Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 5518927, 9 pages
https://doi.org/10.1155/2021/5518927

Research Article

UAV Task Allocation Based on Clone Selection Algorithm

Xiaopan Zhang[1] and Xingjun Chen[2]

[1]*Resources and Environmental Engineering School of Wuhan University of Technology, Wuhan 430070, China*
[2]*Operational Software and Simulation Institution of Dalian Naval Academy, Dalian 116018, China*

Correspondence should be addressed to Xingjun Chen; cxj_dna@yeah.net

Received 26 February 2021; Accepted 20 June 2021; Published 5 July 2021

Academic Editor: Yong Zhang

With the continuous development of computer and network technology, the large-scale and clustered operations of drones have gradually become a reality. How to realize the reasonable allocation of UAV cluster combat tasks and realize the intelligent optimization control of UAV cluster is one of the most challenging difficulties in UAV cluster combat. Solving the task allocation problem and finding the optimal solution have been proven to be an NP-hard problem. This paper proposes a CSA-based approach to simultaneously optimize four objectives in multi-UAV task allocation, i.e., maximizing the number of successfully allocated tasks, maximizing the benefits of executing tasks, minimizing resource costs, and minimizing time costs. Experimental results show that, compared with the genetic algorithm, the proposed method has better performance on solving the UAV task allocation problem with multiple objectives.

1. Introduction

With the rapid development of Internet of Things and 5G communication technologies, UAV systems are increasingly used in the military field and UAV operations have become an important part of modern military operations. Single-UAV combat often lacks support, guarantee, and target cover, and it usually requires an overall force to complete combat missions. Task allocation is one of the most important problems that need to be solved in multi-UAV operations, and it directly affects the efficiency and profitability of operations. Finding the optimal solution of the task allocation problem has been proven to be NP-hard, and the solving difficulty increases exponentially with the scale of UAV cluster and tasks. Furthermore, task allocation is often a multiobjective optimization problem, which makes the model more complicated, e.g., simultaneously maximizing the number of tasks to be performed, maximizing the benefits of performing tasks, minimizing time costs, and minimizing resource consumption.

Many methods for solving the task allocation problem have been proposed, which can be roughly divided into four categories: graph theory [1], integer linear programming [2], state space search [3], and Artificial Intelligence (AI) methods such as genetic algorithm, particle swarm algorithm, simulated annealing, and ant colony algorithm [4–8]. Most methods in the first three categories are complete search algorithms. These algorithms can get the optimal solution, but they require a lot of computing resources and time cost, and it is impractical to apply them to a large-scale problem. AI methods cannot guarantee an optimal solution, but they usually can obtain a local-optimal solution [9] within a reasonable period of time. The above algorithms often optimize a certain goal, such as task revenue [10] or time/resource cost [11].

Artificial immune system (AIS) is an emerging research direction of computational intelligence. A clone selection algorithm (CSA) is proposed based on related immune principles [12–14]. This algorithm is widely used in function optimization [15] (e.g., multimodal optimization and continuous function optimization), pattern recognition [16, 17] (e.g., binary character and face recognition), and scheduling problems [18]. Compared with those complete search algorithms, CSA has some advantages and is convenient for practicality and engineering. At the same time, CSA can be used to solve multiobjective problems. Comparing CSA with the GA [19–21], the main difference is the way the population evolves. In the GA, the population evolves through crossover

and mutation, and in the CSA, cell reproduction is asexual, with each offspring produced by one cell being an exact copy of its parents, and mutation and selection are made through these offspring. This paper proposes to use CSA to optimize four objectives in UAV task allocation, i.e., maximizing the number of successfully assigned tasks, maximizing the benefits of executing tasks, minimizing resource cost, and minimizing time cost, and comprehensively considers the time constraints, resource constraints, and functional constraints in real-world scenarios. Comparing with the brute-force search algorithm and genetic algorithm, experimental results show that the proposed method could achieve better performance on solving high-dimensional multiobjective task allocation problems.

The remaining sections of this paper are organized as follows: Section 2 introduces the description of task allocation problems; Section 3 present the details of the proposed method; Section 4 presents the experimental results and our analysis; the proposed work is summarized in Section 5.

2. Problem Description

Task allocation involves many objects and related attributes. This section will give a formal representation of the elements, goals, and constraints involved in task allocation [22, 23] to facilitate later expression.

2.1. Task Allocation

(1) UAV: as the code of military operations, UAV is expressed as $U = \{u_1, u_2, \cdots, u_m\}$, where u_i represents UAV $i\,(i = 1, \cdots, m)$ and m is the number of UAVs. The initial position of the UAV is expressed as $\mathrm{Pu} = \{\mathrm{pu}_1, \mathrm{pu}_2, \cdots, \mathrm{pu}_m\}$, where pu_i represents the initial position of the UAV i. The ammunition and fuel carried by each UAV are expressed as $\mathrm{ResU} = \{\mathrm{resu}_1, \cdots, \mathrm{resu}_m\}$, where resu_i represents the number of resources carried by the UAV i.

