drones

MDPI

Article

Formation Transformation Based on Improved Genetic Algorithm and Distributed Model Predictive Control

Guanyu Chen, Congwei Zhao, Huajun Gong, Shuai Zhang and Xinhua Wang *

College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China; chenguanyu@nuaa.edu.cn (G.C.); zhaocongwei@nuaa.edu.cn (C.Z.); ghj301@nuaa.edu.cn (H.G.); zhangshuai_0808@163.com (S.Z.)
* Correspondence: xhwang@nuaa.edu.cn

Abstract: In order to solve the problem of multiple aircraft formation transformation to a designated formation, a distributed formation transformation algorithm that decomposes the formation transformation problem into target-matching problems and trajectory-planning problems was studied. According to the actual formation transformation requirements, the target allocation index was proposed, and the improved genetic algorithm which is 23% better than other algorithms was used to achieve target matching. The adaptive cross-mutation probability was designed, and the population was propagated without duplicates by the hash algorithm. The multi-objective algorithm of distributed model predictive control was used to design smooth and conflict-free trajectories for the UAVs in formation transformation, and the trajectory-planning problem was transformed into a quadratic programming problem under inequality constraints. Finally, point-to-point collision-free offline trajectory planning was realized by simulation.

Keywords: distributed model predictive control; formation transformation; target matching; trajectory planning

check for updates

Citation: Chen, G.; Zhao, C.; Gong, H.; Zhang, S.; Wang, X. Formation Transformation Based on Improved Genetic Algorithm and Distributed Model Predictive Control. *Drones* **2023**, *7*, 527. https://doi.org/10.3390/drones7080527

Academic Editor: Sanjay Sharma

Received: 3 July 2023
Revised: 2 August 2023
Accepted: 9 August 2023
Published: 11 August 2023

1. Introduction

UAV formation is a flexible and effective UAV management and organization mode, including formation maintenance, formation change, and mission organization planning execution. In the current advancement of multi-UAV formation research, centralized communication-based multi-UAV formation has yielded significant theoretical outcomes and practical applications in the domains of formation maintenance and cooperative obstacle avoidance [1–5]. However, due to the growing demand for larger formation sizes, the traditional centralized formation approach is inadequate in meeting the requirements of large-scale formation control. With advancements in electronic information technology and network communication, and by harnessing the benefits demonstrated by the distributed formation control method, integrating distributed UAV formation with consistency theory shows promising prospects for future research [6,7].

A well-designed information interaction mechanism is crucial for achieving rapid convergence to the desired formation during multi-UAV operations. It facilitates maintaining formation accuracy, expanding formation size, and enhancing overall redundancy. Currently, three mainstream formation communication strategies exist: distributed [8–10], centralized [11,12], and decentralized [13]. In the distributed strategy, each UAV communicates with its neighboring nodes, eliminating the need for a central node within the formation [14].

Formation control is a crucial aspect of formation algorithms, complementing formation communication. The consistency algorithm, leveraging graph theory, is a widely employed strategy for formation control. For instance, Ren Wei introduced the concept of consistent formation, where information exchange with neighboring nodes ensures consensus among machines on specific cooperative state variables, enabling large-scale distributed cooperative formation [15–17]. Wang et al. proposed an adaptive distributed formation protocol using time-varying gain, obviating the need for a global communication topology map [18]. Deng et al. proposed a multi-step particle swarm optimization algorithm based on swarm intelligence consistency theory for UAV formation trajectory planning, yielding smooth trajectories and broad applicability [19]. Kuriki et al. developed a distributed model predictive control mechanism aligned with the consistency algorithm, enabling cooperative formation with collision avoidance capabilities [20]. Najm et al. proposed an enhanced consistency algorithm for quad-rotor formations utilizing a master–slave communication topology, with the algorithm's controls being fine-tuned using genetic algorithms [21]. The consistency algorithm finds applications in diverse fields such as multi-UAV time-varying formations, sensor networks, and formation task decision making due to its scalability, self-healing properties, and ease of implementation in UAV formation groups.

The purpose of this study is to propose a novel method for UAV formation transformation, which combines an improved genetic algorithm for target matching and a distributed model predictive control strategy for trajectory planning. The significance of this research lies in its potential to provide an effective solution for the challenging and crucial task of UAV formation transformation in various applications of multi-UAV systems, including military reconnaissance, rescue and disaster relief, environmental monitoring, traffic management, and more. This task necessitates the coordination and cooperation of multiple UAVs, considering factors such as communication topology, obstacle avoidance, and control input constraints. Existing methods for UAV formation transformation exhibit limitations, such as high computational complexity, low convergence speed, and duplicate solutions.

The paper presents a novel method for UAV formation transformation that integrates an improved genetic algorithm for target matching and a distributed model predictive control strategy for trajectory planning. An adaptive genetic algorithm is devised to dynamically adjust the crossover probability and variation probability based on the fitness value of individuals, and a hash algorithm is employed to eliminate duplicate solutions. Furthermore, this paper applies an enhanced genetic algorithm in conjunction with a distributed model predictive control strategy to address UAV formation transformation. The feasibility of this approach is then confirmed through simulation and semi-physical experiments.

The rest of the paper is organized as follows: In Section 2, we present the optimization of the target matching using an improved genetic algorithm. In Section 3, we describe the use of distributed model predictive control (DMPC) for collision-free trajectory planning. Section 4 presents simulation results in various scenarios. Section 5 outlines the shortcomings and presents a summary.

2. Objective Assignment Based on Improved Genetic Algorithm

There are two ways to obtain the optimal eye-matching solution from initial to target formation during formation transformation: constructing an exact mathematical model of the UAS and using metaheuristic algorithms. Although most metaheuristic algorithms do not guarantee optimality, they simplify computational effort compared to the exact mathematical model approach and find relatively optimal solutions through finite iterations. Additionally, using metaheuristic algorithms eases deployment and implementation on embedded system boards, making it a common strategy in engineering.

It is possible to turn the target-matching problem into an optimization problem by establishing metrics for time and distance optimization. Optimization problems are a classical application area for genetic algorithms due to their suitability in finding global optimization solutions in large-scale problems containing discrete variables. This approach is faster and provides better quality results than conventional methods. To solve the target point matching, an improved genetic algorithm is used, which includes genetic coding, fitness function calculation, selection, crossover, variation, and outputting an optimal match through a specified number of iterations while meeting set conditions.

2.1. Genetic Operator Coding Design

Selecting the appropriate encoding method for a genetic algorithm is crucial and should depend on the specific problem and solution. Binary and symbolic coding methods are appropriate for solving the target point matching problem. Binary coding offers simple coding and decoding operations, crossover, and easy implementation of mutation operations. Here, each point of the current and desired formations is labeled as 1, 2, 3, $\cdots$, n, and a feasible solution is represented by a two-dimensional matrix. The current queue number serves as the column number while the desired queue number acts as the row number. Each column has a separate match for each row, where the matrix element for the matching row position is 1, and the rest of the row elements in that column are 0. Ten UAVs are arranged in formation as shown in Figure 1.

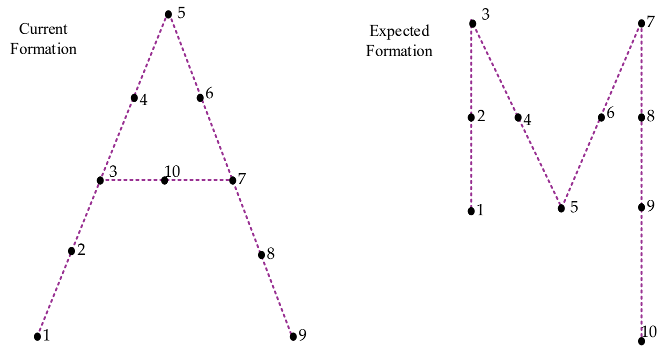

Figure 1. Current formation and desired formation of the aircraft formation.

Assuming that the set of matches corresponding to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 is 7, 3, 2, 8, 9, 4, 1, 10, 6, 5, the corresponding binary code can be expressed in the form of a two-dimensional matrix as follows:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{1}$$

The symbolic encoding in this problem is a direct encoding using each UAV number in the formation 1, 2, 3, $\cdots$, n to construct a numeric string-based genetic operator as a feasible solution. Taking the formation transformation shown as an example, assuming that the set of matches corresponding to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 is 7, 3, 2, 8, 9, 4, 1, 10, 6, 5, the corresponding symbolic encoding can be expressed in the form of a row vector as follows:

A_2 = [7 3 2 8 9 1 10 6 5]; the implicit left-to-right indexes in the row vector match the values corresponding to the indexes, forming a set of target null point matches.

Both of the above encoding methods can be implemented in the formation transformation to solve the target airspace point matching problem, but considering the engineering problem of the actual algorithm, the space used for two-dimensional matrix storage is larger. Assuming that the number of populations in the genetic algorithm is m and the number of formation UAVs is n, the storage space occupied by a single genetic operator with binary encoding is n^2 and the overall space complexity is $O(mn^2)$; the storage space occupied by a single genetic operator with symbolic encoding is n and the overall space complexity is $O(mn)$. Considering the small storage space of the embedded chip SDRAM, the symbolic encoding method is more convenient for the deployment of the algorithm on the embedded system board, which is conducive to the engineering of the algorithm, and the symbolic encoding method is more intuitive for the subsequent cross-variance operation, so the symbolic encoding method is chosen for this problem.

2.2. Adaptation Function Analysis

Given that the genetic algorithm is a stochastic global search and optimization method that mimics the evolutionary mechanism of organisms in nature, it is crucial to establish evaluation metrics to score individuals in the population and implement survival of the fittest. In this problem, a fitness function is constructed using the total distance metric for formation transformation, as well as time and the number of trajectory crossings as optimization objectives.

