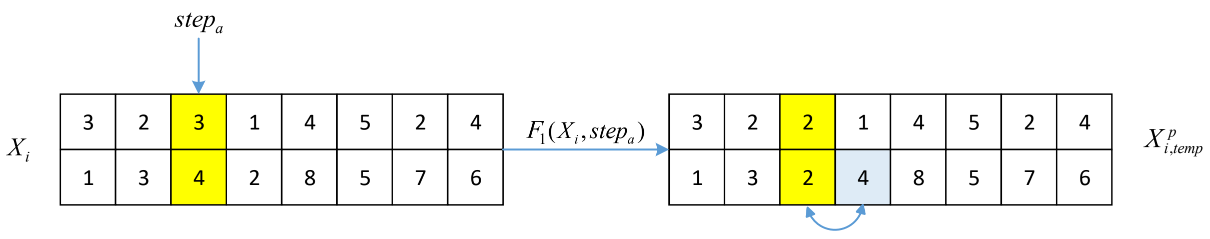


Figure 2. Operation in walking behavior. $F_1(X_i, step_a)$ randomly selects $step_a$ columns in X_i, the values of the first row in these columns are randomly selected from $[1, N_V]$, while the values of the second row are randomly exchanged.

Step 2: Calling behavior. M ($M = N - S - 1$) wolves are fierce wolves and copy part of the leader wolf's position, as shown in Equations (11) and (12) and Figure 3. Any fierce wolf better than the leader replaces the leader wolf and Step 2 is repeated. Otherwise, the next step involves continuing to copy the position of the leader wolf until the distance to the leader wolf (d) is less than the threshold (d_{near}).

$$X_{j,tepm} = F_2(X_j, X_{leader}, step_b) \tag{11}$$

$$X_{j,new} = \begin{cases} X_{j,tepm}, & if \min J(X_{j,tepm}) < J(X_j) \\ X_j, & else \end{cases} \tag{12}$$

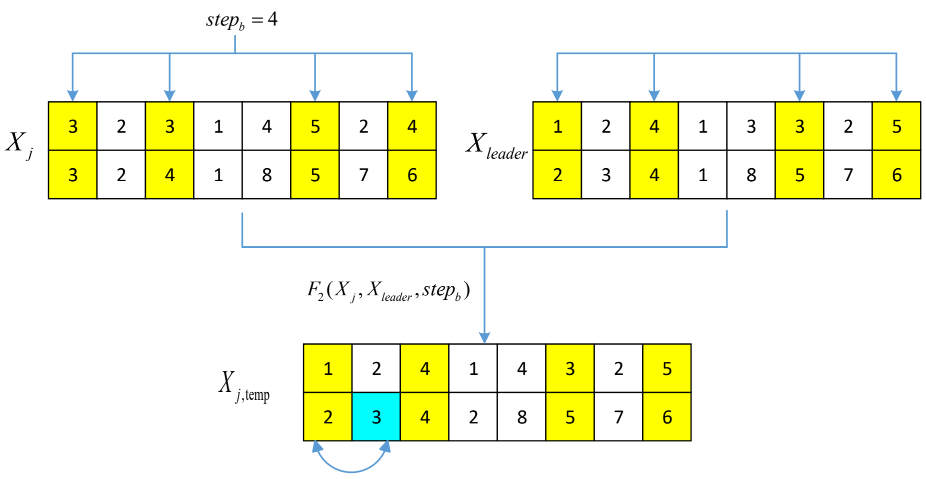


Figure 3. Operation in calling behavior. $step_b$ columns in X_j are randomly selected and are replaced by the corresponding columns of the leader wolf, while the duplicate value in the second row is replaced with the original value in the changed column.

Step 3: Sieging behavior. Except for the leader wolf, all wolves copy little of the leader wolf's position, as shown in Equations (13) and (14). The operation is similar to that of the calling behavior and the difference is that $step_c$ is much less than $step_b$.

$$X_{i,temp} = F_2(X_i, X_{leader}, step_c) \tag{13}$$

$$X_{i,new} = \begin{cases} X_{i,tepm}, & if \min J(X_{i,tepm}) < J(X_i) \\ X_i, & else \end{cases} \tag{14}$$

Step 4: Updating the wolf population. The R ($R = randint\left[\frac{N}{\beta+1}, \frac{N}{\beta}\right]$) worst wolves are removed and replaced with new wolves that are generated randomly.

2.3.3. Analysis of WPA

It can be seen from the optimization process that the algorithm integrates various optimization ideas and considers almost all the problems involved in heuristic methods. Walking behavior is the preliminary means of exploiting the solution space, and better candidate solutions are found through the tentative advances of exploring wolves around themselves. Calling behavior shows the ability to explore the solution space, and it is a global optimization to accelerate the convergence of the algorithm. While sieging behavior is a local optimization aiming at the current best solution to avoid falling prematurely into the local optimum. This behavior helps to improve the accuracy of candidate solutions. By removing the worst individuals and replacing them with new ones, the update of the wolf population increases the population diversity to some extent and exploits the solution space again. Although there are many control parameters in the WPA, small changes to parameters do not affect the optimization performance because the parameters have a wide range of values. In other words, the algorithm is robust to control parameters.

Since the WPA was proposed in 2013, it has been applied to optimization problems in many fields such as parameter optimization [31–33], the power coordinated optimization model in Active Distribution Networks (ADNs) [34,35], image compression [36], image fusion [37], etc.

A series of improvements to the basic WPA have been proposed. Wu et al. [38] designed binary coding based on the WPA and proposed the Binary Wolf Pack Algorithm (BWPA) to solve the 0–1 knapsack problem. Guo et al. [39] improved the BWPA by adopting an adaptive step length and replacing duplicate best wolves with new wolves after each iteration, which enhanced the global convergence and maintained the population diversity. Li et al. [40] introduced the Oppositional Wolf Pack Algorithm (OWPA), in which the initial population is selected from two oppositional populations so as to enhance the global convergence. Xian et al. [41] introduced chemotactic behavior and elimination-dispersal behavior of bacterial foraging optimization (BFO) into the walking behavior of the WPA to overcome the slow convergence speed and avoid entrapment into the local extremum. Lu et al. [42] proposed the Discrete Wolf Pack Algorithm, with the principles of the Particle Swarm Optimization and Genetic Algorithm (PSO-GA-DWPA), which melts the PSO into walking behavior, and introduced the gene fragment replication method of GA into calling behavior and sieging behavior; their simulation results showed that the proposed algorithm was better than WPA both in convergence speed and accuracy. Xiu-Wu et al. [43] adopted a variable step to improve the global search capability. Chen et al. [44] integrated the WPA with GA to retain as many elite genes as possible and delete calling behavior to improve the convergence speed. The above improvements improve the optimization performance to a certain extent, but the problem of long convergence time for discrete problems with high-dimensional solution space has not been effectively solved.

Compared with classical optimization algorithms such as GA and PSO, the WPA has excellent performance in terms of both accuracy and time consumption. However, there is still room to improve it, especially in discrete problems where its performance is not as good as in continuous problems. Improving the details of the WPA leads to a limited improvement in the performance. In order to solve the task assignment model faster, we need to find a more effective method to modify the algorithm.

2.4. Multi-Population Optimization Method

The convergence speed and accuracy of the heuristic algorithms are greatly influenced by population diversity, which reflects in the differences of candidate individuals. In single-population algorithms, population diversity is determined in the initial population and decreases as the individuals keep moving towards the global extreme. Multi-population optimization methods are upgrades of heuristic algorithms based on the theory of co-evolution [45]. The main advantage of multi-population methods is the maintaining of population diversity as far as possible by making candidate individuals of sub-populations spread over the entire search space.

2.4.1. The Basics of the Multi-Population Optimization Method

A population is divided into multiple small sub-populations who evolve with their own evolution operations; every once in a while, sub-populations interact with each other via merging and communication processes to maintain population diversity and avoid premature convergence [46,47]. The general flow of the multi-population optimization method is shown in Figure 4. Generally, the algorithm starts with initialization, including the setting of parameters, the generation of an initial population of solutions and the evaluation of the population. Then, the population is divided into multiple sub-populations and each sub-population is optimized independently with its own algorithm within a certain number of iterations. After that, sub-populations communicate to update themselves according to some rules and this process is called population migration. This step is the key to maintain diversity and accelerate the overall optimization progress. Finally, the process stops once the termination condition is met.

The steps are described in detail as follows.

Step 1: Configuration of the parameters. The parameters of multi-population optimization method include the maximum iterations, the size of the population, the number of sub-populations, the parameters of the algorithm corresponding to each sub-population and the parameters related to population migration.

Step 2: Generation of the initial population. An initial population of solutions is created randomly or via some methods.

Step 3: Evaluation of the population. The fitness of each individual in the population is calculated according to the objective function.

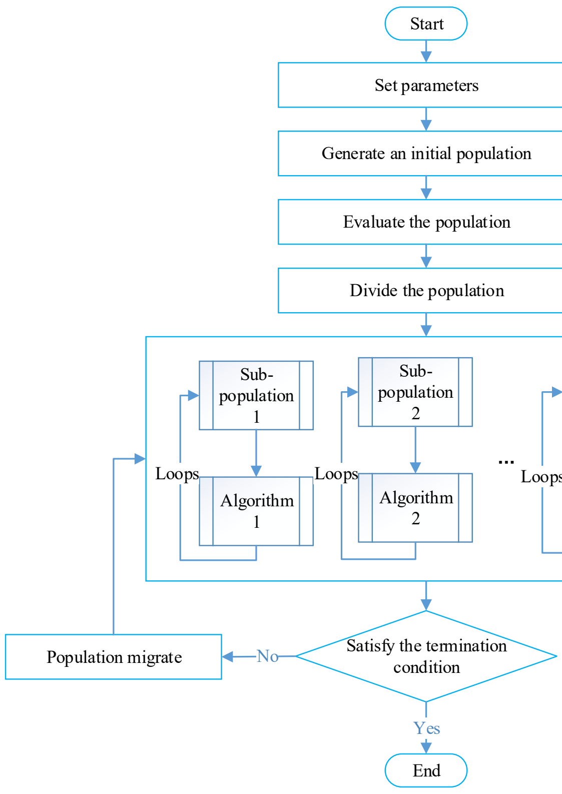


Figure 4. General flow of the multi-population optimization algorithm.

Step 4: Division of the population into sub-populations. Sub-populations may have the same or different sizes. Individuals of the population are divided into sub-populations randomly or in the light of some criterion.

Step 5: Sub-populations perform the optimization process simultaneously and independently, which is the main step of the method. Within certain iterations, each sub-population searches the solution space according to its own algorithm and parameters. According to sub-population selections of the optimization algorithms, multi-population optimization methods can be divided into isomorphic multi-population optimization and heterogeneous multi-population optimization. In the former method, all sub-populations have the same optimization algorithm and parameters, while in the latter method, the optimization algorithms or optimization parameters of each sub-population are different.