(2) Task: as a combat task and an indivisible unit, the task is expressed as $T = \{t_1, t_2, \cdots, t_n\}$, where t_j represents the task $j\,(j = 1, \cdots, n)$ and n is the number of tasks. The initial position of the task is expressed as $\mathrm{Pt} = \{\mathrm{pt}_1, \mathrm{pt}_2, \cdots, \mathrm{pt}_n\}$, where pt_j represents the initial position of task j. The execution of each task requires certain resources to be consumed, which is expressed as $\mathrm{ResT} = \{\mathrm{rest}_1, \cdots, \mathrm{rest}_n\}$, where rest_j represents the resources consumed by task j. Performing each task will obtain a different task revenue expressed as $\mathrm{Reward} = \{\mathrm{reward}_1, \cdots, \mathrm{reward}_n\}$, where reward_j represents the revenue of executing task j. The validity period of each task is limited by time. Each task has the earliest start execution time early_j and the latest start execution time late_j; the executable range of each task is expressed as $\mathrm{TR} = \{[\mathrm{early}_1, \mathrm{late}_1], \cdots, [\mathrm{early}_n, \mathrm{late}_n]\}$, where $[\mathrm{early}_j, \mathrm{late}_j]$ is the executable range of task j. It tasks different time to execute different tasks; the time consumed to execute the task is expressed as $\mathrm{Time} = \{\mathrm{time}_i, \cdots, \mathrm{time}_n\}$, where time_j represents the time consumed to execute task j.

(3) Execution sequence: a task is executed by only one drone. The execution sequence of the UAV is represented as $\mathrm{Su} = \{\mathrm{su}_1, \cdots, \mathrm{su}_m\}$. Among them, su_i represents the execution sequence of UAV i, where su_i is composed of corresponding tasks, specifically represented as $\mathrm{su}_i = \{t_x t_y t_q t_z t_d\}$, where $0 < x, y, q, z, d \le n$. $|\mathrm{su}_i|$ is the number of tasks executed by the UAV i, and they are, respectively, task x, task y, task q, task z, and task d. su_{ij} is the jth task in the execution sequence of the UAV i. According to the position of the task in the execution sequence, the corresponding task can be expressed as $t_x = \mathrm{su}_{i1}$, $t_y = \mathrm{su}_{i2}$, $t_q = \mathrm{su}_{i3}$, $t_z = \mathrm{su}_{i4}$, and $t_d = \mathrm{su}_{i5}$.

(4) Task allocation: UAV and task are the two subjects of allocation. Since a task can only be performed by one UAV, the allocation relationship can be expressed as $A = \{a_1, \cdots, a_n\}$, where a_j represents the assignment relationship of the task j. If task j is not assigned to a drone, then $a_j = \mathrm{NULL}$, and if assigned, it means the inequality $a_j \neq NULL$. At the same time, the assignment to the relevant UAV can be obtained one step further, expressed as $a_j = u_i$; it means that task j is assigned to UAV i for execution.

2.2. Optimization Objective. With the basic description of the above basic elements, we further derive the definition of objectives to be optimized in UAV task allocation problems.

(1) Maximize the number of successfully assigned tasks:

$$\mathrm{target}_1 = \max_{\forall A} \sum_{j=1}^{n} \{a_j \neq \mathrm{NULL}\}, \tag{1}$$

where $a_j \neq \mathrm{NULL}$ is true; the return value is 1; otherwise, the value is 0.

(2) Maximize the benefits of performing tasks:

$$\mathrm{target}_2 = \max_{\forall A} \sum_{j=1}^{n} \{a_j \neq \mathrm{NULL}\} \times \mathrm{reward}_j. \tag{2}$$

(3) Minimize resource cost:

$$\begin{aligned}\mathrm{target}_3 = \min_{\forall Su} & \sum_{i=1}^{m} (\mathrm{arrive\_cost}(\mathrm{pu}_i, \mathrm{su}_{i1})) \\ & + \sum_{j=2}^{|\mathrm{su}_i|} \left(\mathrm{arrive\_cost}\left(\mathrm{su}_{i(j-1)}, \mathrm{su}_{ij}\right)\right) + \sum_{j=1}^{|\mathrm{su}_i|} \mathrm{res}_{\mathrm{su}_{ij}},\end{aligned} \tag{3}$$

where arrive_cost$(\mathrm{pu}_i, \mathrm{su}_{i1})$ represents the resource consumption cost required to execute the first task in the task sequence from agent i to agent.

(4) Minimize time consumption:

$$target_2 = \min_{\forall Su} \max \sum_{i=1}^{m} \mathrm{time\_cost}(\mathrm{su}_i, |\mathrm{su}_i|), \quad (4)$$

where time_cost$(\mathrm{su}_i, |\mathrm{su}_i|)$ represents the time; it tasks for the UAV r_i to execute the tasks in the s_i sequence.

2.3. Constraints. In real-world scenarios, there are often some constraints in task allocation problems. In this paper, we mainly consider the following common constraints [16]:

(1) Time constraint: a UAV can only successfully execute the task when it starts within the executable time range of the task. For the execution sequence Su_i of the UAV u_i, the constraint is as follows:

For $j = 1$,

$$\mathrm{time\_constraint}_{i1}^{T} = \mathrm{arrive}_{\mathrm{time}(\mathrm{pu}_i,1)} = \mathrm{travel}_{\mathrm{time}(\mathrm{pu}_i,\mathrm{su}_{i1})} \le \mathrm{late}_{\mathrm{su}_{i1}}. \quad (5)$$