(1) Formation change flight distance constraint

In the process of formation transformation, the total distance from the current formation to the desired formation is the metric that we need to optimize, and the smaller the distance sum, the better. Suppose there are n UAVs, the position of the UAV in the current formation is (x_i, y_i, z_i), the corresponding matching target point in the desired formation is marked as (a_i, b_i, c_i), and the total distance cost is d_{sum}; then, we have the following equation:

$$d_{sum} = \sum_{i=1}^{n}\left(\sqrt{(x_i - a_i)^2 + (y_i - b_i)^2 + (z_i - c_i)^2}\right) \tag{2}$$

(2) UAV maximum flight time constraint

Assuming that each UAV has the same speed v during the formation change, the time to complete the formation change depends on the UAV that takes the longest time to reach the specified position, and the maximum flight time is recorded as $t_{\max}$, the distance from each UAV to the corresponding matching point is recorded as d_i, and the following equations are obtained:

$$\begin{cases} d_i = \sqrt{(x_i - a_i)^2 + (y_i - b_i)^2 + (z_i - c_i)^2} \\ t_{\max} = \max\left\{\frac{d_1}{v}, \frac{d_2}{v}, \frac{d_3}{v}, \dots, \frac{d_i}{v}, \dots, \frac{d_n}{v}\right\} \end{cases} \tag{3}$$

(3) Constraint on the number of trajectory crossings

To subsequently reduce the difficulty of the collision avoidance algorithm, target matching with the lowest possible number of trajectory crossings should be considered as much as possible. Suppose the two position points of the current formation are A and C, and the expected transformed position points are B and D. Let the coordinates of the four points A, B, C, and D be (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), and (x_4, y_4, z_4). The parametric equations of the line segments AB and CD are

$$\begin{cases} x = x_1 + (x_2 - x_1)t \\ y = y_1 + (y_2 - y_1)t \\ z = z_1 + (z_2 - z_1)t \end{cases}, \begin{cases} x = x_3 + (x_4 - x_3)t' \\ y = y_3 + (y_4 - y_3)t' \\ z = z_3 + (z_4 - z_3)t' \end{cases} \tag{4}$$

where $t \in [0, 1]$ The parametric equation of the line CD is $t^{'} \in [0, 1]$. Now, the question of whether the two trajectories intersect translates into whether there exists $t_1, t_2 \in [0, 1]$ such that the following system of equations holds:

$$\begin{cases} x_1 + (x_2 - x_1)t_1 = x_3 + (x_4 - x_3)t_2 \\ y_1 + (y_2 - y_1)t_1 = y_3 + (y_4 - y_3)t_2 \\ z_1 + (z_2 - z_1)t_1 = z_3 + (z_4 - z_3)t_2 \end{cases} \tag{5}$$

Equation (5) can be expressed in matrix form as follows:

$$a\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = b \tag{6}$$

$$a = \begin{bmatrix} x_2 - x_1 & x_3 - x_4 \\ y_2 - y_1 & y_3 - y_4 \\ z_2 - z_1 & z_3 - z_4 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}, b = \begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{7}$$

When $a^T a$ is invertible, at which point there is an intersection or dissimilarity of the two line segments, solving the system of equations using the least squares method yields

$$\begin{cases} A_{11} = a_{11}^2 + a_{21}^2 + a_{31}^2 \\ A_{12} = a_{11}a_{12} + a_{21}a_{22} + a_{31}a_{32} \\ A_{21} = A_{12} \\ A_{22} = a_{12}^2 + a_{22}^2 + a_{32}^2 \\ B_1 = a_{11}b_1 + a_{21}b_2 + a_{31}b_3 \\ B_2 = a_{12}b_1 + a_{22}b_2 + a_{32}b_3 \\ t_1 = -(A_{12}B_2 - A_{22}B_1)/(A_{11}A_{22} - A_{12}A_{21}) \\ t_2 = (A_{11}B_2 - A_{21}B_1)/(A_{11}A_{22} - A_{12}A_{21}) \end{cases} \tag{8}$$

where $A_{11}A_{22} - A_{12}A_{21} \neq 0$. After solving for t_1, t_2, determine if its value is within the interval [0, 1], and then further determine if the following equation holds:

$$\begin{cases} |(x_1 + (x_2 - x_1)t_1) - (x_3 + (x_4 - x_3)t_2)| < eps \\ |(y_1 + (y_2 - y_1)t_1) - (y_3 + (y_4 - y_3)t_2)| < eps \\ |(z_1 + (z_2 - z_1)t_1) - (z_3 + (z_4 - z_3)t_2)| < eps \end{cases} \tag{9}$$

where *eps* is the set tolerance; if (9) holds, then the line segment AB intersects the line segment CD; otherwise, the line segment AB is opposite to CD. When $a^T a$ is not invertible, then the two line segments are parallel, and the trajectories do not have the possibility of intersection.

Using the above derivation, we can determine whether the assigned trajectories are crossed or not, and we can achieve the statistics of the number of crossings for all trajectories with the time complexity of $O(n^2)$, and the number of crossings is m_{cross}. Considering the above optimization indexes, the following cost function can be constructed:

$$Fitness\ (cur, tar, v, chrom) = k - a_1 d_{sum} - a_2 t_{\max} - a_3 m_{corss} \tag{10}$$

where k is the bounding value constant, v is the formation transformation UAV flight speed, a_1 is the total distance scaling factor, a_2 is the maximum time scaling factor, the scaling factor a_3 is used to maintain uniform order of magnitude for the medium distance cost, the maximum time cost, and the trajectory crossing number cost. $cur_{n\times 3}$ stores the coordinates 1, 2, 3, $\cdots$, n corresponding to the current numbered queue. $tar'_{n\times 3}$ stores the coordinates 1, 2, 3, $\cdots$, n corresponding to the desired numbered queue. *chrom* is a set of feasible symbolic encoding solutions. $tar'_{n\times 3}$ can be adjusted based on the matching information provided by *chrom*, resulting in a two-dimensional matrix that stores the corresponding matched coordinates. This matrix is denoted as $tar'_{n\times 3}$. To solve d_{sum}, the Euclidean distance between the corresponding rows of $cur_{n\times 3}$ and $tar'_{n\times 3}$ is calculated individually and then accumulated.

2.3. Selection Operator and Hash De-Duplication Design

The selection strategy used in this paper utilizes the roulette selection algorithm. Firstly, the fitness of individuals in the population is calculated. Then, individuals are selected at random based on their fitness, using a proportional range to determine the likelihood of selection. As a result, individuals with higher fitness levels are more likely to be selected and retained during the evolutionary iteration process.

If an individual is denoted as i and its fitness is f_i, the probability that it will be selected is expressed as

$$P_i = \frac{f_i}{\sum\limits_{i=1}^{n} f_i} \tag{11}$$

High-fitness individuals are more likely to be selected. The retention method for elite individuals is utilized in the initial selection strategy, where the top 5% of high-fitness individuals in the population are retained, and the remaining individuals are randomly selected using the roulette algorithm.

After the roulette strategy selection, some individuals with higher fitness will be selected repeatedly, which may lead to the individuals with higher fitness occupying the population rapidly in the subsequent iterative solution process, resulting in the local optimal solution at the initial iteration stage. Therefore, steady-state selection without a repeat string should be realized. In this section, the Rabin–Karp string hashing algorithm is used to realize the non-repeated string judgment of genetic operators. Let the symbolic code of each feasible solution be *chrom*; *chrom* is a sequence consisting of numbers a_0, a_1, a_2, $\cdots$, a_{n-1}, which can be treated as a $\max\{a_i\} + 1$ progressive number, and then the corresponding value of this sequence can be calculated by the following equation:

$$f(chrom) = \sum_{i=0}^{n-1} (\max\{a_i\} + 1)^{i} \times a_i \tag{12}$$

Considering that the hash value obtained from the calculation will be very large, it is necessary to mold the result of the calculation $f(chrom)$ on a large prime number P and map this value to the int range. Two different genetic operators will have different encoding values before modulo, but the encoding values may be the same after modulo, because the hash encoding method is not single-shot, and there is a possibility of hash collision. However, according to the birthday paradox, there is a high probability of collision when a sequence of $O\left(\sqrt{\text{mod}}\right)$ numbers is generated randomly, so the modulus can be set as large as possible to avoid collision. So, the actual hash function is modified as follows:

$$f(chrom) = \sum_{i=0}^{n-1} (\max\{a_i\} + 1)^{i} \times a_i (\text{mod} P) \tag{13}$$

The Rabin–Karp string hashing algorithm can be employed to eliminate duplicate genetic operators in the roulette wheel selection strategy. Subsequently, the excluded genetic operators can be replenished by randomly initializing new genetic operators.

2.4. Crossover and Variational Operator Design

A crossover operation involves permuting and reconstructing the local structure of two parent operators to create new individuals. In the iterative process of a genetic algorithm, a crossover operation has a probability of producing new and improved individuals.

In this section, we utilize partial cross-matching. The following Figures 2 and 3 depicts the encoding pre- and post-exchange.

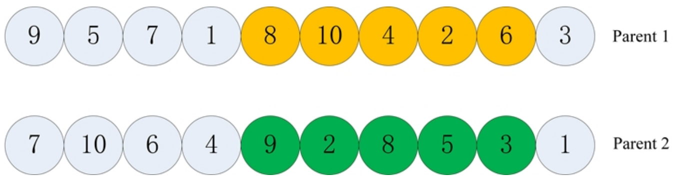

Figure 2. Pre-crossover coding.

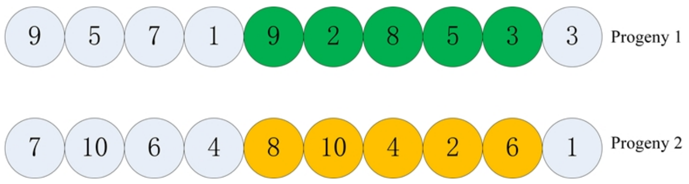

Figure 3. Post-crossover coding.

The operator crossover generation strategy is specific to the following steps: Firstly, we save the numbers in the sequence for each genetic operator using a hash table. Then, we perform a two-point crossover for the genetic operator. Next, we traverse the number sequence for the other genetic operator, removing numbers from the corresponding hash table if present and marking the corresponding number in the genetic operator as visited. Finally, we traverse the exchanged fragment of the genetic operator and sequentially remove and replace the data in the hash table of the genetic operator with the data removed from the unvisited position in the hash table. The resulting modified operator is shown in Figure 4.

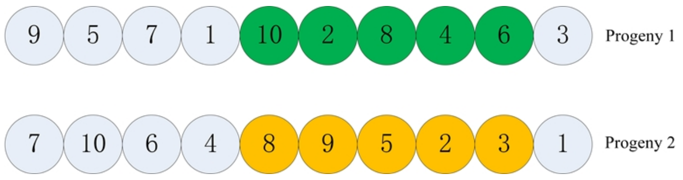

Figure 4. Modified crossover operator.