Step 6: Judging whether the termination condition is met. Once the termination condition is met, the evolution process is stopped and the current best solution is output. Otherwise, Step 7 is begun.

Step 7: Population migration, which is the key for multi-populations to maintain diversity and accelerate optimization. Through specific communication mechanisms and migration rules, each sub-population sends information on some individuals to other sub-populations and receives external individuals to replace its own ones. This is a process of fusion and renewal for sub-populations. Then, Step 5 is returned to.

In the last decade, multi-population optimization methods have been studied in many fields, including cognitive radio networks [48], energy power [49], Cloud manufacturing [50], mobile ad-hoc networks [51], job-shop scheduling [52], path planning [53] and so on. Since each sub-population can run its own optimization algorithm independently, multi-population optimization methods have strong flexibility in selecting algorithms. Hao et al. [54] applied the same algorithm with multiple sets of parameters in different sub-populations to generate high-quality and effective paths of robots. Through a high number of sub-populations interacting in parallel, the effect of each sub-population's parameters is compensated by individuals selected from other populations [55]. Wang et al. [56] used the Fruit fly optimization algorithm (FOA) for all sub-populations, but preprocessed the individuals of sub-populations with different methods including chaos theory and the fish swarm algorithm, so that the whole population could maintain diversity while tending to the optimal solution. Yoshida et al. [57] applied multi-population with PSO, where the initial particles were cloned with differential evolution algorithm in each iteration and the optimal particles were selected from all existed particles to the next iteration in order to maintain population diversity and improve the ability to search the solution space. Nseef et al. [58] proposed that the sub-population size can change adaptively with time, which has a remarkable effect on the solving of the dynamic optimization problem.

2.4.2. Factors of Multi-Population Optimization Method

One important factor concerns how to divide the population; specifically, how many sub-populations there should be and the size of each sub-population. Too many sub-populations may waste the limited computation resources because of population migration, while too few sub-populations may cause the advantage of multi-population optimization not to be significant. The size of sub-population is closely related to specific optimization algorithms. At present, the appropriate number and size of sub-populations are obtained through simulation experiments.

The other factor is the migration strategy, which specifically involves three elements—the migration mode, the trigger condition and the individuals participating in migration.

- Migration mode:

As shown in Figure 5, there are two migration modes. The first one, as shown in Figure 5a, is circular migration, in which sub-populations transmit individual information in one direction. This mode is simple to implement and requires no extra computational cost, but it cannot guarantee the quality of population fusion. The second one, as shown in Figure 5b, is interaction migration, in which each sub-population chooses some neighbors to exchange individual information with. This mode is more flexible to maintain population diversity, but it increases additional computational costs.

- Trigger condition:

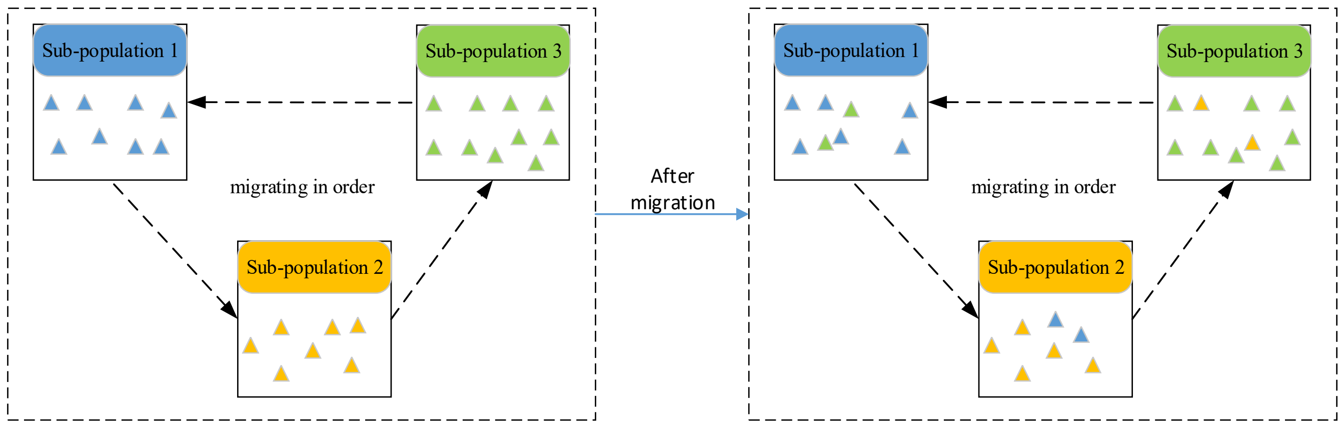
Population migration can maintain population diversity, and the more frequent the migration is, the better the accuracy of the solution will be, but the time of calculation increases accordingly. Therefore, a compromise between the accuracy of the solution and the calculation time is needed. Typically, the trigger condition is set to a certain iteration interval. Events are also used as trigger conditions; sub-populations migrate once a sub-population falls into a local optimum, for example.

- Individuals participating in migration:

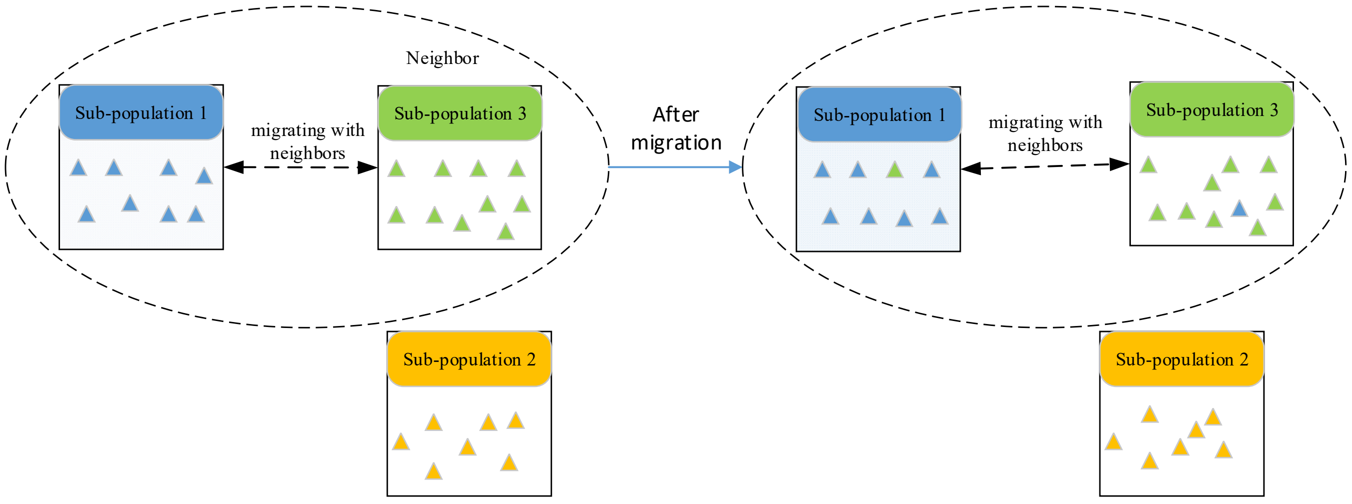
This element involves two issues, the migration rate and individual selection. The former issue concerns how many individuals participate in the migration, and the latter issue concerns which individuals participate in the migration. These two problems are difficult to solve through theoretical derivation. At present, the experience gained through a large number of experiments is that the best individual of a sub-population is sent to another sub-population and replaces its worst individual.

2.5. Basic Communication Structures for Multi-Population Optimization

The classical island model proposed in the early days is shown in Figure 6. In this model, each sub-population and its corresponding optimization algorithm are integrated together as a node. In other words, every node includes not only the state of the sub-population but also the algorithmic process, and it sends and receives the state of specific individuals during the communication with other nodes by point-to-point communication. This undoubtedly increases the difficulty of communication and is not conducive to the large-scale migration of individuals. Besides, the lack of flexibility makes it more difficult to add or remove nodes at runtime.



(**a**)



(**b**)

Figure 5. Migration modes: (**a**) circular migration; (**b**) interaction migration.

To improve the means of communication, the Pool model [59], shown in Figure 7, uses a central Pool to store the individuals and the Pool is accessible by all nodes. Each node interchanges individuals with the Pool, so the communication between nodes will not be affected when adding or removing nodes. Even so, the coupling between the population state and the algorithm still exists; as a result, the framework of the population and algorithm need to be rebuilt when adding a new node and both the size of the sub-population and the parameters of the algorithm cannot be changed during the running time.

An improved Pool model [60] extends the capabilities of the Pool and decouples the state of the sub-population and the algorithm as shown in Figure 8. In this model, the state of all sub-populations is stored in the Pool and a sub-population, which is sent to the corresponding algorithm as a parameter, which is generated by taking random samples of the population. The unified storage of population states can support more methods of population migration.

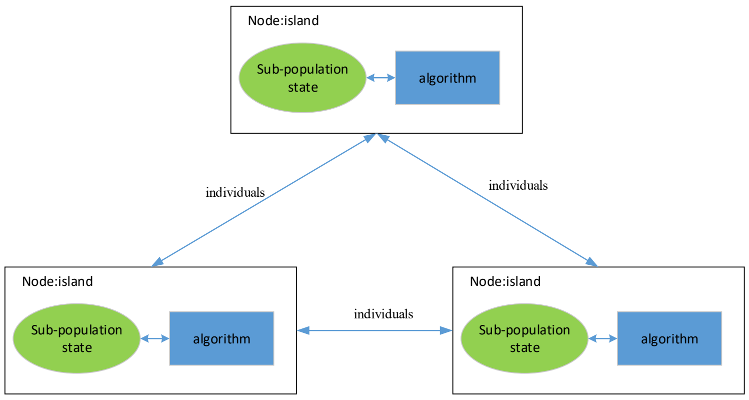


Figure 6. Communication between nodes in the classical island model.