For $j = 2, \cdots, |\mathrm{Su}_i|$,

$$\mathrm{time\_constraint}_{ij}^{T} = \mathrm{arrive}_{\mathrm{time}(u_i,j)} = \mathrm{time}(\mathrm{su}_{ij}) + \mathrm{travel}_{\mathrm{time}(\mathrm{pu}_i,\mathrm{su}_{ij})} \le \mathrm{late}_{\mathrm{su}_{i1}}. \quad (6)$$

(2) Resource constraints: UAV tasks are limited by its own resources. For the execution sequence Su_i of UAV u_i, the time constraints are as follows:

For $j = 1, \cdots, |\mathrm{Su}_i|$,

$$\mathrm{assets\_constraint}_{ij}^{R} = \mathrm{resource}_{\mathrm{cost}(\mathrm{pu}_i,j)} = \mathrm{travel}_{\mathrm{csot}(\mathrm{pu}_i,\mathrm{su}_{i1})} + \sum_{k=2}^{j} \mathrm{rest}_{\mathrm{su}_{ik}} \le \mathrm{resu}_i. \quad (7)$$

(3) Functional constraints: in real scenarios, different types of UAVs have different functions and therefore perform different tasks. This constraint is explained from the perspective of drones and tasks:

(i) From the perspective of agents, for $i = 1, \cdots, m$,

$$\mathrm{function\_constraint}_i^{U} = \{t_x t_y \cdots t_z t_d\}, \quad (8)$$

The constraint indicates that agent i can only execute task x, task y, task z, and task d due to functional limitations, where $0 < x, y, q, z, d \le n$.

(ii) From the perspective of tasks, for $j = 1, \cdots, n$,

$$\mathrm{function\_constraint}_j^{T} = \{u_e, \cdots, u_f\}. \quad (9)$$

This shows that the constraint indicates that task j can only be executed by UAV e, UAV f, etc. due to functional limitations, where $0 < e, f \le m$.

3. The Proposed Method

The CSA algorithm is a kind of the artificial immune system, which mainly contains ideas such as clone selection, receptor editing, and antibody circulation supplement mechanism and selects mature antibody cells through affinity, and uses a limited gene library to identify endlessly changing antigens. The CSA algorithm simulates immune mechanisms such as clone selection and amplification of antibodies, high-frequency mutations, and receptor editing during the immune response process of the immune system, so that it has strong self-learning, self-organization, and adaptive capabilities. Optimization-related fields are widely used. In this paper, the distribution plan of the UAV is mainly coded into antibody cells, and the final Pareto solution set is obtained through continuous iteration of antibodies [24, 25] (described in detail in Section 3.5).

3.1. Basic Framework. In this section, the basic program flow chart of the algorithm will be given, as shown in Figure 1.

It can be seen that the CSA model is relatively simple and convenient for coding operation. The main steps are described as follows:

(1) First, randomly initialize the UAV task allocation solution, organize the execution sequence of each UAV, and evaluate the affinity function of each antibody, and set the number of antibodies to$N$

(2) Judging the number of iterations, when it reaches a certain number (maxGen is the maximum number), the algorithm ends the output distribution plan

(3) Perform cloning operations on the current population, which involves the number of clones and the proportion of clones selected according to their affinity

(4) The cloned antibody is mutated according to the affinity ratio of the cloned individual

(5) Select$N$from the original population and the cloned population 2$N$for the next iteration

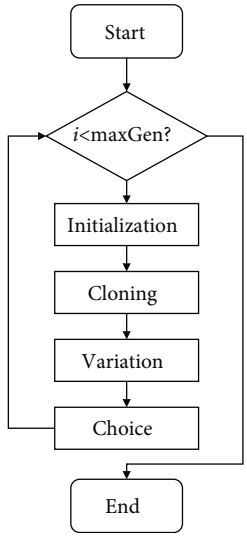


Figure 1: Flow chart of CSA [12].

(6) Repeat steps (2)–(5). This is a simple program description process of the algorithm, and each process will be described in detail below

3.2. Encoding of Antibody. An important part of using evolutionary algorithms is to encode real-world problems into antibodies, which are feasible solutions to allocation problems. For the CSA algorithm, the concept of antibody is very important, and the code of the antibody also determines the actual optimization effect of the algorithm. These solutions are composed of the execution sequence of each UAV. These components are called "genes," and their practical meaning is the actual distribution execution sequence of the UAV. Here is a specific description of the encoding rule [22]. For example, Table 1 shows an example of 3 drones and antibody codes composed of 10 tasks.

The following information can be obtained from the coding example in the figure above. From the functional constraints of the UAV, function_constraint$_1^U = \{1, 7, 2, 4\}$, function_constraint$_2^U = \{3, 6, 4, 5\}$, and function_constrain t$_3^U = \{8, 9, 2,10,1\}$. From the perspective of the functional constraints of the task, function_constraint$_2^T = \{1, 3\}$, and function_constraint$_3^T = \{2\}$. This kind of coding is directly based on the functional constraints of the drone, reducing a lot of useless calculations and judgments, which is conducive to improving efficiency. At the same time, it can also get $\text{su}_1 \subseteq \{1, 7, 2, 4\}$, $\text{su}_2 \subseteq \{3, 6, 4, 5\}$, and $\text{su}_3 \subseteq \{8, 9, 2,10,1\}$. We know the actual execution sequence that the UAV finally allocates must be a subset of the UAV's functional constraints. At the same time, there are some shortcomings in this coding. As far as t_2 is concerned, due to its simultaneous existence in the functional constraints of u_1 and u_3 (in simple terms, there is a many-to-many relationship between the UAV and the task), it is difficult for the algorithm to choose who performs the more excellent, simple processing is carried out in the encoding here, and it is stated that the UAV with lower encoding will be executed first; that is to say, if u_1 is not limited in resources and time, t_2 can be executed first, and then t_2 in the function constraint of u_3 fails; if the u_1 constraint is not satisfied, u_3 starts to judge the conditions for executing t_2.