The variation operation often accompanies the selection and crossover operations to prevent the genetic algorithm from converging to a local optimum too early in the iterative process. Variation is carried out on a single chromosome; we employ the reversal variation, depicted in Figure 5.

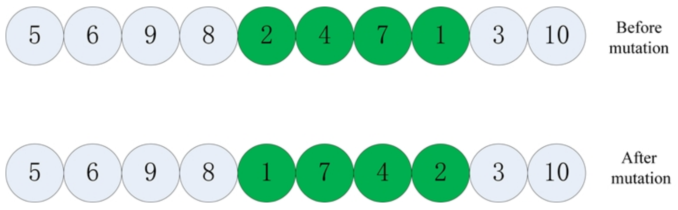

Figure 5. Variational operator.

The operator crossover probability P_c and variance probability P_m are fixed values in conventional genetic algorithms, and these two parameters directly affect the convergence of the algorithm. Based on this, an adaptive genetic algorithm is proposed that can dynamically modify the crossover probability P_c and the variation probability P_m. For individuals whose fitness is higher than the average fitness value of the population, P_c and P_m with smaller probability values are constructed so that the individuals have a higher probability of being retained.

$$P_c = \begin{cases} P_{c1} - \frac{(P_{c1} - P_{c2})(f_{\max} - f')}{f_{\max} - f_{avg}}, & f \geq f_{avg} \\ P_{c1}, & f < f_{avg} \end{cases} \tag{14}$$

$$P_m = \begin{cases} P_{m1} - \frac{(P_{m1} - P_{m2})(f_{\max} - f)}{f_{\max} - f_{avg}}, & f \geq f_{avg} \\ P_{m1}, & f < f_{avg} \end{cases} \tag{15}$$

where $f_{\max}$ is the maximum fitness value in the population; f_{avg} is the average fitness value of the population per generation; $f^{'}$ is the larger fitness value of the two individuals to be crossed; f is the fitness value of the individual to be mutated; and P_{c1}, P_{c2}, P_{m1}, and P_{m2} are constants. The Figure 6 is the overall flow chart of the improved genetic algorithm.

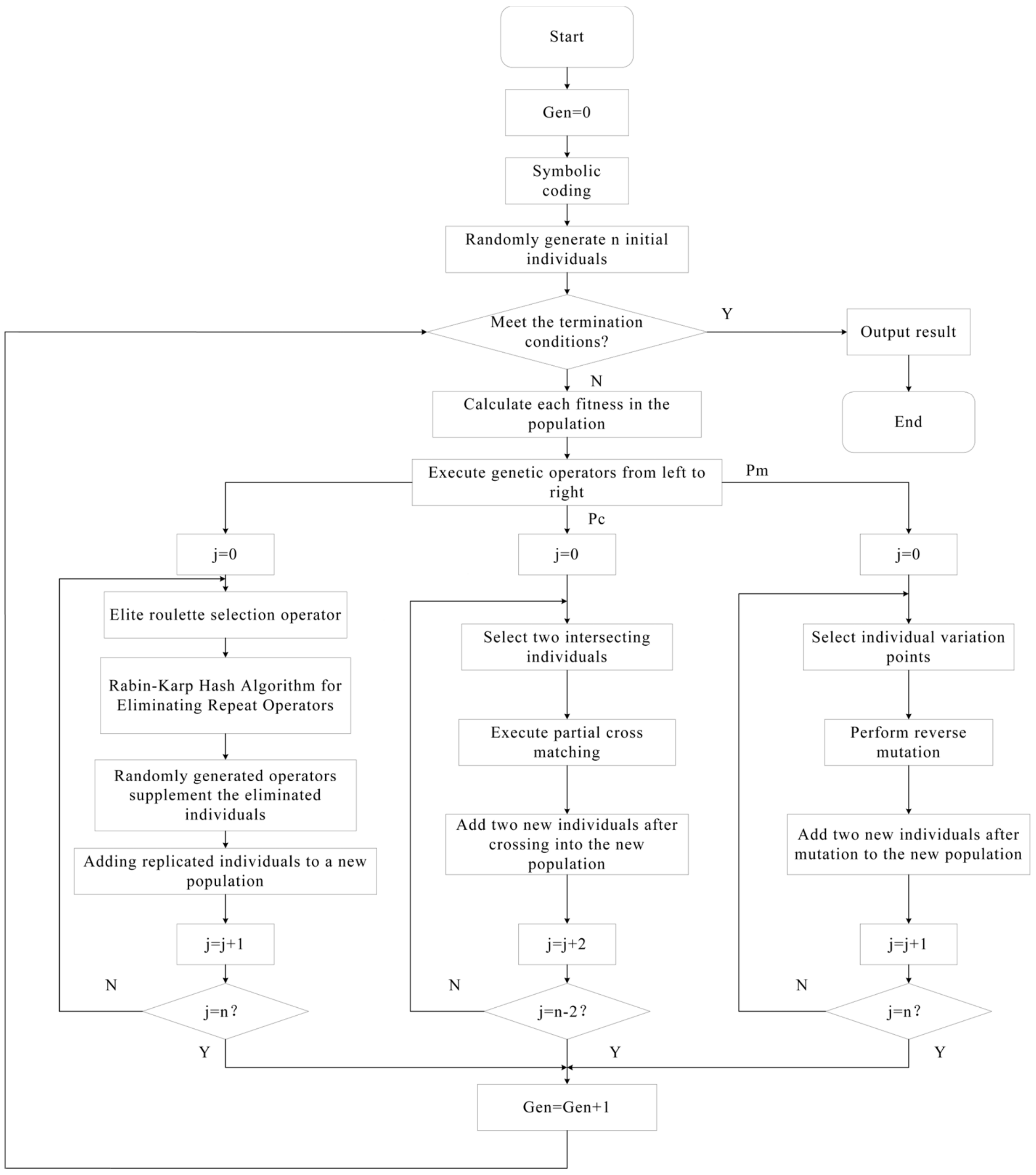
Start
Gen=0
Symbolic coding
Randomly generate n initial individuals
Meet the termination conditions?
Y
Output result
End
N
Calculate each fitness in the population
Execute genetic operators from left to right
Pm
Pc
j=0
j=0
j=0
Elite roulette selection operator
Rabin-Karp Hash Algorithm for Eliminating Repeat Operators
Randomly generated operators supplement the eliminated individuals
Adding replicated individuals to a new population
Select two intersecting individuals
Execute partial cross matching
Add two new individuals after crossing into the new population
Select individual variation points
Perform reverse mutation
Add two new individuals after mutation to the new population
j=j+1
j=j+2
j=j+1
N
N
N
j=n?
j=n-2?
j=n?
Y
Y
Y
Gen=Gen+1

Figure 6. Overall flow chart of the improved genetic algorithm.

3. Design of Trajectory-Planning Strategy Based on Distributed Model Prediction

3.1. The Basic Principle of MPC Algorithm

The model predictive control (MPC) algorithm is a model to predict the performance of the system in a future time period for optimal control, mostly used for digit control, and is a discrete state space expression.

Suppose the following discrete state space expression is available:

$$\mathrm{x}(k+1) = A\mathrm{x}(k) + B\mathrm{u}(k) \tag{16}$$

Let the predicted system input at time k for the system at time $k + i$ be denoted as $\mathrm{u}(k = i \mid k)$ and the predicted system state vector k be denoted as $\mathrm{x}(k = i \mid k)$. The prediction horizon of the system is N. Considering the initial state of the system as x_k, the following recursive equation can be obtained:

$$\begin{aligned}
\mathrm{x}(k \mid k) &= \mathrm{x}_k \\
\mathrm{x}(k+1 \mid k) &= A\mathrm{x}_k + B\mathrm{u}(k \mid k) \\
\mathrm{x}(k+2 \mid k) &= A^2\mathrm{x}_k + AB\mathrm{u}(k \mid k) + B\mathrm{u}(k+1 \mid k) \\
&\vdots \\
\mathrm{x}(k+N \mid k) &= A^N\mathrm{x}_k + A^{N-1}B\mathrm{u}(k \mid k) + \cdots + B\mathrm{u}(k+1 \mid k)
\end{aligned} \tag{17}$$

Let X_k be the column vector consisting of the system state variables at $N + 1$ moments and U_k be the column vector consisting of the system predicted input variables at N moments; the above equation can be written in the following matrix form:

$$X_k = M\mathrm{x}_k + CU_k \tag{18}$$

where $X_k = \begin{bmatrix} \mathrm{x}(k|k) \\ \mathrm{x}(k+1|k) \\ \mathrm{x}(k+2|k) \\ \vdots \\ \mathrm{x}(k+N|k) \end{bmatrix}$, $M = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}$, $U_k = \begin{bmatrix} \mathrm{u}(k|k) \\ \mathrm{u}(k+1|k) \\ \vdots \\ \mathrm{u}(k+N-1|k) \end{bmatrix}$, x_k is the initial state variables, and $C = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}$.

For model predictive control, the commonly used optimization strategy is quadratic programming (QP). The general form of quadratic programming is as follows:

$$f = Z^TQZ + C^TZ \tag{19}$$

where Z^TQZ is quadratic and C^TZ is linear for the model predictive control algorithm in which the cost function needs to be reduced to a general quadratic form for optimal control. To simplify the derivation, this paper treats the reference state quantity as 0, so the error state quantity is the system state quantity, so the following cost function can be obtained:

$$\begin{aligned}
J = &\sum_{i=0}^{N-1} \left(\mathrm{x}(k+i|k)^T Q \mathrm{x}(k+i|k) + \mathrm{u}(k+i|k)^T R \mathrm{u}(k+i|k) \right) \\
&+ \mathrm{x}(k+N|k)^T F \mathrm{x}(k+N|k)
\end{aligned} \tag{20}$$

where Q, R, and F are diagonal matrices. The above cost function can be written in the following form:

$$J = X_k\overline{Q}X_k + U_k^T\overline{R}U_k \tag{21}$$

where $\overline{Q} = \begin{bmatrix} Q & & & & \\ & Q & & & \\ & & Q & & \\ & & & \ddots & \\ & & & & F \end{bmatrix}$ and $\overline{R} = \begin{bmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{bmatrix}$. Substituting Formula (20) into the previous Equation (21), we can further expand it, resulting in the following:

$$\begin{aligned}
J &= (Mx_k + CU_k)^T\overline{Q}(Mx_k + CU_k) + U_k^T\overline{R}U_k \\
&= \left(x_k^TM^T + U_k^TC^T\right)\overline{Q}(Mx_k + CU_k) + U_k^T\overline{R}U_k \\
&= \left(x_k^TM^T\overline{Q} + U_k^TC^T\overline{Q}\right)(Mx_k + CU_k) + U_k^T\overline{R}U_k \\
&= x_k^TM^T\overline{Q}Mx_k + x_k^TM^T\overline{Q}CU_k + U_k^TC^T\overline{Q}Mx_k + U_k^TC^T\overline{Q}CU_k + U_k^T\overline{R}U_k \\
&= x_k^TM^T\overline{Q}Mx_k + 2x_k^TM^T\overline{Q}CU_k + U_k^T\left(C^T\overline{Q}C + \overline{R}\right)U_k
\end{aligned} \tag{22}$$

Let $M^T\overline{Q}M$ be G, $M^T\overline{Q}C$ be E, and $C^T\overline{Q}C + \overline{R}$ be H, so the above equation can be reduced to

$$J = U_k^T H U_k + 2x_k^T E U_k + x_k^T G x_k \tag{23}$$

This transforms the optimization problem of model predictive control into a quadratic programming problem.

