Flight Test Mission Reassignment for Civil Aircraft Based on Flexible Job Shop Problem

Yi Mi, Aijun Li* and Hao Song

Abstract—**Civil aircraft represent intricate technical systems, and their flight test is a system engineering process. This process is essential for verifying their compliance with airworthiness regulations and operational requirements. In this process, planning flight test missions is a type of combinatorial optimization problem. The solution gives an optimal allocation plan adhering to operational constraints. Assigning flight test missions for civil aircraft involves numerous uncertainties and very often, requires unforeseen adjustments. The original missions need to be reassigned due to these adjustments in a scientific way. In this paper, considering the scenarios that emerged during real civil aircraft flight tests, a multi-objective flight test mission planning model is established based on the Flexible Job Shop Problem (FJSP) model. The model is specifically tailored to the flight text mission reassignment. Examples of flight test scenarios that require mission reassignments are addressed using the FJSP model proposed in this paper and optimized using the Particle Swarm Optimization algorithm. Two rounds of initial population updates were conducted to obtain overall extreme values. The results verify the feasibility of obtaining a viable solution for the flight test mission assignment problem under complex constraints derived from real engineering requirements. These solutions can be used to address similar types of engineering problems.**

Index Terms—**Flight Test Mission, Mission Assignment, FJSP, PSO, Neighborhood Operations, Example Simulation**

I. Introduction

Flight testing plays a pivotal role in scientific research and airworthiness evaluation. It involves a systematic compilation of flight test missions aimed at achieving testing objectives [1]. This process is crucial, especially in the development of new civil airplanes for advanced materials and technologies. Given its strategic significance, meticulous planning is imperative.

Efficient flight test mission planning necessitates thorough sorting and analysis of test characteristics. Each flight test mission is meticulously broken down into its smallest units, utilizing basic logical relationships among civil aircraft flight test missions to devise optimized combinations. Subsequently, a comprehensive optimization method is designed. This method entails formulating aircraft model flight test plans and test mission lists.

Manuscript received May 29, 2023; the revised May 15, 2024.

Yi Mi is a PhD candidate of the School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi Province, P.R. China. (e-mail: miyee827@163.com).

Aijun Li is a professor of the School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi Province, P.R. China. (corresponding author to e-mail: liaijun@nwpu.edu.cn).

Hao Song is a senior engineer of the Flight Test Center, Commercial Aircraft Corporation of China, Ltd. (COMAC), Shanghai, P.R. China. (e-mail: ftcsh23@ 163.com).

A flight test subject comprises a minimal set of missions that independently describe specific flight performance and handling qualities of the aircraft, verify the functionality of aircraft system design indicators, and ensure compliance with relevant airworthiness requirements.

The guiding principle of flight test planning is to align all flight test missions with aircraft development goals. This involves comprehensive consideration of various constraints based on available information and data, effective coordination of flight test resources, mitigation of potential uncertainties, and parallel assignment of missions to multiple test aircraft.

For a specific model of civil aircraft, obtaining the Type Certificate (TC) and facilitating commercial operation by airlines necessitates completion of thousands of research studies, qualification review subjects, and a multitude of flight test missions. These missions encompass operational flight tests, functional reliability flight tests, and delivery flight tests, starting from the prototyping phase. Flight test mission planning is thus geared towards judiciously arranging these test subjects for flight testing across several prototype aircraft.

In summary, flight test planning inherently falls within the realm of combinatorial optimization problems, known for their inherent complexity. Most scheduling problems, including flight test planning, are widely regarded as NP-hard problems.

II. The FJSP Model

2.1 Flexible Job-shop Scheduling Problem

The purpose of production scheduling is to systematically arrange the processing time of workpieces to optimize one or several scheduling objectives [2]. Traditional job shop scheduling problems (JSP) typically address scenarios where each process is executed on a predetermined machine, with a fixed processing plan established prior to job scheduling. These problems are designed to tackle scheduling challenges where each workpiece or process necessitates a specific processing machine, lacking flexibility.

However, in real production environments, there often exist multiple machines capable of handling a particular mission, leading to what is known as the flexible job shop scheduling problem. This problem extends the traditional JSP by incorporating the machine assignment problem, which involves assigning each operation to several optional machines, thereby expanding the feasible solution space of the problem. Consequently, the flexible job shop scheduling problem presents a more intricate form of NP-hard problem compared to its classical counterpart. It is also referred to as the job shop scheduling problem with flexible processing routes or machine flexibility [3, 4].