Table 1: Example of antibody coding.

UAV 1	UAV 2	UAV 3
1 7 2 4	3 6 4 5	8 9 2 10 1

3.3. Cloning, Mutation, and Selection. The original CSA algorithm selects cloned antibodies according to the degree of affinity. Most antibodies with higher affinity are cloned for mutation in order to produce better individuals; a small number of antibodies with poor affinity are cloned to prevent mutation. The algorithm enters the local optimum to improve the quality of the solution. Such a strategy is in line with the realistic model and has a certain optimization effect; however, because the problem is a multiobjective optimization, each solution may evolve into a Pareto solution. Therefore, under this problem model, we assume that each antibody in each original population will be cloned according to its affinity to change the following individuals.

The mutation operation is performed in the cloned individual. The main operation of mutation is to randomly change part of the gene in the antibody. The algorithm mutates according to the degree of affinity. It is assumed that individuals with higher affinity have higher quality solutions, individuals with higher clone affinity have a lower mutation rate, and individuals with lower affinity have a higher mutation rate. This is because the quality of individual solutions with high affinity has been further optimized. Less variation is to maintain the quality of its own solutions and to explore around the solution at the same time; while individuals with low affinity undergo a lot of variation to let it explore a larger solution space. This also means that individuals with less affinity change fewer gene segments, while individuals with greater affinity change more. The detailed change parameters will be given in the experimental part.

3.4. Affinity Function. The four objectives in Section 2 are normalized, and the corresponding constraints are added to the calculation of the affinity function. When evaluating each individual x, the following affinity function can be used:

(1) The value of the first objective can be calculated by the following formula:

$$\text{affinity}_1(x) = 1.0 - \frac{\sum_{i=1}^{m} e1(x_i)}{n}, \tag{10}$$

where $x_i = i_1, \cdots, i_b$ and $e1(x_i)$ are calculated by

$$e1(x_i) = \sum_{w=1}^{b} h(i_w), \tag{11}$$

where

$$h(i_w) = \begin{cases} \text{if } v_{i_w} \text{ has been allocated,} \\ 0, \quad \text{or arrival\_time}\left(x_i^E + i_w, \left|x_i^E\right| + 1\right) > \text{late}_{i_w}, \text{or resource\_cost}\left(x_i^E + i_w, \left|x_i^E\right| + 1\right) > \text{res}_i, \\ 1, \quad \text{otherwise,} \end{cases} \tag{12}$$

and it judges whether v_{i_w} can be added to the current execution sequence of u_i, x_i^E (it is the initial empty set) is the current execution sequence of a_i, and $x_i^E + i_w$ means adding i_w to x_i^E

(2) The value of the second objective can be calculated by the following formula:

$$\text{affinity}_2(x) = 1.0 - \frac{\sum_{i=1}^{m} e2(x_i)}{\sum_{i=1}^{n} \text{reward}_i}, \tag{13}$$

where $e2(x_i) = \sum_{w=1}^{b} h(i_w) \times \text{reward}_{i_w}$

(3) The value of the third objective can be calculated by the following formula:

$$\text{affinity}_3(x) = \sum_{i=1}^{m} \frac{e3(x_i)}{m \times \max\_r}, \tag{14}$$

where max_r represents the maximum amount of resources carried by the drone, $e3(x_i) = \sum_{w=1}^{k} h(i_w) \times (\text{travel\_cost}(p_{\text{last}}, p_{vi_w}) + \text{rescost}_{i_w})$, and p_{last} represents the location of the last task assigned to u_i (when $w = 1$, p_{last} represents the location of u_i)

(4) The value of the forth objective can be calculated by the following formula:

$$\text{affinity}_4(x) = \max_{i=1}^{m} \frac{e4(x_i)}{m \times \max\_t}, \tag{15}$$

among them, max_t is the longest time to complete and $e4(x_i) = \sum_{w=1}^{k} h(i_w) \times (\text{travel\_time}(p_{\text{last}}, p_{vi_w}) + \text{timecost}_{i_w})$

It is worth noting that not all tasks encoded in antibodies can be successfully assigned to drones, and the order in which drones perform tasks is implicit in the antibody. Through above transformation, we can input the individual into the four functions in formulas (10)–(15) to facilitate the evaluation of the individual.

3.5. Multiobjective Optimal Solution Set. Since in the case of multiobjective, each individual has multiple attributes (each task is identified as an attribute in the coding here); the comparison between two individuals cannot simply use the size relationship. Therefore, this section will briefly introduce the basis of multiobjective optimization.

3.5.1. Domination Relationship. In the domination relationship between different individuals, let x and y be two different individuals in the multiobjective optimization population, if the following two conditions are met:

(1) For all subgoals (referred to as four goals in this article), x is no worse than y, that is, $f_k(x) \leq f_k(y) (k = 1, 2, \cdots, r)$

(2) There is at least a certain subgoal that makes x better than y. It is expressed as $\exists l \in \{1, 2, \cdots, r\}$, which satisfies $f_l(x) < f_l(y)$

At this time, x is called nondominated and y is dominated, where x dominates y. It can be symbolized as $x \prec y$. If x and y do not meet the above conditions, it proves that there is no dominant relationship between the two.