3.2. Single UAV Prediction Modeling

An improved genetic algorithm can be implemented to determine the optimal target matching during the formation transformation process and determine the destination point of each UAV. In this section, we conduct conflict-free trajectory planning to solve the UAV point-to-point transformation problem. For trajectory generation, we employ a distributed model predictive control (DMPC)-based multi-objective algorithm.

The UAV uses a second-order integrator model. Assume that at time step k, the position vectors in the three directions of x, y, z for UAVi are represented as $d_i[k]_{3\times1}$, the velocity vector is represented as $v_i[k]_{3\times1}$, and the acceleration vector is represented as $a_i[k]_{3\times1}$. If the acceleration is used as the system input and the discretization step is s, then the dynamic model of the UAVi is as follows:

$$\begin{aligned} d_i[k+1] &= d_i[k] + s v_i[k] + \tfrac{s^2}{2} a_i[k] \\ v_i[k+1] &= v_i[k] + s a_i[k] \end{aligned} \tag{24}$$

At each discretized time slice, an optimal input sequence can be obtained based on the controlled object's model, within the given prediction interval. The resulting first input is applied to the actual system, and the current system state variables are measured to initiate the optimal control problem in the subsequent moment. This approach enables rolling control optimization. Considering the current discrete time slice sequence as the first k_t, UAVs are deployed in a distributed manner, iteratively optimizing trajectory generation by sharing predicted state sequences with neighboring UAVs. Each UAV calculates the optimal input sequence considering collision possibilities, system constraints, and motion model parameters. The first input of the obtained sequence is applied to the UAV, followed by a collective movement to the next time step while sharing state variables. The prediction model of the UAVi is as follows:

$$\begin{bmatrix} \hat{d}_i[k+1|k_t] \\ \hat{v}_i[k+1|k_t] \end{bmatrix} = \begin{bmatrix} I_3 & sI_3 \\ 0_3 & I_3 \end{bmatrix} \begin{bmatrix} \hat{d}_i[k|k_t] \\ \hat{v}_i[k|k_t] \end{bmatrix} + \begin{bmatrix} (s^2/2) I_3 \\ sI_3 \end{bmatrix} \hat{a}_i[k|k_t] \tag{25}$$

By choosing acceleration as the model input, the above equation can be reduced to the following form:

$$\hat{x}_i[k+1|k_t] = A\hat{x}_i[k|k_t] + B\hat{u}_i[k|k_t] \tag{26}$$

where $\hat{x}_i \in R^6$, $A \in R^{6\times6}$, $B \in R^{6\times6}$, and $\hat{u}_i \in R^3$. The state of the system at time k_t is as follows:

$$d_i = A_0 X_{0,i} + \Lambda U_i \tag{27}$$

where $X_{0,i} = x_i[k_t]$, $d_i \in R^{3K}$, $U_i \in R^{3K}$, $\Lambda = \begin{bmatrix} \Psi B & 0_3 & \cdots & 0_3 \\ \Psi AB & \Psi B & \cdots & 0_3 \\ \vdots & \vdots & \ddots & \vdots \\ \Psi A^{K-1}B & \Psi A^{K-2}B & \cdots & \Psi B \end{bmatrix}$, $\Lambda \in R^{3K\times3K}$, $\Psi = \begin{bmatrix} I_3 & 0_3 \end{bmatrix}$, and $A_0 = \begin{bmatrix} (\Psi A)^T & (\Psi A^2)^T & \cdots & (\Psi A^K)^T \end{bmatrix}^T \in R^{3K\times6}$.

3.3. Establishment of the Cost Function of the DMPC Algorithm

The goal of optimal control is to minimize the cost function. In this section, the cost function primarily comprises error tracking, control input, and input variation terms. Our aim is to find the optimal input sequence that minimizes the cost function.

(1) Error tracking term

The error tracking term is used to better track the reference trajectory. The smaller the error tracking, the better the actual flight fits the target planning curve. The form of the error tracking term is as follows:

$$e_i = \sum_{K-\kappa}^{K} \left\| \hat{d}_i[k|k_t] - d_{O,i} \right\|_2 \tag{28}$$

where $d_{o,i}$ is the final desired target location point and $K-\kappa$ is the discretization step from K to κ. Using (23), the error tracking term can be written in the following quadratic form:

$$J_{e,i} = U_i^T \left(\Lambda^T \widetilde{Q} \Lambda \right) U_i - 2 \left(d_{O,i}^T \widetilde{Q} \Lambda - (A_0 X_{0,i})^T \widetilde{Q} \Lambda \right) U_i \tag{29}$$

where $\widetilde{Q} \in R^{3K \times 3K}$ is a positive definite diagonal matrix and the elements on the diagonal indicate the error weights corresponding at each sampling point, when $\kappa = 1$, corresponding to $\widetilde{Q} = diag(0_3, \cdots, Q)$; Q is also a positive definite diagonal matrix, $Q \in R^{3\times 3}$. The larger κ is, the more drastically the UAV moves toward the target point and may appear to overshoot at the end point.

(2) Control input items

The smaller the input control term, the smaller the overall energy consumption of the system. Based on the derivation in the previous section, the term can be written in the following quadratic form:

$$J_{c,i} = U_i^T \widetilde{R} U_i \tag{30}$$

where $\widetilde{R} \in R^{3K \times 3K}$ is a positive definite diagonal matrix; the elements on the diagonal represent the corresponding input weights at each sampling point; and $\widetilde{R} = diag(R, \cdots, R)$, $R \in R^{3\times 3}$, and R are also positive definite diagonal matrices.

(3) Input change items

The smaller the input variation term, the smoother the control will be, and no violent oscillations or large changes in trajectory will occur. The cost expression of the input variation term is as follows:

$$\Delta_i = \sum_{k=0}^{K-1} \| \hat{u}_i[k|k_t] - \hat{u}_i[k-1|k_t] \|_2 \tag{31}$$

The above expression can be rewritten as the following quadratic form:

$$J_{\delta,i} = U_i^T \left(\Gamma^T \widetilde{S} \Gamma \right) U_i - 2 \left(U_{ii}^T \widetilde{S} \Gamma \right) U_i \tag{32}$$

where $\widetilde{S} = diag(S, \cdots, S) \in R^{3K \times 3K}$ is a positive definite diagonal array, and the elements on the diagonal represent the weights of the input variables corresponding to $S \in R^{3\times 3}$, $U_{ii} = \begin{bmatrix} u_i[k_t-1]^T & 0_{3\times 1}^T & \cdots & 0_{3\times 1}^T \end{bmatrix}^T \in R^{3K}$, and $\Gamma = \begin{bmatrix} I_3 & 0_3 & 0_3 & \cdots & 0_3 & 0_3 \\ -I_3 & I_3 & 0_3 & \cdots & 0_3 & 0_3 \\ 0_3 & -I_3 & I_3 & \cdots & 0_3 & 0_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_3 & 0_3 & 0_3 & \cdots & -I_3 & I_3 \end{bmatrix}$ at each sampling point, and summing the above three surrogate values gives the final cost function:

$$J_i(U_i) = J_{e,i} + J_{c,i} + J_{\Delta,i} \tag{33}$$

3.4. Collision Avoidance Constraints and Physical Constraint Design

We can use the prediction property of DMPC to detect collision trajectories and impose constraints to avoid collisions. The constraint condition for UAV i to avoid collision with UAV j at time step $k_{C,i}$ is as follows:

$$\vartheta_{ij} = \left\| \mathrm{M}^{-1}\left(\hat{d}_i[k_{C,i}|k_t-1] - \hat{d}_j[k_{C,i}|k_t-1]\right)\right\|_n \geq l_{\min} \tag{34}$$

$l_{\min}$ denotes the minimum distance between the UAVi and the UAVj in the xy plane. M denotes the scaling matrix, which is expanded in the following form:

$$\mathrm{M} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \tag{35}$$

Generally, take $a = b = 1$, $c > 1$; thus, the minimum distance constraint required at the z axis is $l_{z,\min} = cl_{\min}$. The solution time is at k_t, and each UAV has only the information of other UAVs at the time of $k_t - 1$; then, the collision prediction occurs at the time of $k_{c,j} + k_t - 1$, and the set of collision constraints is

$$\Omega_i = \left\{ j \in \{1, \cdots, N\} \middle| \vartheta_{ij} < f(l_{\min}), i \neq j \right\} \tag{36}$$

