Research on improved genetic simulated annealing algorithm for multi-UAV cooperative task allocation

Yao Wang1, Yongkang Shi† and Yunhui Liu1

1 School of Mechanical Engineering, Xinjiang University, Urumqi, Xinjiang Province, 830017, China

*Corresponding author’s e-mail: sunwenxj@sina.cn

Abstract. In order to solve the cooperative search problem of multiple unmanned aerial vehicles (multi-UAVs) in a large-scale area, we propose a genetic algorithm (GA) incorporating simulated annealing (SA) for solving the task region allocation problem among multi-UAVs on the premise that the large area is divided into several small areas. Firstly, we describe the problem to be solved, and regard the task areas allocation problem of multi-UAVs as a multiple traveling salesman problem (MTSP). And the objective function is established under the premise that the number of task areas to be searched by each UAV is balanced. Then, we improve the GA, using the advantages of the SA can jump out of the local optimal solution to optimize the new population of offspring generated by GA. Finally, the validity of the algorithm is verified by using the TSPLIB database, and the set MTSP problem is solved. Through a series of comparative experiments, the validity of GAISA and the superiority of solving the MTSP problem can be demonstrated.

1. Introduction

In recent years, natural disasters have occurred frequently and the scope of disasters has continued to expand. When a natural disaster occurs, the search and rescue of people and the survey of secondary disasters become particularly important. We can use UAVs to search the disaster-stricken areas and protect the lives and property of disaster-stricken people. However, when UAVs perform search tasks in large-area scenes, it is difficult to achieve the expected results due to the limitations of endurance and load capacity. The multi-UAV collaboration technology can solve this problem.

When performing a search task in a large area, the large area can be divided into several small areas to improve the search efficiency. Further, the multi-UAVs cooperative search problem is transformed into three subproblems: area division, sub-area allocation, and sub-area full coverage path planning. First, the large area is divided into several sub-areas by a certain method [1]. Then, the sub-areas are assigned to the UAVs according to the idea of equal division. Finally, the UAVs traverse their respective sub-task area in turn. And then, the multi-UAVs cooperative search problem turns into a multi-UAVs task allocation problem. Therefore, this paper focuses on the task allocation problem of multi-UAVs. Multi-UAVs task allocation is one of research contents in UAV swarm collaboration technology [2], which can be regarded as the MTSP, the MTSP is often seen as a further extension and application of the TSP. Intelligent heuristics commonly used to solve the MTSP problem include ant colony algorithms [3], simulated annealing algorithms [4], genetic algorithms [5], etc.

Omar Cheikhrouhou [6], summarized the relevant research on MTSP problems in recent years, and many scholars have improved the quality and efficiency of solving MTSP problems by improving the traditional algorithms. Although the ant colony algorithm (ACA) is robust and has a strong solving
capability, it is slow to converge and prone to fall into local optima when solving optimization problems of large scale [7]. To address this drawback, Ebadenezhad Sahar [8] proposed an adaptive ACA with unique strategy, which can improve the uncertain convergence time and random decision of the traditional ACA by dynamically adjusting the parameters, so as to improve the convergence speed and search precision. Although the simulated annealing algorithm (SA) can jump out of the local optimum using the Metropolis criterion, the convergence speed is slow. Yan Zhang [9] proposed a state-transfer SA, firstly, a new intelligent operator is added to generate the solution space of MTSP, then introducing a 2-opt neighborhood search structure to improve the ability of capturing the global optimal solution, and finally using the Metropolis criterion to perform the solution, which improves the convergence speed to some extent. To address the problem that genetic algorithms (GA) converge faster but are prone to fall into local optima, Shuai Yuan [10] improved a crossover operator to improve the search ability of GA and obtained near-optimal solutions in dealing with the MTSP problem. Among other studies of intelligent algorithms, Yongbo Chen [11] proposed an improved wolfpack algorithm by improving the individual coding method, and improved the searching ability of the algorithm by introducing transposition and extension operation.