3.5.2. Pareto Solution Set. It can be obtained from the above dominance relationship that all individuals in a population can be sorted by the definition of dominance relationship, but because some solution sets may not have dominance relationships, these solution sets are in the same position. Through these characteristics, a solution set can be obtained. Each individual z in this solution set satisfies

(1) Individuals in the population are dominated by it, expressed as $z \prec p, p \in A$

(2) Other individuals cannot dominate it; that is, there is no dominance relationship between the two, expressed as $z \nprec q, q \nprec z, q \in B$

In (1) and (2), $A \cap B = \varnothing$, and $A \cup B = U$.

At the end of the experiment, we will select the Pareto-based multiobjective optimal solution set as the task allocation plan, in which the subobjectives compared between individuals can refer to the affinity function.

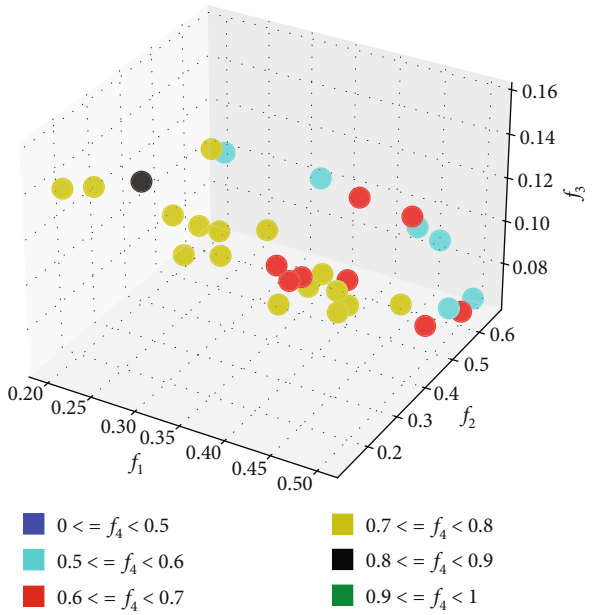
Table 2: Parameter settings.

Parameter name	Value
Hypermutation ratio	[0.2, 0.5, 0.8, 1]
Population size	200
Number of iterations	1000
Number of clones	1
Recombination ratio	[0, 0.5]
Optimization number	4

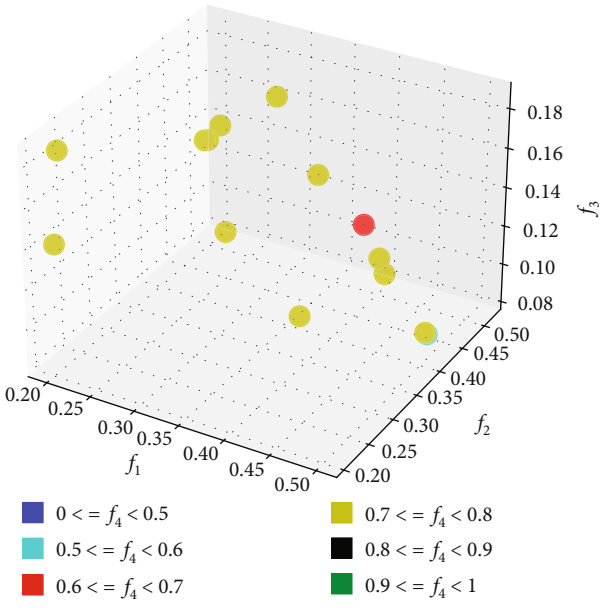
4. Experimental Results

This section will demonstrate the effectiveness of the method proposed in this article through experimental comparison and conclusion analysis.

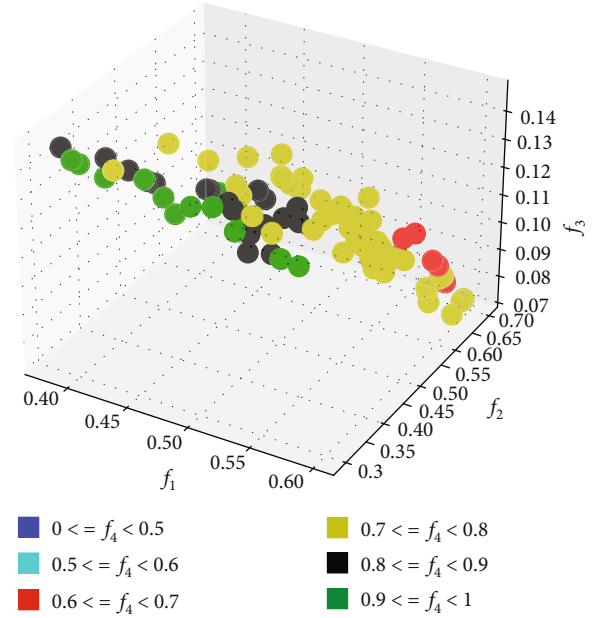
4.1. Experimental Environment. The experimental environment of this article is as follows: Window 10 operating system 64-bit professional edition, Intel i7-7600U CPU, clocked at 2.80 GHz, memory 8 G, and the programming environment is Visual Studio 2010.