If a collision is detected, a new input sequence must be calculated by including a collision constraint. Considering that the optimization problem can sometimes be infeasible, a relaxed obstacle avoidance constraint is developed as follows:

$$\left\| \mathrm{M}^{-1}\left(\hat{d}_i[k_{C,i}-1|k_t] - \hat{d}_j[k_{c,i}|k_t-1]\right)\right\|_n \geq l_{\min} + \varepsilon_{ij} \tag{37}$$

where ε_{ij} is a relaxation factor less than 0. We can use the first-order Taylor expansion at $k_{C,j} + k_t - 1$ of the UAVi to make an approximation to the above equation, with the expansion point taken as $\hat{d}_i[k_{C,i}-1|k_t]$, and the approximate expansion process is as follows: let $f(x,y,z) = \left\| \mathrm{M}^{-1}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - 2\hat{d}_j[k_{C,i}|k_t-1] \right)\right\|_n$; the column vector parametrization is defined as $\left\| \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right\|_n = (x^n + y^n + z^n)^{\frac{1}{n}}$, $\mathrm{M}^{-1} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & c^{-1} \end{bmatrix}$. The partial derivative of $f(x,y,z)$ gives the following results:

$$\begin{cases} \frac{\partial}{\partial x} f(x,y,z) = f^{1-n} \times (x - \hat{d}_{j,1}[k_{C,i}|k_t-1])^{n-1} \\ \frac{\partial}{\partial y} f(x,y,z) = f^{1-n} \times (y - \hat{d}_{j,2}[k_{C,i}|k_t-1])^{n-1} \\ \frac{\partial}{\partial z} f(x,y,z) = c^{-n} f^{1-n} \times (z - \hat{d}_{j,3}[k_{C,i}|k_t-1])^{n-1} \end{cases} \tag{38}$$

Taking the independent variable as $x = \hat{p}_i[k_{c,i}-1|k_t]$ and the expansion point as $x_0 = \hat{d}_i[k_{C,i}\Big|k_t-1]$, by Taylor expansion, we have the following:

$$\begin{aligned} f(x) &\approx f(x_0) + f'(x_0)(x-x_0)^T \\ &= f(x_0) + [f(x_0)]^{1-n}\mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t-1]\right)^T (x-x_0) \end{aligned} \tag{39}$$

It can be inferred from (34) that

$$f(x_0) = \vartheta_{ij} = \left\| \mathrm{M}^{-1}\left(\hat{d}_i[k_{C,i}|k_t-1] - \hat{d}_j[k_{C,i}|k_t-1]\right)\right\|_n \tag{40}$$

Substituting the above equation into (39) yields

$$\begin{aligned} f(x) &\approx \vartheta_{ij} + \left(\vartheta_{ij}\right)^{1-n}\mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T (x - x_0) \\ &= \vartheta_{ij} + \left(\vartheta_{ij}\right)^{1-n}\mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x \\ &-\left(\vartheta_{ij}\right)^{1-n}\mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x_0 \end{aligned} \tag{41}$$

It can be inferred from (37) that

$$f(x) \geq l_{\min} + \varepsilon_{ij} \tag{42}$$

Substituting into the above equation, we obtain

$$\begin{aligned} f(x) &\approx \vartheta_{ij} + \left(\vartheta_{ij}\right)^{1-n}\mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x \\ &-\left(\vartheta_{ij}\right)^{1-n}\mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{c,i} \mid k_t - 1]\right)^T x_0 \geq l_{\min} + \varepsilon_{ij} \end{aligned} \tag{43}$$

Further simplification yields

$$\begin{gathered} \vartheta_{ij}^n + \mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x - \mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x_0 \\ \geq l_{\min}\xi_{ij}^{n-1} + \varepsilon_{ij}\vartheta_{ij}^{n-1} \\ \Rightarrow \mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x - \varepsilon_{ij}\vartheta_{ij}^{n-1} \\ \geq l_{\min}\vartheta_{ij}^{n-1} - \vartheta_{ij}^n + \mathrm{M}^{-n}\left(x_0 - \hat{d}_j[k_{C,i} \mid k_t - 1]\right)^T x_0 \end{gathered} \tag{44}$$

Obtain

$$v_{ij}^T \hat{d}_i[k_{C,i}|k_t] - \varepsilon_{ij}\vartheta_{ij} \geq \rho_{ij} \tag{45}$$

where $v_{ij} = \mathrm{M}^{-n}\left(\hat{d}_i[k_{C,i}|k_t - 1] - \hat{d}_j[k_{C,i}|k_t - 1]\right)^{n-1}$. As physical constraints, this paper considers acceleration as well as position sequences, and the following constraints can be obtained: $a_{\min} \leq a_i[k] \leq a_{\max}$ and $d_{\min} \leq d_i[k] \leq d_{\max}$. Expressing them in matrix form, we have

$$\begin{cases} D_{\min} = \left[d_{\min}^T \cdots d_{\min}^T\right]^T, D_{\max} = \left[d_{\max}^T \cdots d_{\max}^T\right]^T \\ U_{\min} = \left[a_{\min}^T \cdots a_{\min}^T\right]^T, U_{\max} = \left[u_{\max}^T \cdots u_{\max}^T\right]^T \\ D_{\min} - A_0 X_{0,i} \leq \Lambda U_i \leq D_{\max} - A_0 X_{0,i} \\ U_{\min} \leq U_i \leq U_{\max} \end{cases} \tag{46}$$

By listing all the constraints for the K prediction interval and rewriting them in matrix form, the final inequality constraint is obtained as follows:

$$A_{in} U_i \leq b_{in} \tag{47}$$

In order to transform the collision constraint into an affine function of the decision variables, we expand the previously obtained formula by introducing column vectors, $E_i \in R^{n_{c,i}}$, and $n_{c,i}$ denoting the number of neighboring UAVs within the collision radius, and E_i can be expressed as follows:

$$E_i = \begin{bmatrix} \vdots \\ \varepsilon_{ij} \\ \vdots \end{bmatrix} \tag{48}$$

where $i \in \Omega_i = \{j \in \{1, \cdots, N\} | \xi_{ij} < f(r_{\min}), i \neq j\}$, $\mu_{ij}^T \Lambda U_i - \varepsilon_{ij}\xi_{ij} \geq \rho_{ij} - \mu_{ij}^T A_0 X_{0,i}$, $\mu_{ij} \in R^{3K}$, and $\mu_{ij} = \left[0^T_{3(k_e,i-1)} v^T_{ij} 0^T_{3(K-k_{e,i})}\right]^T$. Listing the above inequalities in order by $i = 1, 2, \cdots 3K + n_{c,i}$ and writing them in matrix form yields.

$$A_{coll} \mathrm{Y}_i \leq b_{coll} \tag{49}$$

where $\mathrm{Y}_i = \begin{bmatrix} U_i \\ E_i \end{bmatrix}$; the final cost function with obstacle avoidance information is as follows:

$$J_{aug,i}(\mathrm{Y}_i) = J(U_i) + \mathrm{Y}_i^T H_{\varepsilon,i} \mathrm{Y}_i - f_{\varepsilon,i}^T \mathrm{Y}_i \tag{50}$$

where $f_{\varepsilon,i} = \varsigma \begin{bmatrix} 0^T_{3K\times 1} & 1^T_{n_{c,i}\times 1} \end{bmatrix}^T$ and $H_{\varepsilon,i} = \tau \begin{bmatrix} 0_{3K\times 3K} & 0_{3K\times n_{c,i}} \\ 0_{n_{c,i}\times 3K} & I_{n_{c,i}} \end{bmatrix}$.

4. Simulation Results and Analysis

4.1. Target Assignment Algorithm Simulation Verification

According to simulation comparison results in Figure 7, the improved genetic algorithm demonstrates the smallest total distance traveled. In terms of solving time, the Hungarian algorithm experiences slower solving speed due to the increasing matrix dimension with a growing number of UAVs, while the improved genetic algorithm becomes faster. Additionally, regarding the number of trajectory crossings, the improved genetic algorithm maintains fewer crossings compared to the ordinary genetic algorithm and the Hungarian algorithm, which do not consider this element and result in more random trajectory crossings. In conclusion, the proposed improved genetic algorithm exhibits excellent performance in solving the queue transformation goal assignment problem.

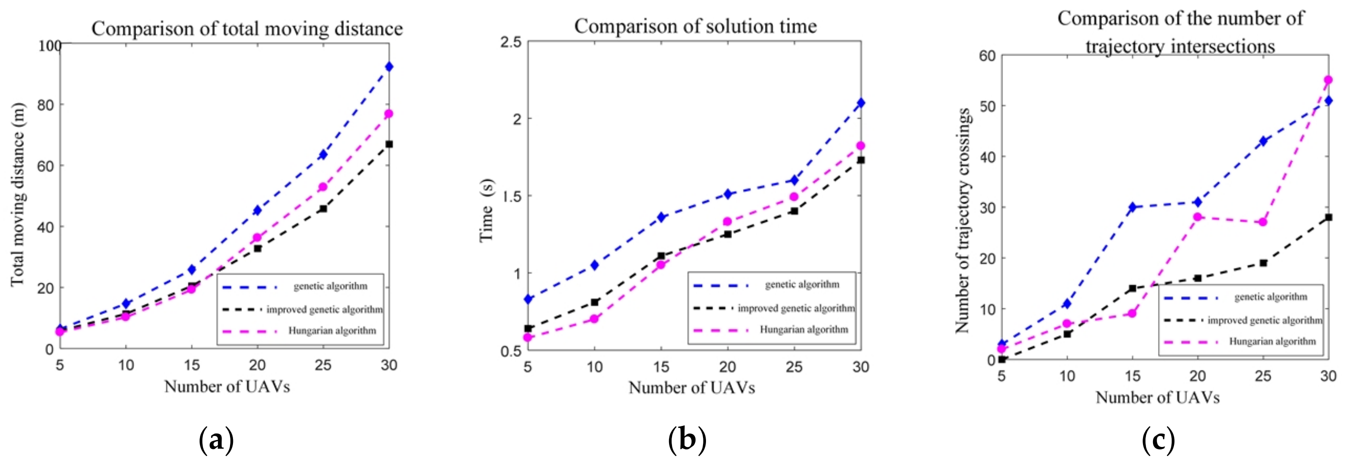

Figure 7. (**a**) Description of comparison of total moving distance; (**b**) description of comparison of solution times; (**c**) description of comparison of the number of trajectory intersections.

The target-matching results of the Hungarian algorithm and the improved genetic algorithm are shown in Figure 8.

The two comparison experiments above reveal that with a small number of UAVs in the formation transformation, trajectories rarely intersect. However, as the number of UAVs increases, the Hungarian algorithm, which does not consider trajectory crossover, generates more crossovers in its optimal solution compared to the improved genetic algorithm.