Formally, the flexible job shop scheduling problem (FJSP) can be defined as follows: a set of jobs, each comprising multiple operations, must be processed across a set of machines where each operation can be performed. The processing time for each operation on different machines may vary, and there are specific constraints that must be adhered to. The optimal scheduling plan for the job shop, which satisfies these constraints, involves allocating jobs and operations to machines based on a designated objective function.

The actual FJSP has the following characteristics.

a) Multi-objective;
b) Multi-binding;
c) Discrete;
d) Uncertainty;
e) Calculate complexity.

In a typical FJSP, the processing time for different operations on various machines may vary. However, during actual flight tests, there is usually no significant disparity in the implementation time for different test subjects across different types of aircraft. As aircraft configurations and flight test technologies mature, the flight test intensity that test aircraft can endure may fluctuate, often escalating gradually throughout different phases of civil aircraft flight testing.

The FJSP can be categorized into two types: the Total Flexible Job Shop Scheduling Problem (T-FJSSP) and the Partial Flexible Job Shop Scheduling Problem (P-FJSSP) [3]. If each operation can be processed by any machine, then it is a T-FJSSP. Conversely, if at least one operation can only be processed by specific machines, then it becomes a P-FJSSP. Thus, T-FJSSP can be viewed as a special case of P-FJSSP [5, 6]. The dynamic scheduling of flight test missions aligns with the P-FJSSP problem type.

For single-objective FJSP, existing literature [7] demonstrates that the relocation of critical processes can notably reduce completion time, and it has devised two neighborhood structures based on such critical process relocation, alongside an effective taboo search algorithm.

Moreover, literature [8] devised a genetic algorithm (GA) employing a blend of two machine allocation rules and three operation sequencing rules to generate high-performance initial populations. Additionally, literature [9] proposed a dual-population distribution estimation algorithm, utilizing two sub-populations to independently adjust machine allocation and operation sequencing sequences, and leveraging the dominant population to construct a probability model for generating new individuals. Meanwhile, literature [10] introduced a bounded-depth diversification search algorithm, relying on a sorting heuristic strategy and employing block symbols to formulate neighborhood structures.

In addressing the FJSP, literature [11] explored hybrid algorithms combining Particle Swarm Optimization (PSO) and Simulated Annealing (SA) to tackle machine allocation and process sequencing challenges. Furthermore, literature [12] proposed a Genetic Algorithm (GA) incorporating bottleneck drifting and local search techniques, while literature [13] introduced an effective hybrid PSO and Tabu Search (TS) method.

In the realm of static scheduling problems within FJSP, it is assumed that all machines and jobs are initially in an accessible state. After the initial scheduling, the processing sequence for each job is established and remains unchanged during subsequent processing. However, real-world workshops inevitably face uncertain events such as machine failures or delayed material arrivals, necessitating dynamic scheduling to adapt to such contingencies.

Additional literature [14] delves into the multi-objective allocation problem subject to multiple constraints. Meanwhile, literature [15] introduces the Multi-Objective Criteria Location Cuckoo Search (MOQOCS) algorithm employing a quasi-position-based learning mechanism. Literature [16] utilizes a Tabu search mechanism to solve combinatorial optimization problems. Lastly, literature [17] offers a comprehensive dynamic scheduling model for a multi-objective flexible job shop, considering objectives like completion time, machine load, weighted tardiness, and energy consumption, and utilizes the Pareto method to address the problem.

2.2 Variables and Mathematical Symbols

n : Total number of flight test mission modules;

m : The total number of test aircraft.