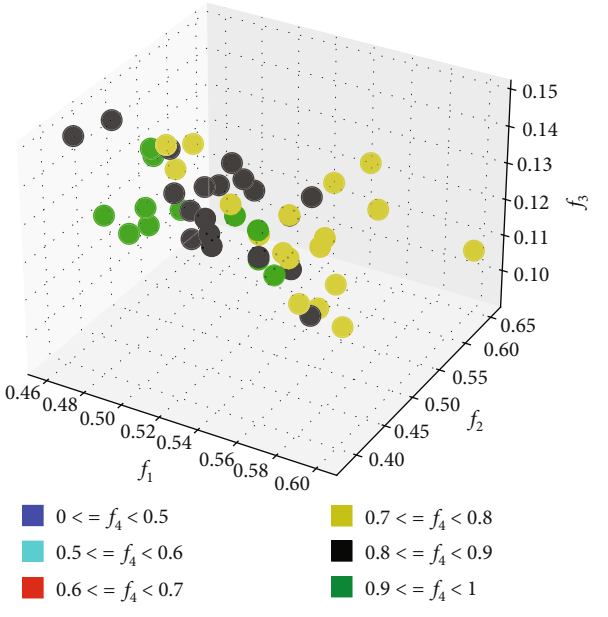
(a) CSA algorithm ($m = 5$, $n = 10$)



(b) GA algorithm ($m = 5$, $n = 10$)

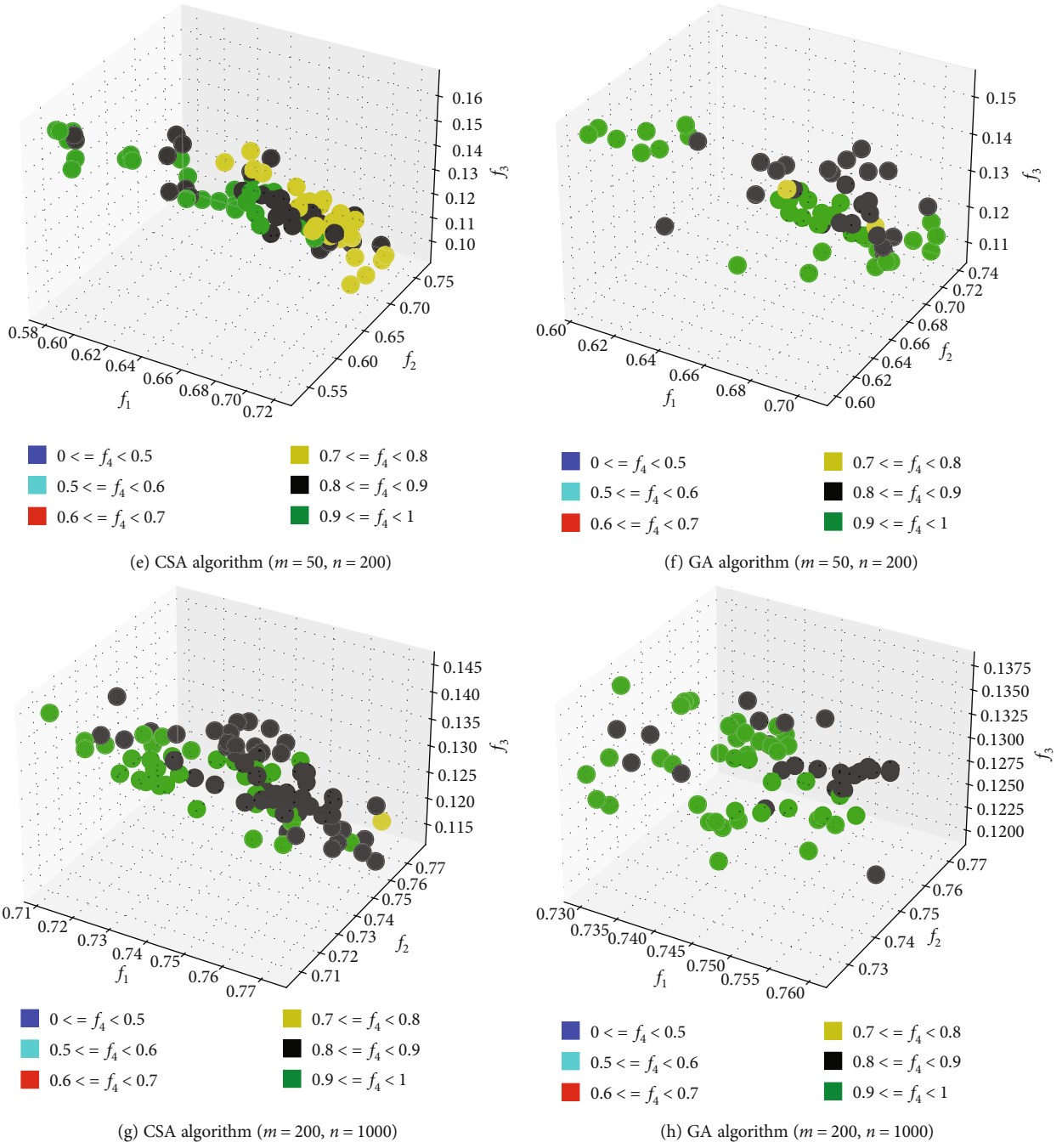


(c) CSA algorithm ($m = 20$, $n = 50$)



(d) GA algorithm (m=20, n=50)

Figure 2: Continued.