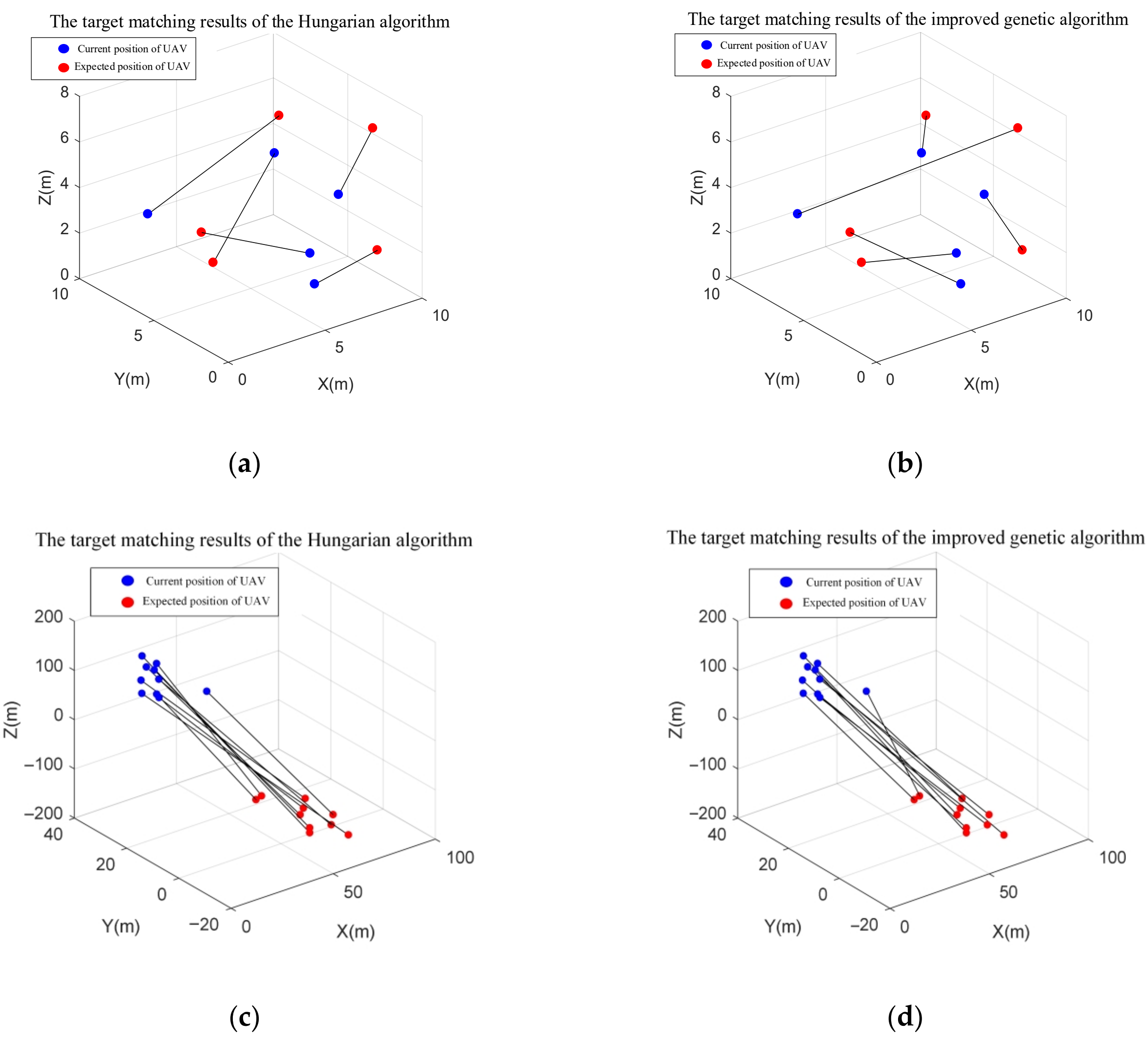

Figure 8. (**a**) Description of Hungarian algorithm target-matching results for 5 UAVs; (**b**) Description of improved genetic algorithm target-matching results for 5 UAVs; (**c**) Description of Hungarian algorithm target-matching results for 10 UAVs; (**d**) Description of improved genetic algorithm target-matching results for 10 UAVs.

4.2. Simulation Verification of Formation Transformation Collision Avoidance Constraints

This subsection examines the impact of collision avoidance constraints in the trajectory-planning algorithm to demonstrate that collision-free arrival at the target position is possible during the formation transformation. We set the initial formation of four UAVs as a rectangle to necessitate trajectory crossover during the transformation process. The crossover transformation is then applied to the diagonal vertices within the 2D *xoy* plane. The coordinates of UAVs 1–4 are set as (0, 0), (50, 50), (50, 0), and (0, 50), respectively, and the simulation results are presented in Figures 9–12.

It can be seen from the figure that in the diagonal vertex exchange process, collision avoidance constraint plays a role in trajectory planning; when UAV 1 and UAV 2 are exchanged in position, the UAV 1 trajectory is shifted, avoiding the situation of head-on collision with UAV 2.

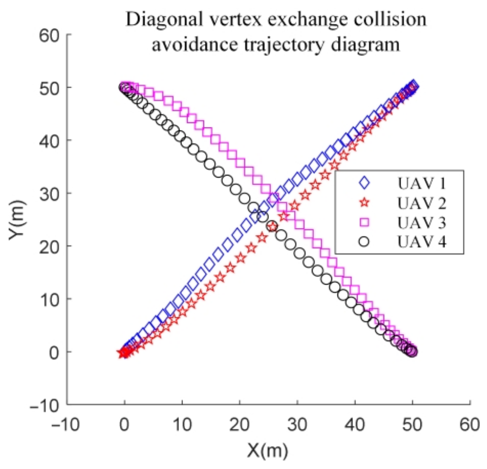

Figure 9. Diagonal vertex exchange collision avoidance trajectory diagram.

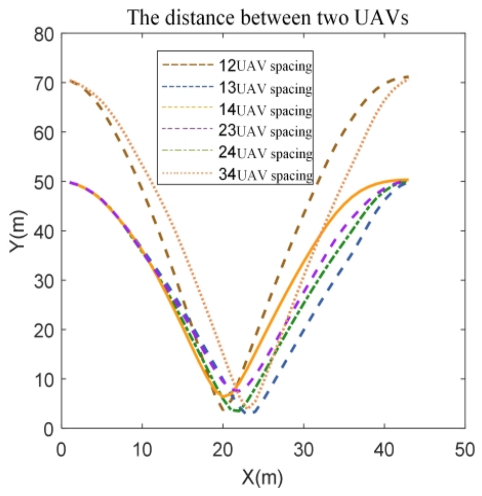

Figure 10. Distance between two UAVs.

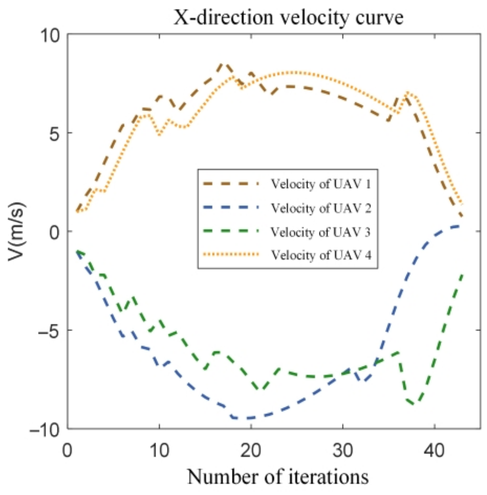

Figure 11. The x-direction velocity curves for 4 UAVs.

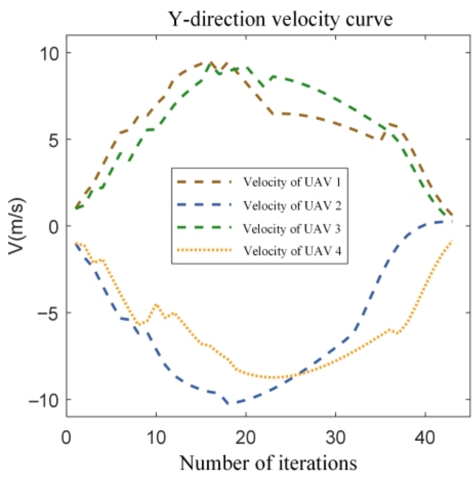

Figure 12. The y-direction velocity curves for 4 UAVs.

4.3. Simulation and Verification of Trajectory-Planning Algorithm

We independently simulated transformation scenarios for distance-sparse and distance-dense formations to gauge the efficacy of the trajectory planner. Firstly, we defined the following simulation conditions for the distance sparse formation:

The current formation of 13 UAVs is set as the letter "N", the desired formation is set as the letter "A", and the compactness factor of the formation is set as k_{tight}. The larger the k_{tight} is, the thinner the formation is. For this simulation, k_{tight} = 10; the specific coordinates of formation "N" and formation "A" are presented in Table 1.

Table 1. Initial and desired formation coordinate settings for 13 UAVs.

UAV Number	Position Coordinates of the Actual Formation N (m)	Position Coordinates of the Desired Formation A (m)
1	k_{tight} (10, 10, 10)	k_{tight} (−50, −50, 20)
2	k_{tight} (10, 10, 12)	k_{tight} (−49, −50, 22)
3	k_{tight} (10, 10, 14)	k_{tight} (−48, −50, 24)
4	k_{tight} (10, 10, 16)	k_{tight} (−47, −50, 26)
5	k_{tight} (10, 10, 18)	k_{tight} (−46, −50, 28)
6	k_{tight} (12, 10, 16)	k_{tight} (−45, −50, 30)
7	k_{tight} (14, 10, 14)	k_{tight} (−44, −50, 28)
8	k_{tight} (16, 10, 12)	k_{tight} (−43, −50, 26)
9	k_{tight} (18, 10, 10)	k_{tight} (−42, −50, 24)
10	k_{tight} (18, 10, 12)	k_{tight} (−41, −50, 22)
11	k_{tight} (18, 10, 14)	k_{tight} (−40, −50, 20)
12	k_{tight} (18, 10, 16)	k_{tight} (−46, −50, 24)
13	k_{tight} (18, 10, 18)	k_{tight} (−44, −50, 24)

Figures 13–18 depict the initial state diagrams of formation transformation and the corresponding target-matching results.

In the simulation of compact formation transformation, the formation compactness factor k_{tight} was set to 2, and the rest of the conditions were set to be consistent with the sparse formation transformation; the simulation results are presented in Figures 19–22.