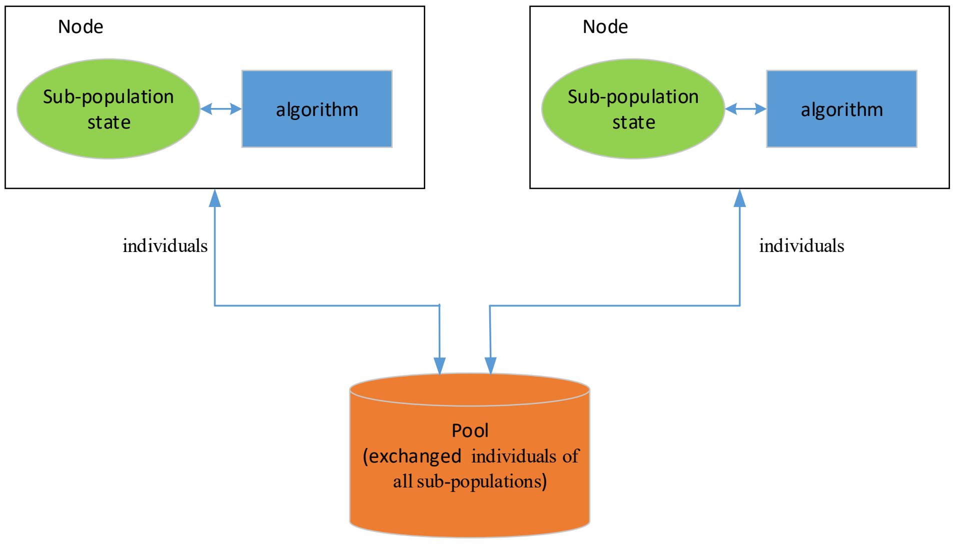


Figure 7. Communication between nodes in the Pool model.

2.6. Parallel Propulsion Mode

The parallel propulsion strategies of multiple nodes are coupled and affect each other. Common parallel propulsion modes include synchronous sequential propulsion, synchronous parallel propulsion and asynchronous parallel propulsion, as shown in Figure 9.

In synchronous sequential propulsion, as shown in Figure 9a, each task node of the system is executed in accordance with the specified order. During the whole progress, only one task node can be executed, represented as the colored progress bar shown in Figure 9a, while other task nodes are in the state of waiting, represented as the grey progress bar shown in Figure 9a. The logic of this propulsion mode is simple without complex synchronization mechanisms. Since the tasks are executed one by one and the result of the previous task can be sent to the next task as a parameter, the entire progress has low traffic. However, this mode is suitable for tasks nodes that are sequentially dependent. Otherwise, it will cause unnecessary time consumption and worse operation efficiency because every task node has large amounts of idle waiting time.

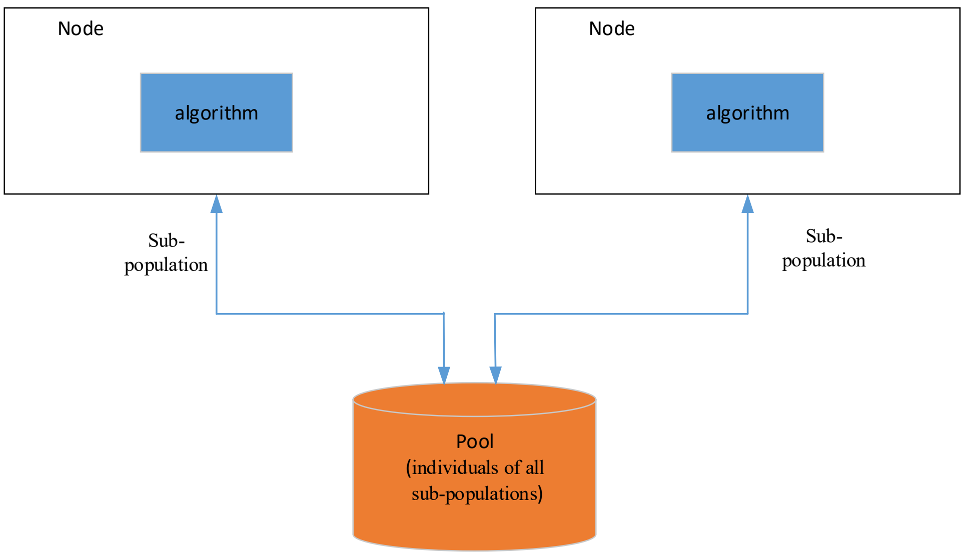


Figure 8. Communication between nodes in improved Pool model.

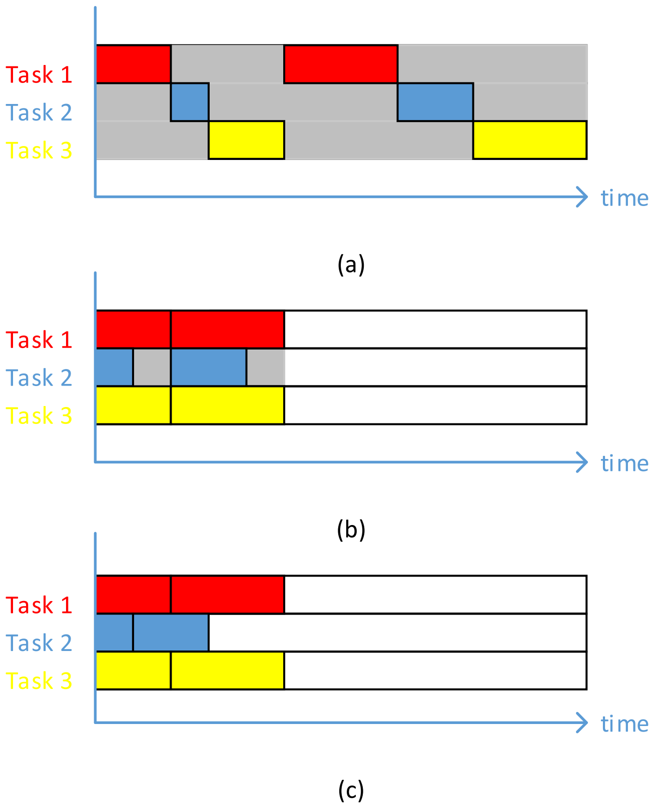


Figure 9. Parallel propulsion modes: (**a**) synchronous sequential propulsion; (**b**) synchronous parallel propulsion; (**c**) asynchronous parallel propulsion.

In synchronous parallel propulsion, as shown in Figure 9b, all task nodes are executed at the same time. Due to the different computing capacity of each task node, the propulsion speeds are different. The task nodes with faster propulsion speeds need to spend certain amount of time waiting to maintain synchronization with other task nodes. In this mode, all tasks are in the state of synchronous parallel operation, so the solving efficiency is greatly improved compared with synchronous sequential propulsion. Even if a task node is blocked during a stage, other task nodes can still be executed. It is obvious that there needs to be a synchronization mechanism to keep the system robust. This mode is suitable for task nodes that need to communicate with each other regularly.

In Figure 9c, all task nodes are executed in parallel at their own propulsion speeds without any waiting time. This mode is more suitable for task nodes with high independence. Otherwise, communication data between nodes will be chaotic.

3. The Proposed Multi-Population Parallel Wolf Pack Algorithm (MPPWPA)

According to the analysis of the WPA and the multi-population optimization method, this section proposes a novel multi-population parallel Wolf Pack Algorithm (MPPWPA) for task assignment modelling with high-dimensional solution space.

3.1. Elite-Mass Population Distribution

In order to make full use of excellent individuals to accelerate convergence, a virtual sub-population, called the elite sub-population, is constructed on the upper layer of the existing sub-populations, which are defined as mass sub-populations. The elite population has the same size as any mass sub-population and collects the best individuals from the total population. The composition of the elite sub-population does not change until the mass sub-populations migrate. According to the approximate average division proposed in Section 3.3, individuals of elite sub-population will be divided approximately equally to all mass sub-populations. In every iteration, the elite sub-population consisting of these individuals is optimized first, then the mass sub-populations, including these individuals, are optimized. This is similar to the process in which a company centrally trains leaders of departments and returns them to their respective positions to direct the departments. The high-quality individuals, after quadratic optimization, are of great significance for accelerating convergence. We define such a population distribution as an elite-mass population distribution and the optimization process is shown in Figure 10.

3.2. Pretreatment of the Population

In order to enhance the exploration of the search space, the population is pretreated before population division. The basis of the pretreatment is to double the population size by adding variants of some individuals in the original population and to generate new individuals randomly. Individuals with the same solution are called redundant individuals. If an individual cannot obtain a better solution after multiple iterations and it does not contribute to the optimization, we define this individual as aging. In the optimization process, too many redundant individuals aging will lead to premature convergence. During the pretreatment, the redundant individuals of the population are removed periodically to prevent premature convergence. The original population is expressed as $Pop = \{wolf_i | i = 1, 2, \cdots, N\}$, where N is the size of population. The steps of pretreatment are as follows:

Step 1: Sort the Pop in ascending order;

Step 2: The first $\tau \cdot N$ individuals from the Pop form a mutant population expressed as $Pop_m = \{wolf_i | i = 1, 2, \cdots, N_m\}$, where $N_m = \tau \cdot N$ is the size of mutant population, and $\tau \in [0, 1]$ is the mutation ratio. Individuals in Pop_m undergo minor variation, which is similar to the walking behavior of WPA.

Step 3: Build a temporary population $Pop_t = Pop + Pop_m$, whose population size is $N_t = N + N_m$. If the time interval (defined as ΔI) for redundancy removal is met, remove the redundant individuals of Pop_t and the population size decreases to N_t';

Step 4: Randomly generate a new population Pop_r with the size of $(2 \cdot N - N_t)$ or $(2 \cdot N - N_t')$;

Step 5 Coalesce all populations $Pop_{total} = Pop + Pop_m + Pop_r$. Sort the Pop_{total} in ascending order and the first N individuals form the new Pop that replaces the original Pop.

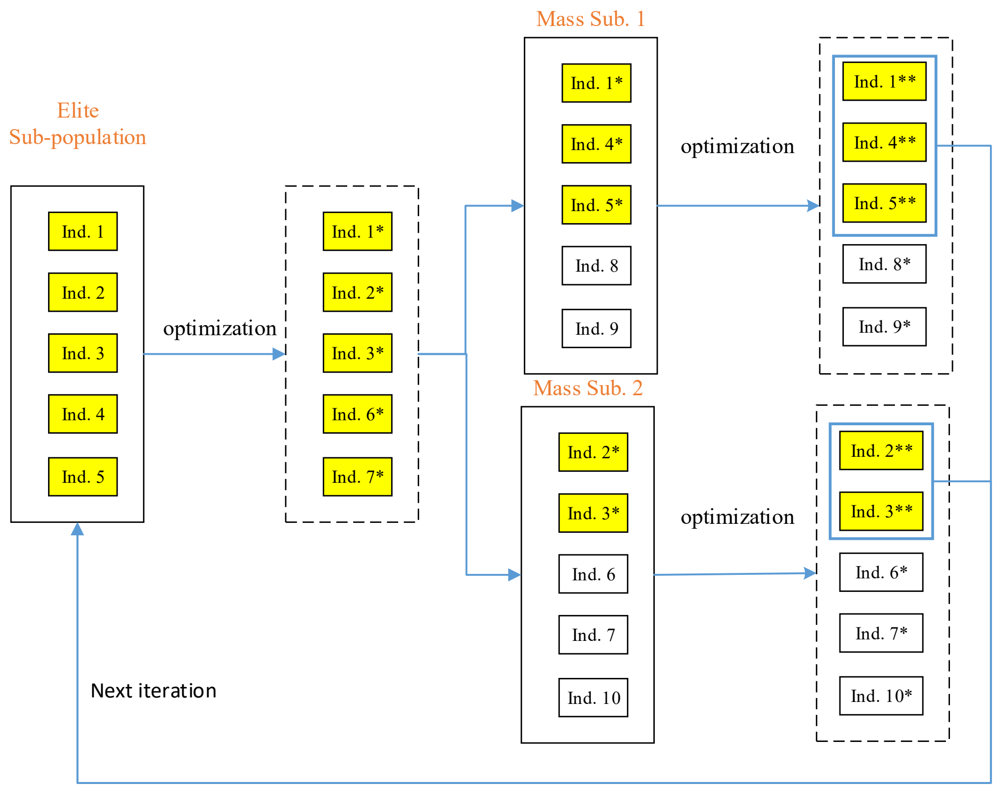


Figure 10. Optimization process of elite-mass population distribution.