(e) CSA algorithm ($m = 50$, $n = 200$)

(f) GA algorithm ($m = 50$, $n = 200$)

(g) CSA algorithm ($m = 200$, $n = 1000$)

(h) GA algorithm ($m = 200$, $n = 1000$)

Figure 2: distribution of Pareto solution sets for different scales of problems.

First, randomly generate relevant agent and task data. In order to facilitate comparative experiments, randomly generate test data in the preparation phase. In order to simulate the experiment, supposing the initial test position of each UAV is $\text{pu}_i = (x, y)(i = 1 \cdots m)$, x and y are random integers between [0,100); the earliest possible execution time early_j is a random number between [0,100], and the latest execution time late_j is a random number between [early_j,150); the time required to execute the task time_j is random number between [1,10); the resources consumed to execute the task rest_j is a random number between [0,100); the revenue of the task reward_j is a random number between [0,1) [22]. For functional constraints, this article assumes that each task must have one or more drones that can perform it. After a certain number of iterations, the overall UAV and task change. We make corresponding changes to the antibody through the change information. In order to facilitate the experiment, it is assumed that the time cost and resource cost of transmission

Table 3: Comparison results of different algorithms.

Problem scale	GA (No. of nondominated solutions)	CSA (number of nondominated solutions)	No. of solutions (GA dominates CSA)	No. of solutions (CSA dominates GA)
5_10	14	31	0	1
20_50	43	87	0	5
50_200	57	88	0	3
200_1000	59	93	0	6

Table 4: Optimization objective results for different ratios of tasks to UAVs.

Problem scale	Number of nondominated solutions	Min f_1	Min f_2	Min f_3	Min f_4
50_100	21	0.2500	0.2402	0.0802	0.6804
50_150	16	0.4733	0.4227	0.0988	0.7448
50_200	13	0.5750	0.5479	0.1026	0.7358
50_250	11	0.6680	0.6200	0.1111	0.7001

have a linear relationship with distance. The experimental parameters of the main CSA algorithm are shown in Table 2.

4.2. Comparison of Results of Different Algorithms. In this paper, the CSA algorithm and the GA algorithm are consistent with the experimental test data. The algorithm parameter environment and parameters are as shown in the subsection. Four sets of task allocation test data of different scales will be carried out to compare the final Pareto set. Figure 2 shows the two algorithms $(m = 5, n = 10)$, $(m = 20, n = 50)$, $(m = 50, n = 200)$, and $(m = 200, n = 1000)$, respectively. The distribution of Pareto solution set. Table 3 compares the dominance of the Pareto solution set.

It can be observed from Figure 2 that in the comparative experiments of different scales, the number of nondominated solutions of the CSA algorithm is more, and the distribution area of the solutions is wider, which also means that the diversity is better, from the analysis in Table 3. It can be obtained that the solution of the CSA algorithm is generally better than that of the GA algorithm. This trend becomes more obvious with the increase of scale. The main reason is that the mutation strategy of the CSA algorithm is better than the mutation strategy of the GA; the CSA variation dynamically changes individuals with the degree of affinity, while the GA algorithm only performs a small amount of mutation on the basis of crossover and variation range is often small, resulting in a small exploration pace, and the dynamic change of the variation range is beneficial to improve the efficiency of the solution. The experimental results just proved this point.

4.3. Results on Different Ratios of Tasks to UAVs. In this section, the results of CSA algorithm on different ratios of tasks to UAVs are presented, i.e., $n = 100$, 150, 200, and 250, when $m = 50$. Since there are many Pareto solutions and it is impossible to compare all the solutions, there is only one of the similar data (i.e., if the difference between two data is within 0.05 in f_1, f_2, f_3, f_4). Table 4 compares the experimental results of different ratios.

According to Section 3.4, the smaller the target value of f_1, f_2, f_3, f_4, the better the optimization effect. In Table 4, it can be seen that in the comparative experiments of different ratios, when the number of UAVs remains the same, as the task scale increases rapidly, the difficulty of solving the problem is further increased. It will lead to a decrease in the number of nondominated solutions and a decrease in quality (the values of f_1, f_2, f_3, f_4 continue to increase), which is in line objective facts. Combining this, in order to maximize the number of tasks successfully assigned, maximize the benefits of tasks execution, minimize the resource costs, and minimize the time costs, the ratio of drones and tasks must be balanced as much as possible. Increase the number of drones as much as possible to optimize the allocation plan, when considering the constraints of many conditions.

5. Conclusion

In order to solve the problem of UAV combat task allocation, this paper proposes a clone selection algorithm based on the artificial immune system; this algorithm simultaneously optimizes four objectives, namely, maximizing the number of tasks successfully assigned, maximizing the benefits of tasks execution, minimizing the resource costs, and minimizing the time costs. And the effectiveness of method is proved by experimental comparison. In the later stage, the comparison algorithm model will be improved mainly by the high-dimensional multiobjective optimization strategy of genetic algorithm (for example, niche strategy and reference point based on hyperplane), and the gene recombination of clone selection algorithm will be added to optimize the CSA algorithm to further improve the quality of solutions.

Data Availability

All the data can be generated according to the steps described in our paper, and readers can also ask for the data by contacting cxj_dna@yeah.net.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the Natural Science Foundation of China (No. 71701208).