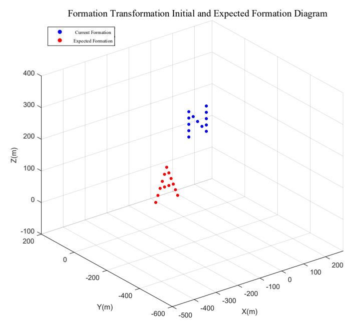

Figure 13. Initial state of 13-UAV sparse formation transformation.

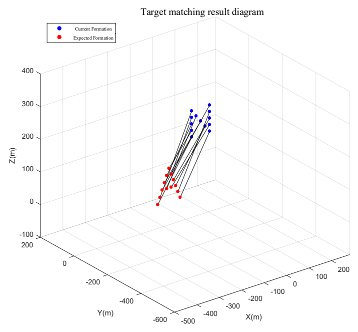

Figure 14. Improved genetic algorithm target-matching results for 13 UAVs.

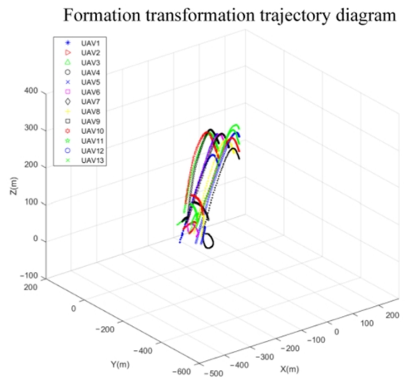

Figure 15. Sparse formation transformation trajectory-planning diagram for 13 UAVs.

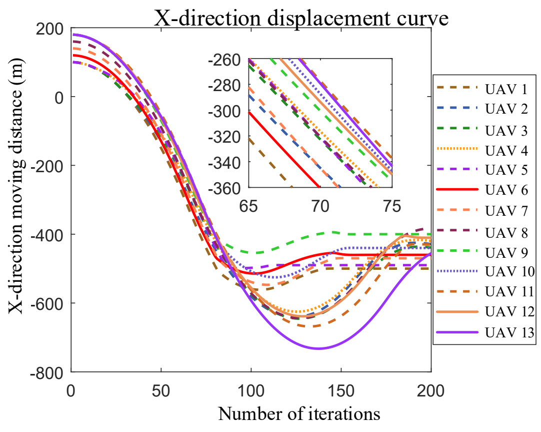

Figure 16. Sparse formation transformation x-direction displacement curve for 13 UAVs.

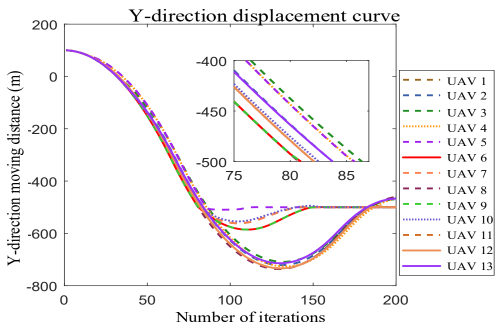

Figure 17. Sparse formation transformation *y*-direction displacement curve for 13 UAVs.

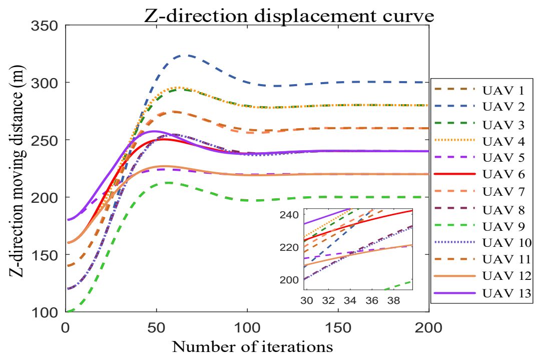

Figure 18. Sparse formation transformation *z*-directional displacement curve for 13 UAVs.

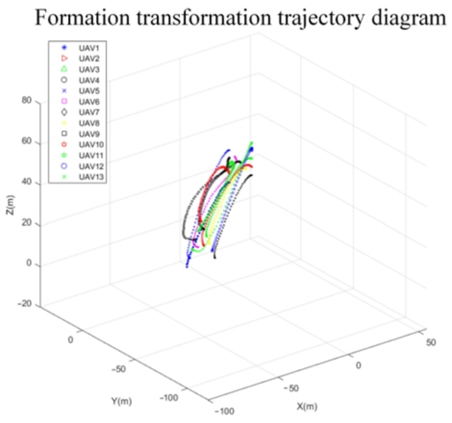

Figure 19. Compact formation transformation trajectory diagram for 13 UAVs.

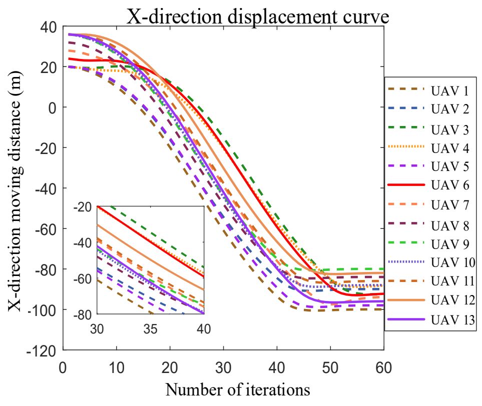

Figure 20. Compact formation transformation *x*-direction displacement curve for 13 UAVs.

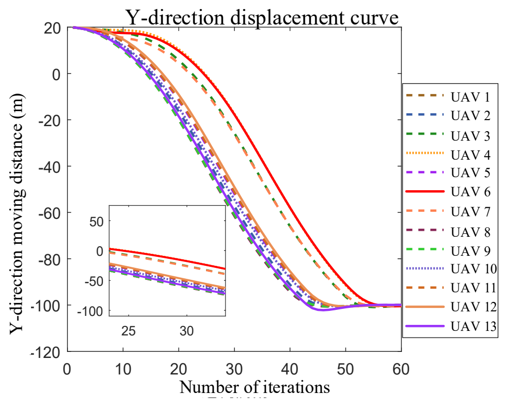

Figure 21. Compact formation transformation y-direction displacement curve for 13 UAVs.

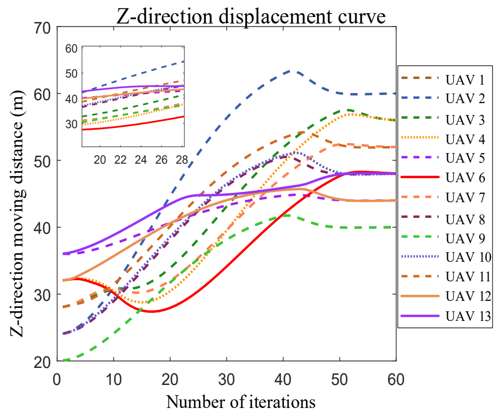

Figure 22. Compact formation transformation z-directional displacement curve for 13 UAVs.

From the simulation results, Figure 14 shows that the improved genetic algorithm successfully performs the UAV formation position assignment. Figure 15 shows the path diagram of 13 UAVs planned from their own starting point to the end point, from which it can be seen that the trajectories of individual UAVs do not cross each other. Figures 16–18 show the displacement curves of each UAV in the x, y, z directions with time during the formation transformation. From the final end coordinates, the x-axis coordinates of the 13 drones are concentrated in the range of -500 m to -400 m, while the y-axis coordinates are concentrated around -500 m. In the z-axis direction, two drones converge at a height of 200 m, two at 220 m, four at 240 m, two at 280 m, and one at 300 m. The actual simulation results are consistent with the expected results set in the experiment. Therefore, the trajectories planned by the UAVs can ensure the smooth transformation of each aircraft from the current formation to the desired formation.

When the formation spacing was decreased, the single-axis distance between UAVs ranged between 2 m and 4 m. Simulation results indicate that collision-free trajectory planning can still be achieved with smooth convergence from the current formation "N" to the desired formation "A", as shown in Figures 20–22.

4.4. Semi-Physical Simulation Results and Analysis

The formation transformation algorithm was previously verified by Matlab simulation. In order to further verify the effectiveness of the proposed algorithm in real engineering applications, semi-physical simulations were performed using Airsim Unreal Engine in conjunction with the Ardupilot flight control and distributed networking data transmission.

The goal of the simulation was to change the 13 UAVs from the initial "M" formation to the "A" formation, and the initial and final formations of UAV 1 to UAV 13 are shown in Figure 23.

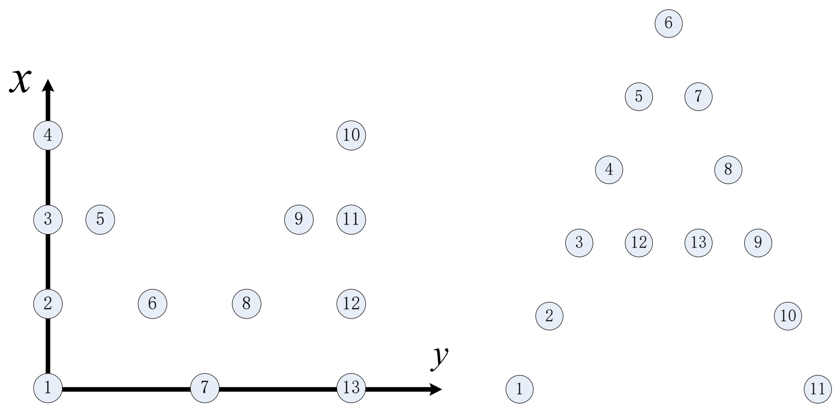

Figure 23. Formation transition diagram.

The initial coordinates of UAVs 1–13 are (0, 0, 10), (6, 0, 10), (12, 0, 10), (18, 0, 10), (12, 3, 10), (6, 6, 10), (0, 9, 10), (6, 12, 10), (12, 15, 10), (18, 18, 10), (12, 18, 10), (6, 8, 10), and (0, 18, 10). The coordinates of the final formation are (60, 60, 50), (64, 63, 50), (68, 66, 50), (72, 69, 50), (76, 72, 50), (80, 75, 50), (76, 78, 50), (72, 81, 50), (68, 84, 50), (64, 87, 50), (60, 90, 50), (68, 69, 50), and (68, 72, 50). The simulation results are plotted in Figures 24–27.