The above research is based on the traditional single algorithm proposed the corresponding improvement strategy, the optimization effect has certain promotion, but still has certain limitation. So, some scholars integrate the traditional algorithm two by two. Chao Jiang [12] proposed an ant colony-partheno GA to solve the large scale MTSP problem, which solves the problem of slow convergence, but this algorithm mainly solves the MTSP problem of single target, the multi-objective MTSP problem remains to be studied. He Qing [13] proposed an improved genetic simulated annealing algorithm to solve the problem that GA is easy to fall into local optimum and the convergence speed of SA is slow, but the algorithm mainly solves the TSP problem, and the feasibility of solving the MTSP problem is to be verified.

In summary, for the MTSP problem, most scholars improve on the basis of a single algorithm, and the optimization effect is not good. The related studies on algorithm fusion are less and mainly focus on TSP or MTSP single-objective problems. And most algorithms take the first point as the starting point directly, without considering the effect of the starting point in the practical application. To address the above problems and the main research object of this paper, a genetic algorithm integrating simulated annealing (GAISA) is proposed. The SA is used to optimize and adjust the updated population, which can effectively avoid the GA falling into the problem of local optimum, and at the same time, combined with the fast convergence of the GA itself, to solve the MTSP problem of multi-UAVs task allocation.

2. Problem Description & Modeling

2.1. Problem Description

Suppose that a large area has been divided into \( N \) sub-areas according to a certain method. These sub-areas need to be assigned to \( M \) UAVs as a task. Assume that the UAVs fly at a certain altitude during the search process. As shown in Figure 1, a point \( S \) is selected in each sub-task area, and the UAVs need to reach this point in the sub-region before starting the traversal in the task area. These points form a task set \( S = \{ S_1, S_2, \ldots, S_N \} \), at which point the multi-UAVs task allocation problem can be transformed into MTSP problem to solve.

2.2. Modeling

In the TSP problem, a traveling salesman visits all the cities, only once per city, and then returns to the starting point after visiting all the cities. In practical applications, when a UAV performs a search task, it often takes off outside the task area, and then visits the task area in turn, and each task area is visited only once. Return to the starting point after traversing all task areas. So, the model is:
where \( C \) is the distance cost, \( D_{S_k,S_{k+1}} \) represents the distance that the UAV returns from the last task point \( S_N \) to the starting point \( S_0 \), \( k \) counts from 0, and the effect of the starting point is considered in the distance cost, \( D(S_k, S_{k+1}) \) is the distance from the city \( S_k \) to \( S_{k+1} \). \( \sum_{k=0}^{N-1} D(S_k, S_{k+1}) \) represents the total distance from the first city \( S_1 \) to the last city \( S_N \). The distance between two points is \( \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \).

Figure 1. Schematic diagram of the transformation of sub-regions into task points.

In the MTSP problem, the task sequence of \( UAV_i \) can be represented as \( S_i = \{S_0^i, S_1^i, ..., S_n^i\} \), where \( S_0^i \) represents the start position of \( UAV_i \) and \( S_n^i \) represents the last task point, \( n^i \) is the number of tasks that the UAV needs to perform. The UAV needs to return to its starting point after completing the last task. The path cost for each UAV can be expressed as:

\[
C_i = D_{S_0^i,S_0^i} + \sum_{k=0}^{n^i-1} D(S_k^i, S_{k+1}^i)\\
\]

where \( C_i \) represents the path cost of \( UAV_i \), \( D_{S_0^i,S_0^i} \) represents the distance from \( S_0^i \) to \( S_0^i \), and \( D(S_k^i, S_{k+1}^i) \) represents the distance from \( S_k^i \) to \( S_{k+1}^i \). \( \sum_{k=0}^{n^i-1} D(S_k^i, S_{k+1}^i) \) represents the total distance that the \( UAV_i \) visits from starting point \( S_0^i \) to the last mission point \( S_n^i \), and \( k \) represents the \( k \) mission point.