3.3. *Approximate Average Division Method*

As is known, the optimization performance of heuristic algorithms is closely related to population size. Within a certain range, the more individuals the population includes, the better the optimization performance will be [45]. In essence, multi-population optimization reduces the population size of an optimization unit and makes up for the performance loss by absorbing high-quality individuals from other optimization units. The contribution of a sub-population to global optimization depends on the quality of its best individuals, because high-quality individuals can affect the exploration of global optimal solutions. In MPPWPA, all sub-populations are optimized by the WPA, so the size of each sub-population is the same. In order to equalize the quality of all sub-populations, an approximate average division method is proposed. The number of sub-populations is set as Num. All individuals are sorted in ascending order and every Num individuals constitute a segment, then the individuals in each segment are randomly assigned to mass sub-populations. In this way, the population can be divided almost uniformly.

Detailed steps are shown in Figure 11.

Step 1: The population Pop is sorted in ascending order;

Step 2: The first $\frac{N}{Num}$ individuals are labeled as members of the elite sub-population;

Step 3: Pop is divided into multiple segments with the same length Num;

Step 4: Num individuals in each segment are randomly assigned to Num mass sub-populations.

The proposed pretreatment and approximate average division method maximize the competitiveness of each sub-population, and thus, they can be used in population migration. Mass sub-populations to are coalesced to Pop, and then pretreated and re-divided. The problems of migration rate and individual selection in population migration are also avoided successfully.

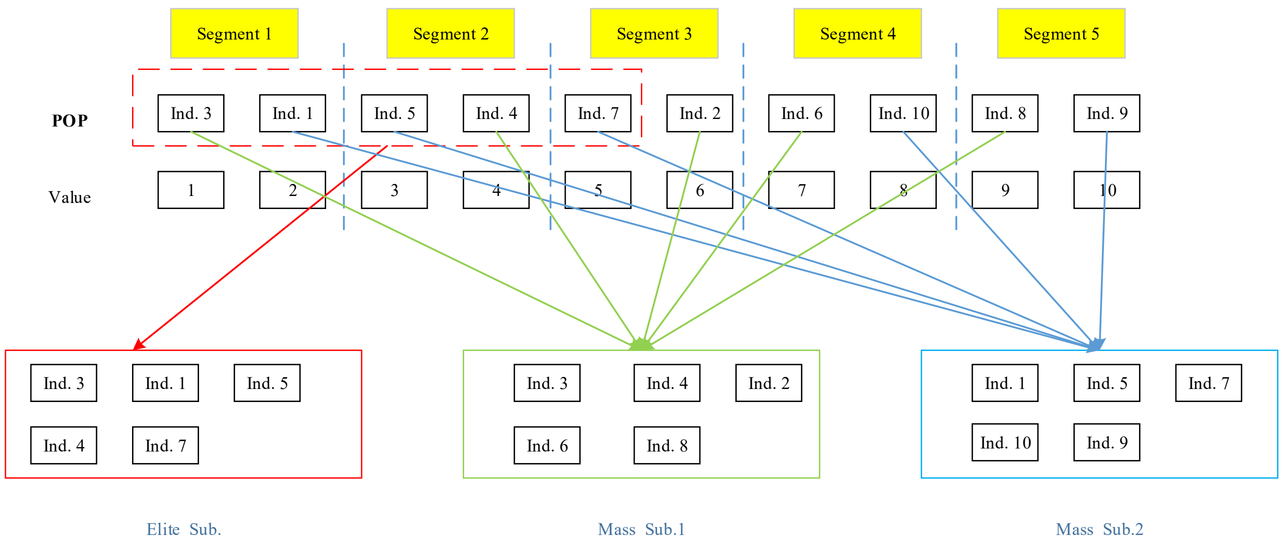


Figure 11. The process of approximate average division.

It should be noted that the migration probability (P_m) is used to determine whether to carry out population migration after each iteration. If $rand() < P_m$, all mass sub-populations fuse and the population will be re-divided; otherwise, all sub-populations continue to be optimized. The flow of the MPPWPA is shown in Figure 12.

3.4. System for MPPWPA

Introducing the improved Pool model in Figure 8, the system for MPPWPA is proposed, as Figure 13 shows. The system consists of one managing process and multiple working processes that actually perform algorithms.

The managing process has four main responsibilities: (1) Control all processes. (2) Manage all parameters. (3) Pretreat and divide the population in stages of initialization and population migration. (4) Execute the optimization of the elite sub-population.

Each algorithm is encapsulated into a module, whose inputs are parameters of the algorithm and the state of population, and the output is the state of the sub-population after iteration. Each working process only needs to call and run the required algorithm module, saving a lot of computational costs for processing data.

3.5. Parallel Propulsion Mode for MPPWPA

By analyzing the features and the applicable scope of three modes, combined with the flow of the multi-population parallel optimization algorithm, a synchronous parallel propulsion mode is introduced to MPPWPA, as shown in Figure 14.

In the synchronous parallel propulsion mode, the managing process is performed first, which then generates three working processes with its result. After that, the managing process waits for all working processes to finish and receives the results from the working processes to prepare for the next stage. Three working processes are performed with the synchronous parallel propulsion because their propulsion speeds are similar and their results need to be handled by the managing process at the same time.

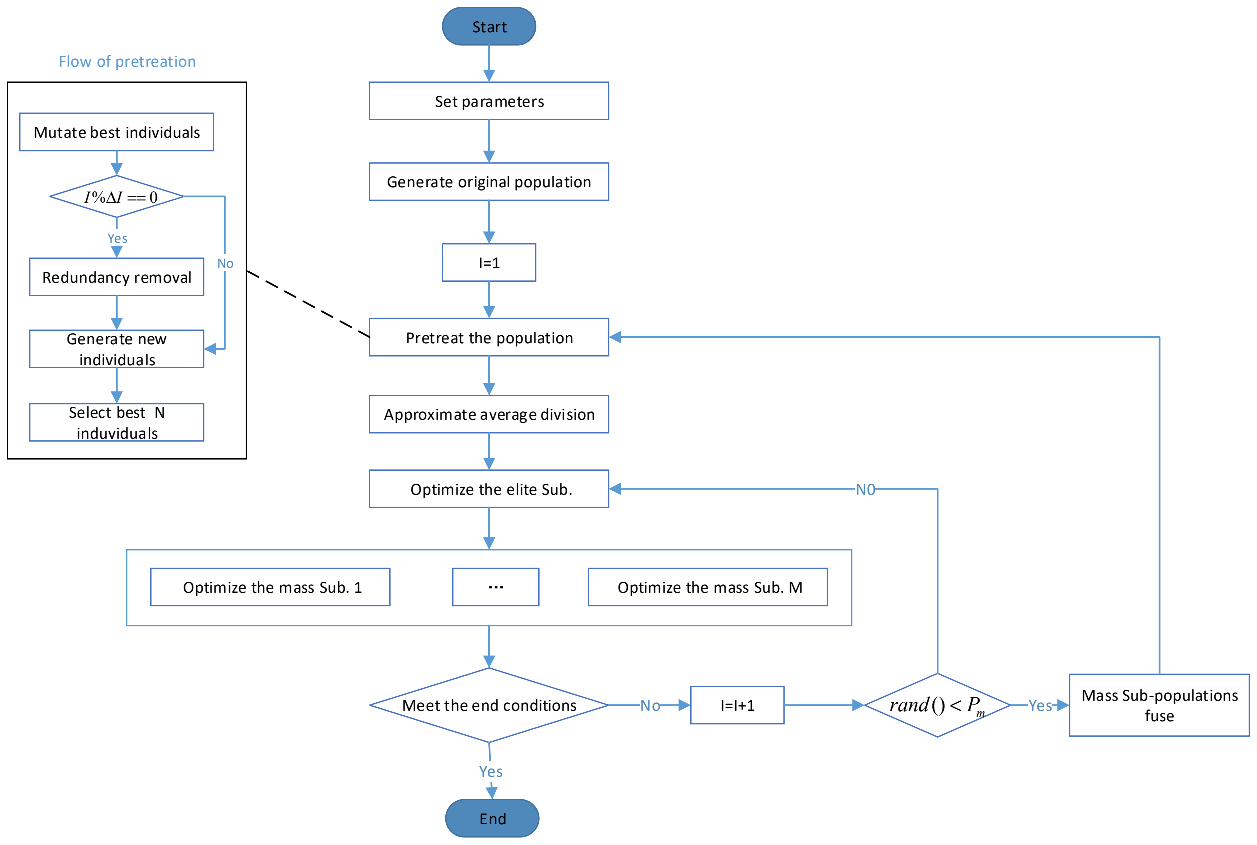


Figure 12. The flow of the MPPWPA.