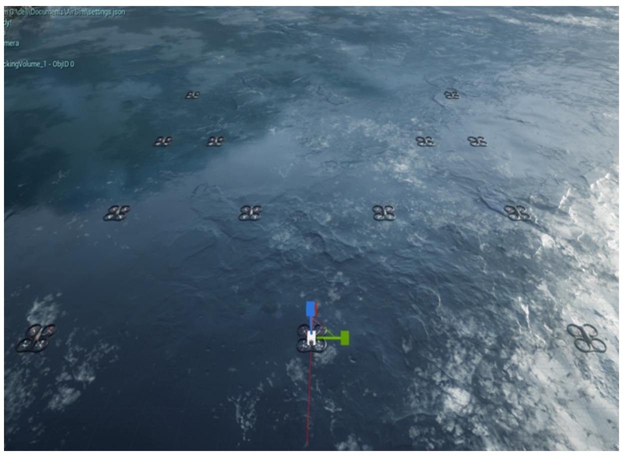

Figure 24. Initial formation.

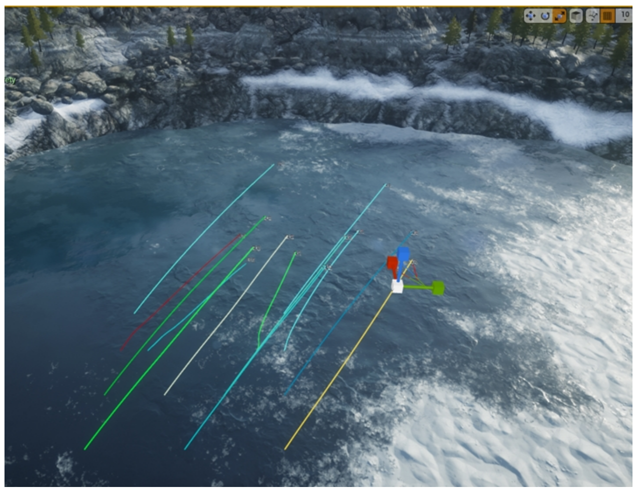

Figure 25. Initial trajectory.

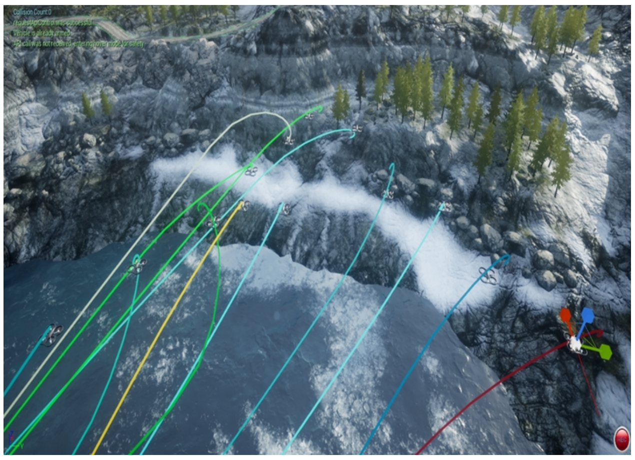

Figure 26. Final formation.

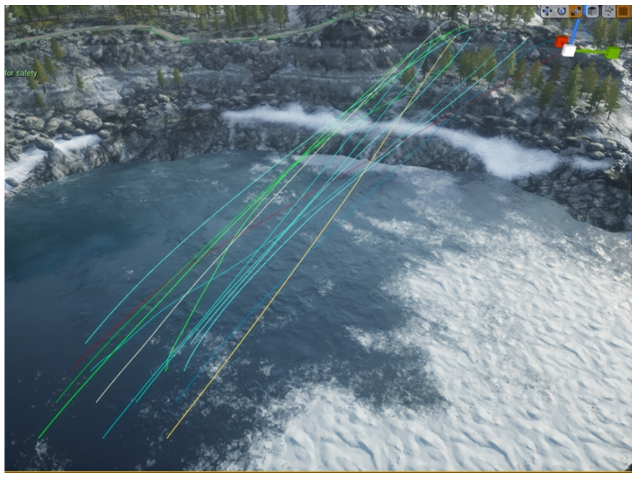

Figure 27. Overall trajectory diagram.

From the simulation diagram, it can be seen that each UAV flies according to the target position point assigned by the improved genetic algorithm, there is no cross-collision in the trajectory of the formation transformation, and there is an error of about 2 m in the composition of "A", but the transformed formation is basically maintained.

5. Conclusions

When designing a formation transformation algorithm, the possibility of failure in planning trajectories using distributed model predictive control algorithms for overly compact formations should be acknowledged. Moreover, further discussions should be conducted regarding collision avoidance constraints to address the formation transformation problem under more compact conditions.

This paper primarily addresses the formation transformation problem, which involved decomposing it into target-matching and trajectory-planning problems. The target-matching algorithm adopts the improved genetic algorithm, designs the adaptive cross-mutation probability, and realizes the population reproduction without a repeated string using a hash algorithm, and the improved genetic algorithm is combined with the distributed model predictive control strategy to achieve the formation transformation. The feasibility of the engineering of the method was verified by simulation as well as semi-physical experiments.

Author Contributions: Conceptualization, G.C.; methodology, C.Z.; software, S.Z., X.W. and H.G.; validation, G.C., C.Z. and S.Z.; writing—review, G.C., C.Z., X.W., S.Z. and H.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Cai, Z.; Wang, L.; Zhao, J.; Wu, K.; Wang, Y. Virtual target guidance-based predictive control for formation control of multiple UAVs. *Chin. J. Aeronaut.* **2020**, *33*, 1037–1056. [CrossRef]
2. Luis, C.E.; Schoellig, A.P. Trajectory generation for multiagent point-to-point transitions via distributed model predictive control. *IEEE Robot. Autom. Lett.* **2019**, *4*, 375–382. [CrossRef]
3. Rawlings, J.B.; Mayne, D.Q.; Diehl, M. *Model Predictive Control: Theory, Computation, and Design*; Nob Hill Publishing: Madison, WI, USA, 2017.
4. Liu, C.-H. Centralized control-based formation of multiple unmanned aerial vehicles. *J. Huazhong Univ. Sci. Technol. Nat. Sci. Ed.* **2015**, *43*, 481–485.
5. Zhou, L.; Li, S. Distributed model predictive control for multi-agent flocking via neighbor screening optimization. *Int. J. Robust Nonlinear Control* **2017**, *27*, 1690–1705. [CrossRef]
6. Shao, Z.; Yan, F.; Zhou, Z.; Zhu, X. Path planning for multi-UAV formation rendezvous based on distributed cooperative particle swarm optimization. *Appl. Sci.* **2019**, *9*, 2621. [CrossRef]
7. Cai, Z.; Zhou, H.; Zhao, J.; Wu, K.; Wang, Y. Formation control of multiple unmanned aerial vehicles by event-triggered distributed model predictive control. *IEEE Access* **2018**, *6*, 55614–55627. [CrossRef]
8. Christofides, P.D.; Scattolini, R.; de la Peña, D.M.; Liu, J. Distributed model predictive control: A tutorial review and future research directions. *Comput. Chem. Eng.* **2013**, *51*, 21–41. [CrossRef]
9. Wang, T.-M.; Zhang, Y.-C.; Liang, J.-H.; Chen, Y.; Wang, C.-L. Multi-UAV collaborative system with a feature fast matching algorithm. *Front. Inf. Technol. Electron. Eng.* **2020**, *21*, 1695–1712. [CrossRef]
10. Chen, R.; Yang, B.; Zhang, W. Distributed and collaborative localization for swarming UAVs. *IEEE Internet Things J.* **2020**, *8*, 5062–5074. [CrossRef]
11. Brandão, A.S.; Sarcinelli-Filho, M. On the Guidance of Multiple UAV using a Centralize Formation Control Scheme and Delaunay Triangulation. *J. Intell. Robot. Syst.* **2016**, *84*, 397–413. [CrossRef]
12. Xiao, H.; Chen, C.L.P. Incremental updating multirobot formation using nonlinear model predictive control method with general projection neural network. *IEEE Trans. Ind. Electron.* **2018**, *66*, 4502–4512. [CrossRef]
13. Zhang, J.; Meng, F.; Zhou, Y.; Lu, G.; Zhong, Y. Decentralized Formation Control of Multi-UAV Systems Under wind Disturbances. In Proceedings of the 2015 34th Chinese Control Conference, Hangzhou, China, 28–30 July 2015; pp. 7392–7397.
14. Azam, A.; Dey, S.; Mittelmann, H.D.; Ragi, S. Decentralized UAV Swarm Control for Multitarget Tracking using Approximate Dynamic Programming. In Proceedings of the 2021 IEEE World AI IoT Congress (AIIoT), Virtual, 10–13 May 2021; pp. 0457–0461. [CrossRef]
15. Ren, W. Consensus Based Formation Control Strategies for Multi-Vehicle Systems. In Proceedings of the American Control Conference, Minneapolis, MN, USA, 14–16 June 2006; pp. 6–16.
16. Ren, W.; Sorensen, N. Distributed Coordination Architecture for Multi-Robot Formation Control. *Robot. Auton. Syst.* **2008**, *56*, 324–333. [CrossRef]
17. Ren, W.; Peeld, R.W. *Distributed Consistency in Collaborative Control of Multimarine Bodies*; Wu, X.F., Translator; Electronic Industry Press: Beijing, China, 2014; pp. 5–7.
18. Wang, P.; Liu, C. Distributed formation control of second-order nonlinear multi-intelligent body systems. *Unmanned Syst. Technol.* **2021**, *4*, 26–31. [CrossRef]
19. Gou, Z.; Wu, Y.; Deng, J.N. Research on a full process flight path planning method for UAV formation based on swarm intelligence-consistency theory. *Control. Decis. Mak.* **2023**, *38*, 1464–1472. [CrossRef]
20. Kuriki, Y.; Namerikawa, T. Formation Control with Collision Avoidance for a Multi-UAV System using Decentralized MPC and Consensus-Based Control. In Proceedings of the European Control Conference, Linz, Austria, 15–17 July 2015; pp. 3079–3084.
21. Najm, A.A.; Ibraheem, I.K.; Azar, A.T.; Humaidi, A.J. Genetic Optimization-Based Consensus Control of Multi-Agent 6-DoF UAV System. *Sensors* **2020**, *20*, 3576. [CrossRef] [PubMed]