With the objective of minimizing the total path cost of all UAVs, the objective function established is:

\[
\min C_{\text{total}} = \min \left( \sum_{i=1}^{M} \left( D_{S_{0}^{i},S_{0}^{i}} + \sum_{k=0}^{n^{i}-1} D(S_{k}^{i},S_{k+1}^{i}) \right) \right)\\
\sum n^{i} = N, i \in M
\]

3. GAISA design and improvement

3.1. Initialization

Firstly, the initial population size is \( G_0 \), and the maximum iteration number is \( K_{\text{max}} \). When the number of tasks is known to be \( N \), the length of the initialization chromosome is \( N \). Then the chromosome is decimal coded, and the \( N \) genes of the chromosome correspond to \( N \) different values, i.e., the
numbers within 1 to \( N \) are randomly arranged to represent \( N \) different task points. The initialized chromosomes are segmented according to the number of UAVs \( M \). Each UAV corresponds to a chromosome segment, the corresponding number of genes on the chromosome segment is the UAV needs to visit the task point, and the order of the gene arrangement is the order of the task point being visited. As Figure 2 shows the schematic diagram of chromosome coding for 15 task points.

Initialize the 1st chromosome as the global optimal solution \( R_{\text{global min}} \), initialize the path cost of the 1st chromosome as the global optimal cost \( C_{\text{global min}} \). Then initialize the current iteration number \( K = 1 \) and start the loop iteration.

3.2. Selection
In the case of the three UAVs, as shown in Figure 2, the initialized chromosome is divided into three segments. Then the starting position \( S_0 \) is added to the corresponding chromosome fragment. Chromosome segment 1 represents the first UAV needs to visit the task point and the order of the visits. Then the cost of each chromosome's fragment is calculated separately and then summed up and noted as the total cost \( C_{\text{total}} \).

$$\text{Fit} = \frac{1}{\sum_{i=1}^{M} C_{\text{total}}}$$  \hspace{1cm} (5)

Firstly, the fitness values of all individuals are calculated according to the fitness function. Then, a binary tournament method is used to select the highly adaptive individuals as the next generation, and keep the population always evolving in a favorable direction. Finally, the child population with the number of individuals \( N_{\text{ref}} \) is selected for subsequent operations by setting the genetic proportion \( P_{\text{gap}} \).

3.3. Crossover
The crossover operation involves randomly selecting two individuals in the offspring population with probability \( P_c \) and then manipulating their chromosomes. As shown in Figure 3, first, two numbers are randomly generated within the chromosome length, and the chromosome segment between the two numbers is moved to the forefront of the other chromosome. Then, each gene on the chromosome is traversed in turn, and the duplicate gene that follows is removed. This operation is repeated until there are no duplicate genes on that chromosome. That is, a new chromosome is obtained.

The mutation operation is performed with probability \( P_m \) on the chromosomes in the population after crossover. Any two genes are selected on the chromosome, and the selected 2nd gene is inserted
after the 1st gene. If the first gene is located behind the second, the genes at the other locations move back in turn. Otherwise, move forward in turn. That is, a mutated chromosome is generated. For example, the gene sequence on the chromosome is \( (3,7,1,4,2,6,8,5) \), the first gene selected at random is 2, and the second gene selected at random is 7. After performing the mutation operation, the gene sequence on the new chromosome becomes \( (3,1,4,2,7,6,8,5) \). Follow this method until all individuals have been traversed.

3.4. SA optimizes child population

The SA simulates the process of solid physical annealing. The initial temperature \( T_0 \) is lowered continuously, and a certain number of cycles are carried out at each temperature. Each cycle perturbs the current solution, resulting in a new solution. When the new solution is better than the current solution, the new solution is actively accepted. When the new solution is worse, the new solution is accepted according to the probability \( p = \exp(\Delta C/\Delta T) \). This process is iterated until the temperature reaches the set threshold or the set number of iterations is reached.

The advantages of the SA in its ability to climb the slope and easily jump out of the local optimum are fully utilized to further optimize and adjust the new population of the generated child population. The specific process is as follows.

(1) Initialization. Initialize the current child population, making \( \text{Selch} = \text{newSelch} \), size \( N_{sel} \), temperature \( T_0 \), maximum number of cycles for the outer layer \( MaxL_{out} \), maximum number of cycles for the inner layer \( MaxL_{in} \), cooling factor \( \alpha \), and the current individual \( j = 1 \).