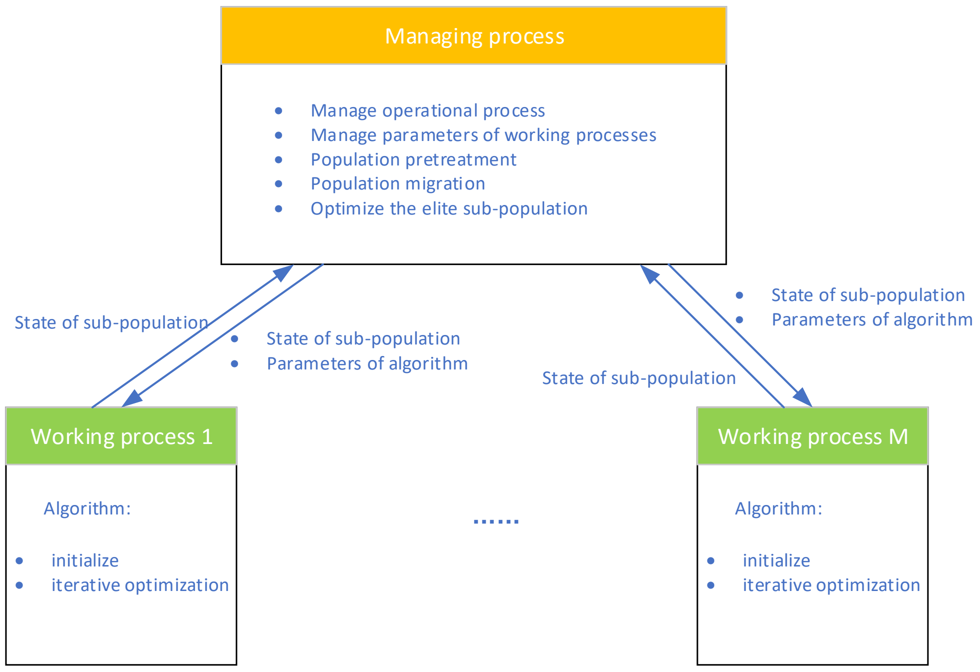


Figure 13. System for MPPWPA.

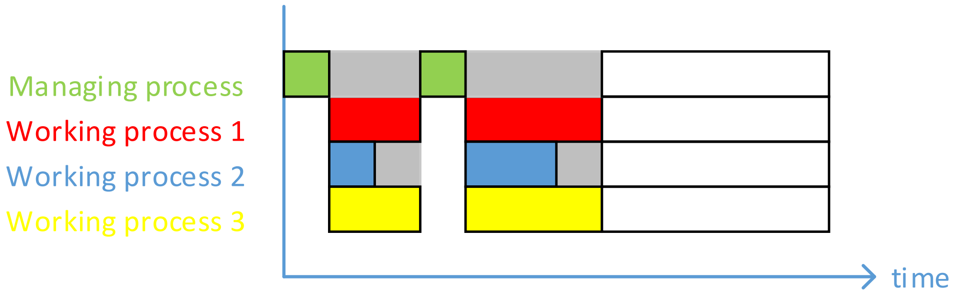


Figure 14. Parallel propulsion for MPPWPA.

4. Experiments of Task Assignment for UAV Swarm Using MPPWPA

To verify the effectiveness of the proposed MPPWPA, we chose the basic WPA (Figure 3), PSO [2,30], GA [23] and ABC algorithm [29] for comparison. The PSO and GA are widely used heuristic algorithms while the basic WPA and ABC algorithm are relatively new but have received much attention in recent years.

The algorithms were coded in Python, and all simulations were run on a 2.50-GHz computer with a quad-core Intel i5 CPU and 4 GB of RAM. A Monte Carlo study, consisting of 20 runs, is used in this section to compare the performance of the PSO, GA, ABC, WPA and MPPWPA with different parameters.

4.1. Preparation for Simulation Experiments

A Monte Carlo study, consisting of 20 independent runs, was used to compare the performance of the PSO, GA, ABC, WPA and MPPWPA with different parameters for the cost function of Equation (6) where $\omega_1 = \omega_2 = 0.5$.

Three scenarios were: five UAVs against eight targets, 20 UAVs against 30 targets and 100 UAVs against 150 targets. The dimensions of the variable in the three scenarios were 8, 30 and 150 respectively. The initial locations of the UAVs and targets were generated randomly and projected onto a two-dimensional plane, as shown in Figure 15.

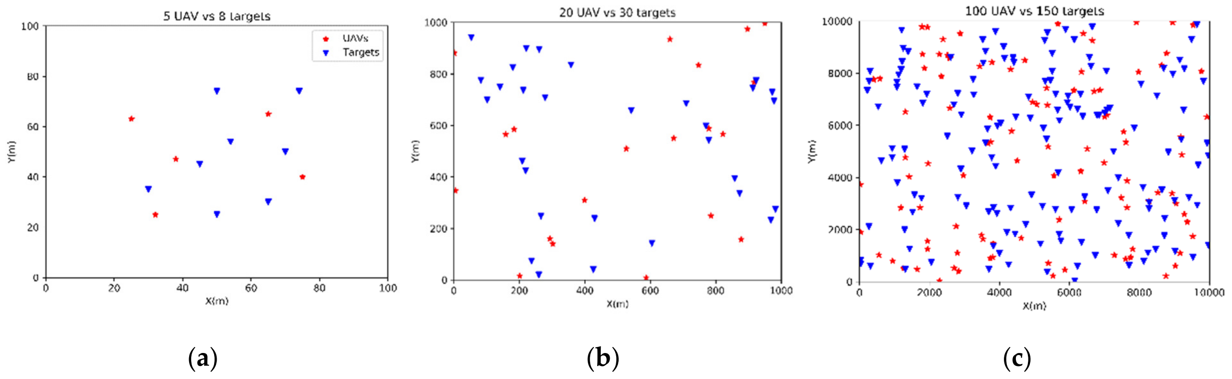


Figure 15. Initial positions of UAV swarm and targets. (**a**) Scenario 1: 5 UAVs and 8 targets were randomly distributed in a space of 100 × 100 m. (**b**) Scenario 2: 20 UAVs and 30 targets were randomly distributed in a space of 1000 × 1000 m. (**c**) Scenario 3: 100 UAVs and 150 targets were randomly distributed in a space of 10,000 × 10,000 m.

The parameters of the WPA for different scenarios were set as shown in Table 2.

Table 2. Parameters of WPA in three scenarios.

Parameters	Scenario 1	Scenario 2	Scenario 3
$I_{\max}$	200	400	1000
N	160	160	160
$step_a$	2	2	2
$step_b$	4	14	70
$step_c$	1	1	1
$T_{\max}$	10	10	10
d_{near}	2	14	70
α	4	4	4
β	5	5	5
$h_{\min}$	1	1	1
$h_{\max}$	5	5	5

The control parameters of the MPPWPA are shown in Table 3. This research mainly studied the influence of Num and P_m on the optimization performance. Other parameters related to WPA were the same with those in Table 2.

Table 3. Control parameters of MPPWPA.

Parameters	Meaning	Values
Num	The number of mass sub-populations	2, 4, 8, 16
P_m	Probability of migration	1, 0.8, 0.6, 0.4, 0.2, 0
τ	Mutation ratio	20%
ΔI	Interval of redundancy removal	2 iterations in scenario 15 iterations in scenario 25 iterations in scenario 3

4.2. The Results and Analyses of Simulation Experiments

4.2.1. Simulation Results in Scenario 1

Firstly, we set $P_m = 1$ and compared the results of MPPWPA with different Num. Since the dimension of the solution space was low, the exact solution of the problem could be obtained using the exhaustive method in 756 s, and the minimum value of the objective function was 28.71. The results of several heuristic algorithms involved in the experiment are shown in Figure 16 and the statistics for the results are shown in Table 4 where the best results are marked in boldface. For four basic algorithms, it is clearly shown that WPA performed better than PSO both in terms of accuracy and convergence speed, performed better than the GA in terms of accuracy, but performed worse than the GA in terms of convergence speed, and the WPA performed worse than the ABC both in terms of accuracy and convergence speed. As an improvement of the WPA, the MPPWPA was improved in terms of both accuracy and convergence speed. For the task assignment problem with an eight-dimensional solution space, when Num ranged from 2 to 8, the optimization performance gradually improved, but the performance did not continue to improve when Num was 16. In addition, we found that the larger the Num was, the more stable the convergence value would be. The results indicated that increasing the number of sub-populations is conductive to exploiting the optimal solution and decreasing the convergence time.

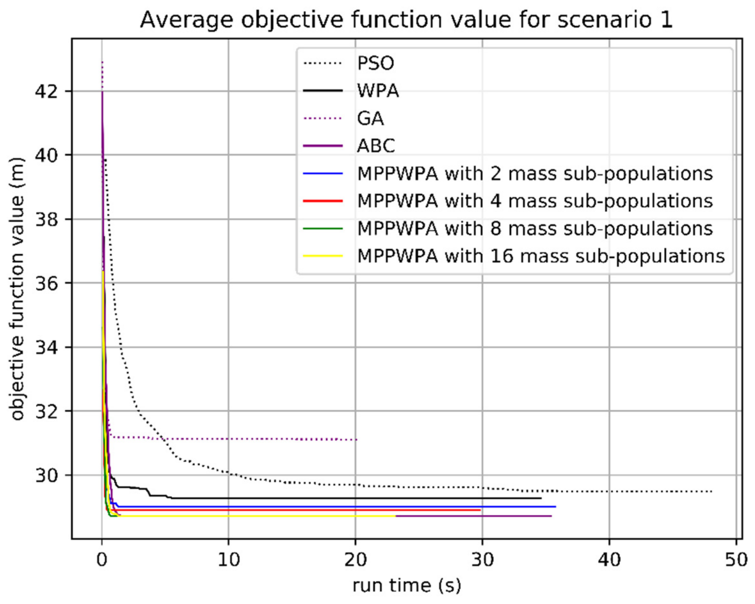
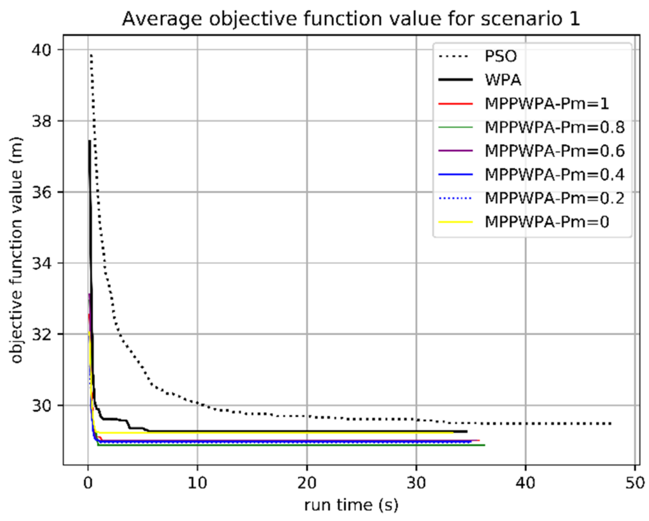


Figure 16. Average objective function values with different *Num* in scenario 1.