References

[1] V. Chaudhary and J. K. Aggarwal, "A generalized scheme for mapping parallel algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 3, pp. 328–346, 1993.

[2] W. W. Chu, L. J. Holloway, Min-Tsung Lan, and K. Efe, "Task allocation in distributed data processing," *IEEE computer*, vol. 13, no. 11, pp. 57–69, 1980.

[3] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, vol. 6, no. 3, pp. 42–50, 1998.

[4] V. Lesser, C. L. Ortiz Jr., and M. Tambe, *Distributed Sensor Networks: A Multiagent Perspective*, vol. 9, Springer Science & Business Media, 2012.

[5] A. Chapman, R. A. Micillo, R. Kota, and N. Jennings, *Decentralised Dynamic Task Allocation: A Practical Game–Theoretic Approach*, The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS '09), Hungary, 2009.

[6] S. Fitzpatrick and L. Meertens, *Distributed Sensor Networks. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series)*, vol. 9, Springer, Boston, MA, 2003.

[7] H. Yedidsion, R. Zivan, and A. Farinelli, "Applying max-sum to teams of mobile sensing agents," *Engineering Applications of Artificial Intelligence*, vol. 71, pp. 87–99, 2018.

[8] C. Wei, Z. Ji, and B. Cai, "Particle swarm optimization for cooperative multi-robot task allocation: a multi-objective approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2530–2537, 2020.

[9] W. Lee, N. Vaughan, and D. Kim, "Task allocation into a foraging task with a series of subtasks in swarm robotic system," *IEEE Access*, vol. 8, pp. 107549–107561, 2020.

[10] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[11] G. Yang and V. Kapila, "A dynamic-programming-styled algorithm for time-optimal multi-agent task assignment," *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol. 2, 2001, pp. 1959–1964, Orlando, FL, USA, 2001.

[12] S. Valluru, *Implementation of a Clonal Selection Algorithm*, 2014, Technical Report.

[13] W. Tian and J. Liu, "An improved immune clone selection algorithm for multi robot task scheduling," in *2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009)*, pp. 128–131, Zhangjiajie, China, 2009.

[14] H. Dai, Y. Yang, and C. Li, "Hybrid crossover based clonal selection algorithm and its applications," in *In international conference on intelligent data engineering and automated learning*, H. Yin, Y. Gao, B. Li, D. Zhang, M. Yang, Y. Li, F. Klawonn, and A. J. Tallón-Ballesteros, Eds., pp. 468–475, Springer, Cham, 2016.

[15] M. Y. Zhao, K. E. Tang, G. Lu et al., "A novel clonal selection algorithm and its application," in *2008 International Conference on Apperceiving Computing and Intelligence Analysis*, pp. 385–388, Chengdu, China, 2008.

[16] J. H. Li, H. W. Gao, and S. A. Wang, "A novel clone selection algorithm with reconfigurable search space ability and its application," in *2008 Fourth International Conference on Natural Computation*, vol. 6, pp. 612–616, Jinan, China, 2008.

[17] J. A. White and S. M. Garrett, "Improved pattern recognition with artificial clonal selection?," in *Artificial Immune Systems. ICARIS 2003*, J. Timmis, P. J. Bentley, and E. Hart, Eds., vol. 2787 of Lecture Notes in Computer Science, pp. 181–193, Springer, Berlin, Heidelberg, 2003.

[18] X. Pan, F. Liu, and L. Jiao, "A dynamic clonal selection algorithm for project optimization scheduling," in *Simulated Evolution and Learning. SEAL 2006*, T. D. Wang, Ed., vol. 4247 of Lecture Notes in Computer Science, pp. 821–828, Springer, Berlin, Heidelberg, 2006.

[19] K. Huang, Y. Dong, D. Wang, and S. Wang, "Application of improved simulated annealing genetic algorithm in task assignment of swarm of drones," in *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pp. 266–271, Xi'an, China, 2020.

[20] Y. Jia, "Research on UAV task assignment method based on parental genetic algorithm," in *Advances in Swarm Intelligence. ICSI 2019*, Y. Tan, Y. Shi, and B. Niu, Eds., vol. 11655 of Lecture Notes in Computer Science, pp. 439–446, Springer, Cham, 2019.

[21] Z. Zha, C. Li, J. Xiao et al., "An improved adaptive clone genetic algorithm for task allocation optimization in ITWSNs," *Journal of Sensors*, vol. 2021, Article ID 5582646, 12 pages, 2021.

[22] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, "Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm," *IEEE Access*, vol. 8, pp. 19306–19318, 2020.

[23] J. Zhou, X. Zhao, D. Zhao, and Z. Lin, "Task allocation in multi-agent systems using many-objective evolutionary algorithm NSGA-III," in *Machine Learning and Intelligent Communications. MLICOM 2019*, X. Zhai, B. Chen, and K. Zhu, Eds., vol. 294 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 677–692, Springer, Cham, 2019.

[24] J. Teich, "Pareto-front exploration with uncertain objectives," in *Evolutionary Multi-Criterion Optimization. EMO 2001*, E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, Eds., vol. 1993 of Lecture Notes in Computer Science, pp. 314–328, Springer, Berlin, Heidelberg, 2001.

[25] D. A. Van Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a Pareto front," in *Late breaking papers at the genetic programming 1998 conference*, pp. 221–228, Madison, Wisconsin, 1998.