(2) Circulation. When \( j \leq N_{sel} \), the \( j \) individual is selected for simulated annealing treatment. And then make \( j = j + 1 \). Otherwise, perform step 3).

a. First, calculate the cost \( C_j \) of the current individual, assign \( C_j \) to the global optimal cost \( C_{global} \), the local optimal cost \( C_{local} \), and make the solution \( R_j \) of the current individual equal to the global optimal solution \( R_{global} \), the local optimal solution \( R_{local} \). And make the number of outer loops \( L_{out} = 1 \), the number of inner loops \( L_{in} = 1 \), and \( T_0 = T \).

b. When \( L_{out} \leq MaxL_{out} \), perform step c. Otherwise, perform step e.

c. When \( L_{in} \leq MaxL_{in} \), Generate a new solution, \( R_{ta} \). Calculate the cost of \( R_{ta} \), \( C_{ta} \). If \( C_{ta} \leq C_{local} \), then \( C_{ta} \) is assigned to \( C_{local} \), and \( R_{ta} \) is assigned to \( R_{local} \). Otherwise, \( R_{ta} \) is accepted with probability \( p = \exp(- (C_{ta} - C_{local})/T) \). Then compare \( C_{local} \) with \( C_{global} \). If \( C_{local} \leq C_{global} \), then assign \( C_{local} \) to \( C_{global} \), and \( R_{local} \) to \( R_{global} \). Then, perform step d.

d. Let \( T = \alpha \times T \), \( L_{out} = L_{out} + 1 \). Then, perform step b.

e. Add \( R_{global} \) to the new population \( \text{newSelch} \). Update the population \( \text{newSelch} = \text{Selch} \).

(3) Output the optimized new population, \( \text{Selch} \).

3.5. Iteration

Some individuals of the parents were added into the optimized population, and the size of the new population reached \( G_n \). Calculate the cost of each individual in the new population. The optimal individual in the new population is selected as the local optimal solution \( R_{local_{min}} \), and the cost of the individual is regarded as the local optimal cost \( C_{local_{min}} \). And then compare it to \( C_{global_{min}} \). If \( C_{local_{min}} < C_{global_{min}} \), assign \( C_{local_{min}} \) to \( C_{global_{min}} \) and \( R_{local_{min}} \) to \( R_{global_{min}} \). When \( K \leq K_{max} \), let \( K = K + 1 \), then loop through step (2) \( \sim \) (7). When \( K > K_{max} \), output \( R_{global_{min}} \), \( C_{global_{min}} \).
4. Simulation experiment and analysis

4.1. Verify the validity of the algorithm

There is no database like TSPLIB to verify the MTSP algorithm, so select the special case of MTSP—TSP to verify the effectiveness of the algorithm. In this paper, the simulation tool is MATLAB2020b, the computer processor is Intel I5 processor, the memory is 16G.

| Number | Instance | Known optimal | SA        | GA        | GAISA     |
|--------|----------|---------------|-----------|-----------|-----------|
| 1      | att48    | 33523.7085    | 33831.7295| 33831.7295| 33523.7085 |
| 2      | berlin52 | 7544.3659     | 7849.3952 | 7800.2944 | 7544.3659 |
| 3      | eil51    | 429.9833      | 435.7511  | 437.0277  | 428.8718  |
| 4      | eil76    | 545.3876      | 557.8051  | 565.6292  | 544.3691  |
| 5      | eil101   | 642.3095      | 659.2545  | 684.4918  | 640.2116  |
| 6      | pr76     | 108159.4383   | 110840.2522| 110193.2505| 108159.4383 |
| 7      | pr107    | 44303.0000    | 47499.0627| 47131.1044| 44301.6837 |
| 8      | st70     | 678.5975      | 688.9873  | 682.5802  | 677.1096  |

Eight cases of different sizes and types from the official website of TSPLIB were selected for the experiments. SA, GA and GAISA are run 20 times respectively, and the optimized results are shown in Table 1. The optimal solutions are all reserved for four decimal places.

Compared with the official optimal solution and other algorithms, 8 sets of results are equal to or better than the official optimal solution, and are better than other optimal solutions. It can be proved that the improved algorithm is effective and has some advantages compared with other algorithms.