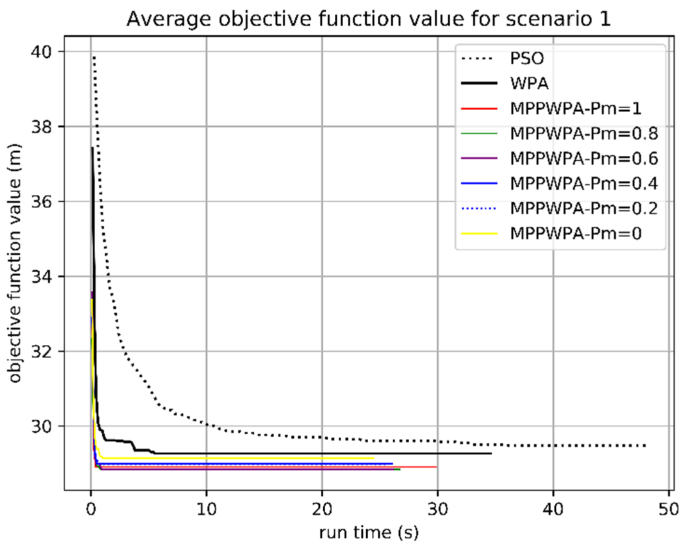
Table 4. Objective function value and convergence time with different *Num* in scenario 1.

Algorithm	***Num***	**Objective Function Value (m)**		**Convergence Time (s)**	
		Ave.	**Std.**	**Ave.**	**Std.**
PSO	-	29.49	0.78	10.40	8.44
GA	-	31.10	1.58	1.64	3.23
ABC	-	**28.71**	**0**	1.30	0.24
WPA	-	29.65	1.23	2.22	3.76
MPPWPA	2	29.01	0.43	1.66	0.25
	4	28.90	1.01	1.28	0.23
	8	**28.71**	**0**	**1.11**	0.17
	16	**28.71**	**0**	1.27	0.24

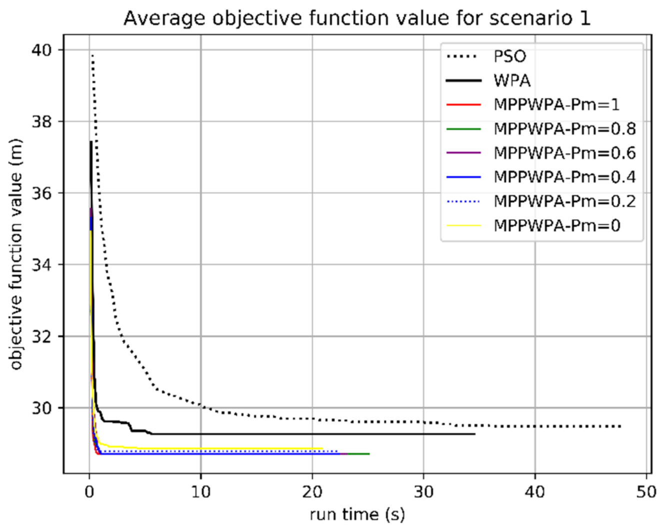
Then, we compared the influence of different P_m values on the optimization performance. The results are shown in Figure 17 and the statistics for the results are shown in Tables 5 and 6. When *Num* was 2, 4 or 8, a larger migration probability was useful to fully search the solution space to obtain the optimal value. Interestingly, when $Num = 16$, a stable optimal value could be obtained without population migration and it could be obtained faster with a small P_m or a large P_m. Population migration with medium frequency may have degraded the optimization performance. This indicated that the quadratic optimization of elite-mass population distribution was an invisible population migration. It also suggested that population migration was not always beneficial to the development of sub-populations. In particular, when the population size was very small (10 individuals when $Num = 16$), the adverse effects of population migration may not have been eliminated by iteration.



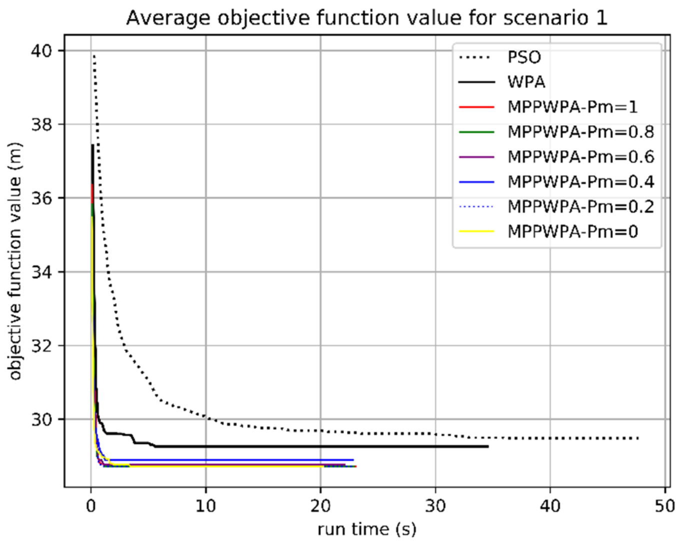
(**a**)



(**b**)



(**c**)



(**d**)

Figure 17. Average objective function values with different Num and P_m in scenario 1: (**a**) $Num = 2$; (**b**) $Num = 4$; (**c**) $Num = 8$; (**d**) $Num = 16$.

Table 5. Objective function value with different Num and P_m in scenario 1.

Num	**2**		**4**		**8**		**16**	
Pm	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**
1	29.01	0.45	28.90	1.01	**28.71**	**0**	**28.71**	**0**
0.8	**28.87**	0.78	28.85	1.00	**28.71**	**0**	**28.71**	**0**
0.6	29.01	0.81	**28.83**	1.02	**28.71**	**0**	28.86	0.35
0.4	28.97	**1.14**	28.98	0.82	**28.71**	**0**	28.89	0.58
0.2	28.95	0.82	28.98	1.00	28.77	0.77	**28.71**	**0**
0	29.59	1.5	29.20	**0.8**	28.98	0.33	**28.71**	**0**

Table 6. Convergence time with different Num and P_m in Scenario 1.

Num	**2**		**4**		**8**		**16**	
Pm	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**
1	1.66	0.25	1.28	0.23	**1.11**	0.17	1.27	0.24
0.8	1.62	0.18	**1.16**	0.13	1.27	0.24	1.28	0.36
0.6	1.56	0.14	**1.16**	0.15	1.27	0.24	**1.24**	0.26
0.4	**1.55**	0.22	1.26	0.29	1.27	0.30	1.40	0.34
0.2	1.59	0.20	1.28	0.74	1.23	0.34	1.30	0.22
0	1.57	0.17	1.21	0.22	1.16	0.37	1.47	0.22

4.2.2. Simulation Results in Scenario 2

We set $P_m = 0.8$ and compared the results of MPPWPA with different Num values. When the dimension of the solution space in scenario 2 was enlarged to 30, the exhaustive method failed. The results of heuristic algorithms involved in the experiment are shown in Figure 18 and the statistics for the results are shown in Table 7. The WPA clearly performed better than the PSO and GA and had a faster convergence speed compared to ABC. The proposed MPPWPA had obvious advantages both in terms of accuracy and convergence speed, compared with the four basic algorithms. For the task assignment problem with eight-dimensional solution space, the MPPWPA improved the optimization performance both in terms of accuracy and convergence speed. When Num ranged from 2 to 8, the optimization performance gradually improved, while the convergence speed declined when Num reached 16. This was because frequent population migration caused great disturbances to small-size sub-populations; it improved the exploitation but reduced the exploration. As shown in Figure 18, the MPPWPA had an excellent convergence speed in the early stage of iteration. This means that it could obtain better values in a short time compared with PSO, GA, ABC and WPA, which is of great significance in practical applications.

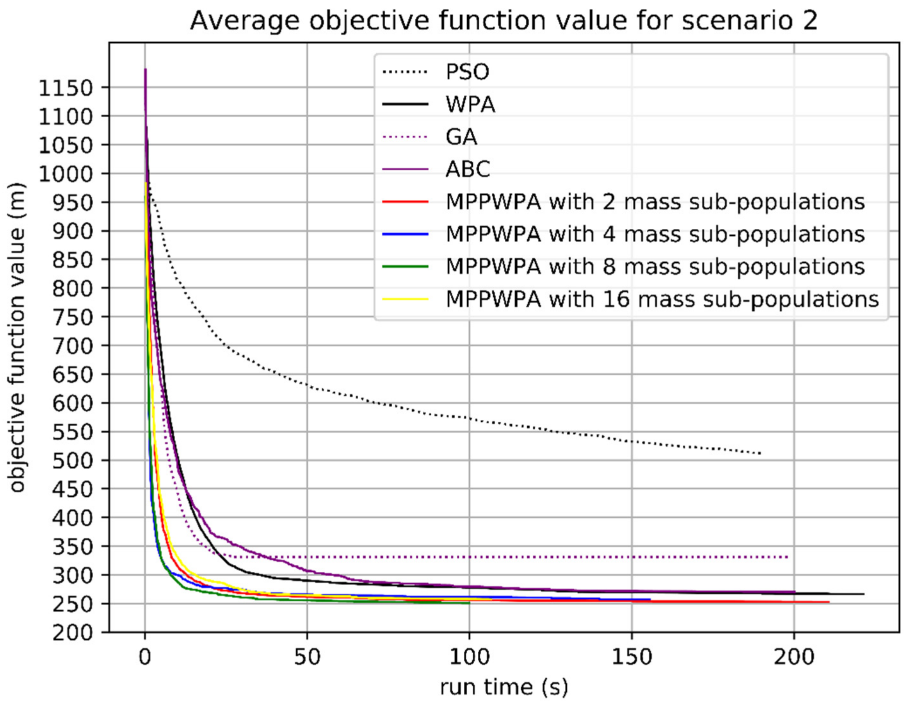


Figure 18. Average objective function values with different Num in scenario 2.

Table 7. Objective function value and convergence time with different *Num* in scenario 2.

Algorithm	***Num***	**Objective Function Value (m)**		**Convergence Time (s)**	
		Ave.	**Std.**	**Ave.**	**Std.**
PSO	-	511.8	37	174.73	47.78
GA	-	330.86	30.36	21.87	4.25
ABC	-	269.78	5.8	170.73	42.86
WPA	-	266.3	19.48	158.83	49.29
MPPWPA	2	252.6	15.1	109.2	47.0
	4	257.1	19.3	72.1	24.2
	8	**250.6**	13.6	**63.7**	22.5
	16	257.2	13.5	64.9	12.3

Then, we compared the influence of different P_m on the optimization performance. The results are shown in Figure 19 and the statistics for the results are shown in Tables 8 and 9. Compared with the results in scenario 1, population migration was necessary to improve the convergence speed and accuracy for the task assignment problem with 30-dimensional solution space. When $Num = 2$ and $Num = 4$, the convergence trends under different P_m were basically the same and there was no significant difference in optimization performance. When $Num = 8$, excluding 0 and 1, a large P_m contributed to ensuring accuracy, while a small P_m contributed to a shortening of convergence time. When $Num = 16$, the algorithm preferred a small P_m, but not 0. Compared with the WPA, the MPPWPA could obtain a better solution regardless of the value of P_m, which shows that the MPPWPA was robust to P_m.

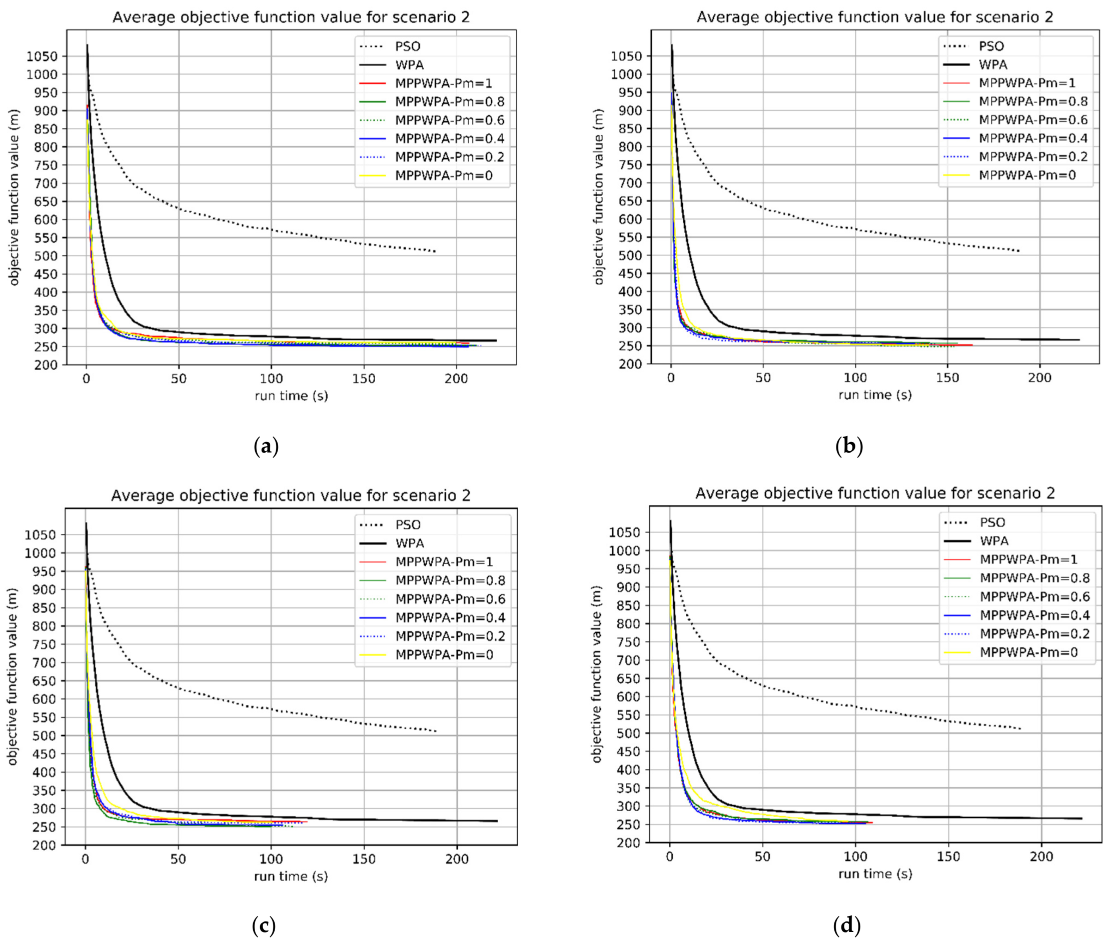


Figure 19. Average objective function values with different *Num* and P_m in scenario 2: (**a**) $Num = 2$; (**b**) $Num = 4$; (**c**) $Num = 8$; (**d**) $Num = 16$.

Table 8. Objective function value with different P_m in scenario 2.

Num	**2**		**4**		**8**		**16**	
Pm	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**
1	259.7	15.5	251.7	16.5	263.9	16.2	254.9	20.6
0.8	252.6	15.1	257.1	19.3	**250.6**	13.6	257.2	13.5
0.6	257.4	20.9	**248.27**	14.1	**250.6**	13.0	254.2	14.0
0.4	**249.5**	14.4	255.9	15.1	254.1	18.1	**251.9**	15.3
0.2	252.4	14.5	254.8	18.4	260.2	14.5	252.7	14.9
0	259.9	15.2	251.7	13.8	261.7	21.3	258.5	15.1

Table 9. Convergence time with different P_m in scenario 2.

Num	**2**		**4**		**8**		**16**	
Pm	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**	**Ave.**	**Std.**
1	108.5	41.2	70.3	36.1	64.1	19.7	65.2	15.4
0.8	109.2	47.0	72.1	24.2	63.7	22.5	64.9	12.3
0.6	**101.1**	37.9	68.3	21.1	69.2	28.8	57.1	10.0
0.4	109.0	29.8	**67.4**	27.9	**53.8**	8.4	47.1	6.9
0.2	110.5	40.87	78.1	23.5	54.4	16.8	**45.0**	9.8
0	110.1	42.9	80.8	17.5	66.7	8.13	65.7	1.4

In terms of convergence performance, the algorithm with 16 mass sub-populations performed better, as it could obtain a stable best solution in a shorter time, while the algorithm with two mass sub-populations needed more time to obtain a best solution. In terms of convergence trend, the algorithm with eight mass sub-populations performed better because it converged faster in the early stage, and thus, was found to be suitable for application requirements where a solution is needed in a specified short time such as 10 s or 20 s.

From Figure 19d, we can see that the optimization performance of the algorithm with 16 mass sub-populations was poor when $P_m = 0$, which was different from the performance in scenario 1. The reason is that the optimization abilities of the sub-populations with small sizes are insufficient when the solution space is 30-dimensional; thus population migration is needed to improve the exploration of the solution space. This phenomenon indicates that the population size of a heuristic optimization algorithm influences the optimization performance.

4.2.3. Simulation Results in Scenario 3

According to the analysis of the optimization performance in the first two scenarios, the proposed MPPWPA was effective in solving the task assignment problem and had good robustness to control parameters. In order to verify its effectiveness for problems with higher dimensional variables, we applied it in scenario 3, where the solution space was 150-dimensional. As shown in Figure 20 and Table 10, MPPWPA exhibited satisfying performance, especially in terms of convergence time, compared with the PSO, GA, ABC and WPA. Through the statistical data, it was found that the convergence time of the MPPWPA was 62% less than that of the WPA, which is of great significance to the research of UAV swarm task assignment.

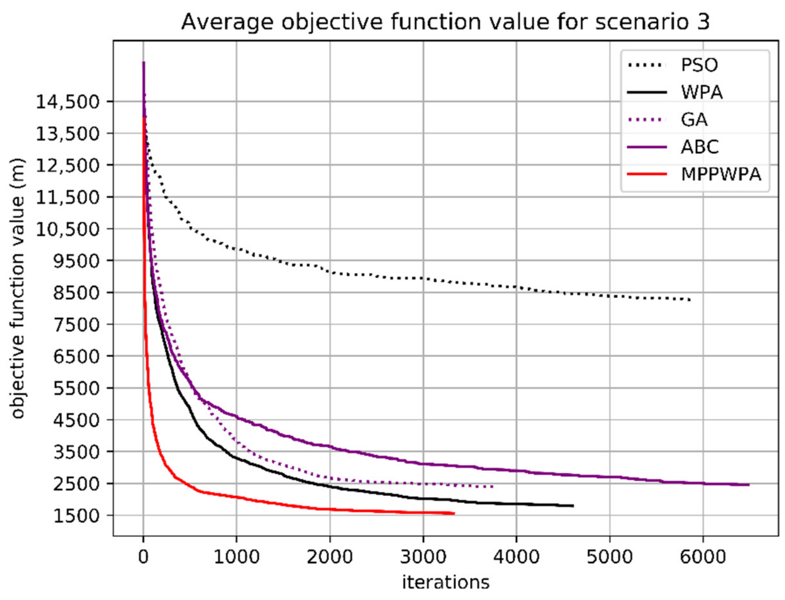


Figure 20. Average objective function values in scenario 3.

Table 10. Objective function value and convergence time in scenario 3.

Algorithm	**Objective Function Value**		**Convergence Time**	
	Ave.	**Std.**	**Ave.**	**Std.**
PSO	8268	218	5579	1464
GA	2390	53	2581	503
ABC	2455	134	4643	644
WPA	1789	90.04	3262	753
MPPWPA	**1552**	50.49	**1242**	371

5. Conclusions

With the aim of reducing the optimization time for the problem of UAV swarm task assignment, the multi-population parallel Wolf Pack Algorithm (MPPWPA) was proposed. According to building of the task assignment model and the analysis of the WPA, the multi-population optimization method, the communication structures and the parallel propulsion mode, this paper proposed the following methods: (a) an elite-mass population distribution to optimize better solutions twice; (b) a pretreatment of the population to increase population diversity; (c) an approximate average division method to equalize the quality of all sub-populations; (d) a parallel optimization structure to realize the MPPWPA.

Firstly, the proposed algorithm was verified as reasonable by means of simulation of a problem with an eight-dimensional solution space, and it showed good performance in for the problem with high dimensional variables. Then, we studied the influence of its control parameters: Num and P_m. The results of simulation experiments indicated that (1) when the dimension of the solution space was low, MPPWPA with different Num and P_m could obtain the best solution; (2) when the dimension of the solution space was high, Num and P_m had little effect on the convergence time, but they did not change the overall effectiveness of the algorithm; (3) when the dimension of the solution space was very high, the MPPWPA had satisfying performance, especially in terms of convergence time; (4) in order to obtain a better optimization performance, the size of sub-population needed to match the dimensions of the variable.

The parallel computing in this paper was implemented on a single computer, and future work will focus on the feasibility of multi-computer distributed parallel computing or cloud computing to solve large-scale task assignment. In addition, we set the P_m as a constant in this work; we will study variable migration probabilities in future work. In the longer term, we also need to consider how to handle moving targets. Whether to track the target according to the original assignment result or to re-assign the task is a question that is worthy of further study.