4.2. Solution and analysis of MTSP

When the number of UAV \( M \) is 5 and the number of task points \( N \) is 100, the optimization results of SA, GA and GA-SA are compared. The optimization results without considering the starting position separately and for single point takeoff at the same position (5, 5) are shown in Figures 5 and 6.

|                    | SA         | GA         | GA-SA      |
|--------------------|------------|------------|------------|
| without considering the starting position | 795.1002  | 871.9606  | 713.7841  |
| single point takeoff from the same position | 939.5552  | 973.0088  | 900.2327  |

Figures 4. Optimization results without considering the starting position.
Figures 5. Optimization results for single point takeoff from the same position.

The above results show that the cost of GAISA in optimizing the MTSP problem is all smaller than that of the GA and SA before improvement. And the optimized routes have no crossover, and the effect is significantly better than the algorithm before improvement, which can prove the superiority of GAISA.

5. Conclusions
In this paper, the multi-UAVs task areas assignment problem is firstly transformed into an MTSP. Then, an improved genetic simulated annealing algorithm is proposed to solve it. Finally, by solving 8 examples in the TSPLIB database and comparing with the pre-improved algorithm and other papers, the effectiveness of GAISA is verified. And then, GAISA is used to solve the set MTSP problem, and the superiority of GAISA in solving the MTSP problem is verified by comparing it with the pre-improvement GA and SA. The improved algorithm not only retains the advantages of fast convergence of GA, but also further enhances the ability to jump out of local optimum. Therefore, GAISA can be used to solve the task area allocation problem of multi-UAVs, and other types of task allocation problems of multi-UAVs. Then, combined with related path planning algorithms and so on, the cooperative search of large-scale areas of multi-UAVs can be accomplished. But we only consider the MTSP problem of the same type of UAV under the condition of mission number equilibrium. The study of the MTSP problem for heterogeneous UAVs can be considered in subsequent studies.

References
[1] Jian, D., Fei, X. og Qifeng, C. (2020) Multi-UAV cooperative search on region division and path planning. Acta Aeronautica et Astronautica Sinica, 41(S1): 149-156.
[2] Gaowei, J. og Jianfeng, W. (2020) Research review of UAV swarm mission planning method. Systems Engineering and Electronics: 1-19.
[3] Delu, W., Xuesong, Z. og Ming, H. (2019) Joint Task Allocation Method Based on Multi-pheromone Ant Colony Algorithm. Journal of China Academy of Electronics and Information Technology, 14(08): 798-807+812.
[4] Xingxing, L., Yang, M., Yanghe, F., Guangping, Z. og Hao, M. (2017) Research on Multi-objective Simulated Annealing Algorithm for Multi-traveling Salesman Problem. Journal of Nanjing Normal University(Natural Science Edition), 40(03): 80-86.
[5] Duofu, Y., Gang, L. og Bing, H. (2019) Multi-chromosome Genetic Algorithm for Multiple Traveling Salesman Problem. Journal of System Simulation, 31(01): 36-42.
[6] Cheikhrouhou, O. og Khoufi, I. (2021) A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy. Computer Science Review, 40: 100369.
[7] Pengzhen, D., Zhenmin, T. og Yan, S. (2014) An object-oriented multi-role ant colony optimization algorithm for solving TSP problem. Control and Decision, 29(10): 1729-1736.
[8] Ebadinezhad, S. (2020) Deaco: Adopting dynamic evaporation strategy to enhance aco algorithm for the traveling salesman problem. Engineering Applications of Artificial Intelligence, 92:
103649.

[9] Zhang, Y., Han, X., Dong, Y., Xie, J., Xie, G. og Xu, X. (2021) A novel state transition simulated annealing algorithm for the multiple traveling salesmen problem. The Journal of Supercomputing: 1-26.

[10] Yuan, S., Skinner, B., Huang, S. og Liu, D. (2013) A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. European Journal of Operational Research, 228(1): 72-82.

[11] Chen, Y., Jia, Z., Ai, X., Yang, D. og Yu, J. (2017) A modified two-part wolf pack search algorithm for the multiple traveling salesmen problem. Applied Soft Computing, 61: 714-725.

[12] Jiang, C., Wan, Z. og Peng, Z. (2020) A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Systems with Applications, 139: 112867.

[13] Qing, H., Yile, W. og Tongwei, X. (2018) Application of improved genetic simulated annealing algorithm in tsp optimization. Control and Decision, 33(02): 219-225.