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. and J.W.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and Y.M.; visualization, Y.L.; supervision, Y.M. and J.W.; project administration, Y.M.; funding acquisition, Y.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Science and Technology on Electro-optic Control Laboratory and the Aviation Science Fund, grant number 20175151024.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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. Wu, H.; Li, H.; Xiao, R.; Liu, J. Modeling and simulation of dynamic ant colony's labor division for task allocation of UAV swarm. *Phys. A Stat. Mech. Appl.* **2018**, *491*, 127–141. [CrossRef]
2. 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.
3. Samuel, R. Routing and scheduling of vehicles and crews: The state of the art. *Comput. Oper. Res.* **1983**, *10*, 63–211.
4. Mazzeo, S.; Loiseau, I. An ant colony algorithm for the capacitated vehicle routing. *Electron. Notes Discret. Math.* **2004**, *18*, 181–186. [CrossRef]
5. Solomon, M.M. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Oper. Res.* **1987**, *32*, 254–265. [CrossRef]
6. Dror, M.; Trudeau, P. Savings by split delivery routing. *Transp. Sci.* **1989**, *23*, 141–145. [CrossRef]
7. Zhang, H.; Ge, H.; Yang, J.; Tong, Y. Review of vehicle routing problems: Models, classification and solving algorithms. *Arch. Comput. Methods Eng.* **2021**, *28*, 1–27. [CrossRef]
8. Min, H. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transp. Res. A Gen.* **1989**, *23*, 377–386. [CrossRef]
9. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
10. Shima, T.; Rasmussen, S.J. Tree search algorithm for assigning cooperating UAVs to multiple tasks. *Int. J. Robust Nonlinear Control* **2008**, *18*, 135–153.
11. Alighanbari, M.; How, J. Cooperative task assignment of unmanned aerial vehicles in adversarial environments. In Proceedings of the American Control Conference, Portland, OR, USA, 8–15 June 2005; pp. 4661–4666.
12. 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.
13. 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.
14. 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, 1 January 2009; pp. 457–464.
15. Mao, H.; Tian, S.; Chao, A. *UAV Mission Planning*; National Defense Industry Press: Beijing, China, 2015. (In Chinese)
16. Di, B.; Zhou, R.; Wu, J. Distributed coordinated heterogeneous task allocation for unmanned aerial vehicles. *Control Decis.* **2013**, *28*, 274–278.
17. Oh, G.; Kim, Y.; Ahn, J.; Choi, H. Market-based task assignment for cooperative timing missions in dynamic environments. *J. Intell. Robot. Syst.* **2017**, *87*, 97–123. [CrossRef]
18. Zhang, J.; Wang, Y.; Li, F.; Zhou, T. Dynamic task assignment problem of multi-agent. *Electron. Technol. Softw. Eng.* **2018**, *18*, 255–258.
19. Brunet, L.; Choi, H.; How, J. Consensus-based auction approaches for decentralized task assignment. In Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI, USA, 18–21 August 2008.
20. Yu, X.; Guo, J.; Zheng, H. Extended-CBBA-based task allocation algorithm for on-orbit assembly spacecraft. *Unmanned Syst. Technol.* **2019**, *4*, 46–53.

21. 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]
22. 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, 8–15 June 2005; pp. 2989–2994.
23. Xiao, K.; Lu, J.; Nie, Y.; Ma, L.; Wang, X.; Wang, G. A benchmark for multi-UAV task assignment of an extended team orienteering problem. *arXiv* **2020**, arXiv:2009.00363v1.
24. 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.
25. Bousad, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [CrossRef]
26. Li, X.; Shao, Z.; Qian, J. An optimizing method based on autonomous animats: Fish-swarm algorithm. *Syst. Eng.-Theory Pract.* **2002**, *22*, 32–38.
27. Passino, K. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Syst. Mag.* **2002**, *22*, 52–67.
28. Eusuff, M.M.; Lansey, K.E. Optimization of water distribution network design using the shuffled frog leaping algorithm. *J. Water Resour. Plan. Manag.* **2003**, *129*, 210–225. [CrossRef]
29. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
30. Wu, H.; Zhang, F.; Wu, L. New swarm intelligence algorithm—Wolf pack algorithm. *Syst. Eng. Electron.* **2013**, *35*, 2430–2437.
31. Wu, H.; Zhang, F. An uncultivated wolf pack algorithm for high dimensional functions and its application in parameters optimization of PID controller. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1477–1482.
32. Gao, C.; Yu, X.; Zhu, Y. Optimization of hydraulic turbine governor parameters based on WPA. In Proceedings of the 2018 IOP Conference Series: Earth and Environmental Science, Chongqing, China, 25–26 November 2017; Volume 108, p. 052011.
33. Zhang, X. Short-term load forecasting for electric bus charging stations based on fuzzy clustering and least squares support vector machine optimized by wolf pack algorithm. *Energies* **2018**, *11*, 1449. [CrossRef]
34. Zhuang, H.; Jiang, X. A wolf pack algorithm for active and reactive power coordinated optimization in active distribution network. In Proceedings of the 2016 IOP Conference Series: Earth and Environmental Science, Beijing, China, 19–22 August 2016; Volume 40, pp. 1–9.
35. Ding, C.-Q.; Li, C.-P.; Liu, B.; Jiang, X.-P.; Tang, Y.-H.; Sun, H.; Zhou, W. Multi-objective congestion dispatch of active distribution network based on source-load coordination. *Autom. Electr. Power Syst.* **2017**, *41*, 88–95.
36. Menassel, R.; Nini, B.; Mekhaznia, T. An improved fractal image compression using wolf pack algorithm. *J. Exp. Theor. Artif. Intell.* **2018**, *30*, 429–439. [CrossRef]
37. Feng, X.; Hu, K.; Lou, X. Infrared and visible image fusion based on the total variational model and adaptive wolf pack algorithm. *IEEE Access* **2020**, *8*, 2348–2361. [CrossRef]
38. Wu, H.; Zhang, F.; Zhan, R.; Wang, S.; Zhang, C. A binary wolf pack algorithm for solving 0–1 knapsack problem. *Syst. Eng. Electron.* **2014**, *36*, 1660–1667.
39. Guo, L.; Liu, S. An improved binary wolf pack algorithm based on adaptive step length and improved update strategy for 0–1 knapsack problems. In Proceedings of the Communications in Computer and Information Science, Changsha, China, 22–24 September 2017.
40. Li, H.; Wu, H. An oppositional wolf pack algorithm for parameter identification of the chaotic systems. *Optik* **2016**, *127*, 9853–9864. [CrossRef]
41. Xian, S.; Li, T.; Cheng, Y. A novel fuzzy time series forecasting model based on the hybrid wolf pack algorithm and ordered weighted averaging aggregation operator. *Int. J. Fuzzy Syst.* **2020**, *22*, 1832–1850. [CrossRef]
42. Lu, Y.; Ma, Y.; Wang, J.; Han, L. Task assignment of UAV swarm based on wolf pack algorithm. *Appl. Sci.* **2020**, *10*, 8335. [CrossRef]
43. Yu, X.; Yu, H.; Liu, Y.; Xiao, R. A clustering routing algorithm based on wolf pack algorithm for heterogeneous wireless sensor networks. *Comput. Netw.* **2020**, *167*, 106994.
44. Chen, Y.; Mei, Y.; Yu, J.; Su, X.; Xu, N. Three-dimensional unmanned aerial vehicle path planning using modified wolf pack search algorithm. *Neurocomputing* **2017**, *266*, 445–457.
45. Jiao, L.; Liu, J.; Zhong, W. *Co-Evolutionary Computing and Multi-Agent Systems*; SciencePress: Beijing, China, 2006.
46. Zhang, Y.; Zhang, H. Dynamic scheduling of blocking flow-shop based on multi-population ACO algorithm. *Int. J. Simul. Model.* **2020**, *19*, 529–539. [CrossRef]
47. Park, J.; Park, M.W.; Kim, D.W.; Lee, J. Multi-population genetic algorithm for multilabel feature selection based on label complementary communication. *Entropy* **2020**, *22*, 876. [CrossRef]
48. Chen, L.; Li, Q.; Zhao, X.; Fang, Z.; Peng, F.; Wang, J. Multi-population coevolutionary dynamic multi-objective particle swarm optimization algorithm for power control based on improved crowding distance archive management in CRNs. *Comput. Commun.* **2019**, *145*, 146–160. [CrossRef]
49. Digalakis, J.G.; Margaritis, K.G. A multipopulation cultural algorithm for the electrical generator scheduling problem. *Math. Comput. Simul.* **2002**, *60*, 293–301. [CrossRef]

50. Yang, B.; Wang, S.; Cheng, Q.; Jin, T. Scheduling of field service resources in cloud manufacturing based on multi-population competitive-cooperative GWO. *Comput. Ind. Eng.* **2021**, *154*, 107104. [CrossRef]
51. Turky, A.; Sabar, N.R.; Song, A. A multi-population memetic algorithm for dynamic shortest path routing in mobile ad-hoc networks. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July 2016; pp. 4119–4126.
52. Zhang, W.; Wen, J.B.; Zhu, Y.C.; Hu, Y. Multi-objective scheduling simulation of flexible job-shop based on multi-population genetic algorithm. *Int. J. Simul. Model.* **2017**, *16*, 313–321. [CrossRef]
53. Arantes, M.; Arantes, J.; Toledo, C.; Williams, B. A hybrid multi-population genetic algorithm for UAV path planning. In Proceedings of the Genetic and Evolutionary Computation Conference, Denver, CO, USA, 20–24 July 2016; pp. 853–860.
54. Hao, K.; Zhao, J.; Yu, K.; Li, C.; Wang, C. Path planning of mobile robots based on a multi-population migration genetic algorithm. *Sensors* **2020**, *20*, 5873. [CrossRef]
55. Li, X.; Ma, S.; Wang, Y. Multi-population based ensemble mutation method for single objective bilevel optimization problem. *IEEE Access* **2016**, *4*, 7262–7274. [CrossRef]
56. Wang, X.; Chen, H.; Heidari, A.A.; Zhang, X.; Xu, J.; Xu, Y.; Huang, H. Multi-population following behavior-driven fruit fly optimization: A Markov chain convergence proof and comprehensive analysis. *Knowl.-Based Syst.* **2020**, *210*, 106437. [CrossRef]
57. Yoshida, H.; Fukuyama, Y. Parallel multi-population differential evolutionary particle swarm optimization for voltage and reactive power control in electric power systems. In Proceedings of the 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 19–22 September 2017.
58. Nseef, S.K.; Abdullah, S.; Turky, A.; Kendall, G. An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems. *Knowl.-Based Syst.* **2016**, *104*, 14–23. [CrossRef]
59. Merelo, J.J.; Mora, A.M.; Fernandes, C.M.; Esparcia-Alcazar, A.I.; Laredo, J.L. Pool vs. island based evolutionary algorithms: An initial exploration. In Proceedings of the 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Victoria, BC, Canada, 12–14 November 2012; pp. 19–24.
60. García-Valdez, M.; Trujillo, L.; Merelo, J.J.; de Vega, F.F.; Olague, G. The EvoSpace Model for Pool-Based Evolutionary Algorithms. *J. Grid Comput.* **2015**, *13*, 329–349. [CrossRef]