M : The set of total test aircraft, $\{M_1, M_2, \cdots, M_m\}$;

J : The set of all flight test modules, $\{J_1, J_2, \cdots, J_n\}$;

O : The set of all flight test subjects, $\{O_{11}, O_{12}, \cdots, O_{ne_n}\}$;

M_k : The k^{th} flight test aircraft, $k = 1, 2, \cdots, m$;

n_i : Number of subjects in flight test mission module i ;

O_{ij} : Subject j^{th} of mission module i ;

M_{ij} : The set of test aircraft available for the subject O_{ij} , $M_{ij} \subset M$;

$P_{ij,k}$: The implementation time of the subject O_{ij} on test aircraft k ;

$t_{ij,k}$: The start time of the subject O_{ij} on test aircraft k (assuming that the start time of the subject is a discrete value);

C_{ij} : Completion Time of Subject O_{ij} ;

e_i : The total number of subjects corresponding to the module J_i ;

$Z \triangleq$: Total number of subjects included in all modules, $Z = \sum_{i=1}^{n} e_i$;

$x_{ij,k,t}$ indicates the judgment criteria executed during the test flight subject on the test aircraft at the time t .

$$x_{ij,k,t} = \begin{cases} 1, \text{ if } O_{ij} \text{ operates on FTA } k \text{ at the moment } t \\ 0, \text{ else} \end{cases}.$$

III. Description of Planning and Optimization Scenarios for Civil Aviation Flight Test Missions.

3.1 Flight Test Mission Planning Scenarios

According to the "New Aircraft Access Management Requirements for Models," flight test planning should commence prior to the preliminary design phase. During the initial planning stage, conditions may be idealized, resulting in static planning (job scheduling). At this stage, the initial state of all test aircraft and test missions is established, and the execution order of each mission workpiece is determined, yielding a set of flight test schemes. If there are no subsequent changes, this is referred to as completing static job scheduling.

As development progresses on new aircraft types, our understanding of the associated technologies, systems, and materials also matures. Consequently, our comprehension of flight test missions evolves accordingly. Throughout this process, flight test missions may encounter modifications across various scenarios until the actual implementation of flight tests for the new aircraft.

During the actual flight test planning process, several inevitable uncertainties arise, including adjustments to the number of test aircraft, discrepancies in aircraft configuration and status compared to the plan, aircraft malfunctions, and delays in modifications. When such events occur frequently, the predetermined planning scheme becomes impractical, necessitating the adoption of an appropriate method for rescheduling. This scenario is commonly referred to as the dynamic programming (scheduling) problem.

3.2 Flight Test Mission Dynamic Planning Scenario

In the dynamic programming problem of flight test missions, the execution of missions (including any changes in planning conditions before the first flight) is viewed as a dynamic process. Unlike traditional static scheduling, dynamic scheduling problems require not only considering the initial processing state of the workshop but also frequently incorporating dynamic factors such as inserting emergency missions. Hence, the performance indicators of dynamic scheduling problems surpass those of static scheduling. Alongside traditional scheduling performance indicators such as maximum completion time and average flow-through time, it is also crucial to assess the deviation degree of the new scheduling from the initial original scheduling. Consequently, dynamic scheduling typically involves multiple performance indicators, and the optimization goals may be contradictory or even incompatible with each other. In the context of civil aircraft test flights, the primary planning indicators predominantly focus on minimizing the maximum completion time and minimizing the initial allocation deviation, thereby framing a multi-objective flight test mission planning problem [3, 18].

Potential uncertainties that may arise during flight testing of civil aircraft are as follows:

a) Inadequate aircraft configuration or unsuitable testing conditions may prevent the conduct of subject flight testing;
b) Malfunctions of the test aircraft can affect the implementation of flight testing;
c) New test subjects may be introduced;
d) The implementation sequence of flight test subjects may be subjectively adjusted for certain reasons;
e) Flight test subjects may need to be repeated if not completed.

The preceding uncertain events have a high probability of occurring in real civil aircraft flight test missions, and therefore it is necessary to introduce dynamic reallocation (scheduling) methods to optimize and update planning schemes. The scenario designed in this paper considers that each test aircraft has a factory interval and cannot be put into flight test operations simultaneously, so a planning scheme for reallocation of the initial allocation is considered.

IV. Flight Test Mission Planning Problem Model

4.1 Logical Constraints in Flight Test Missions

The FJSP model for the flight test mission planning problem is delineated as follows: The primary objective is to successfully complete a sequence of flight test modules utilizing a fleet of test aircraft. Each module comprises multiple flight test subjects that must be conducted in a specific order (referred to as subject logic), and each subject is exclusive to a single test aircraft. At any given time, each test aircraft is capable of executing only one flight test subject. The key decision revolves around the optimal sorting of subjects for the group of test aircraft to streamline the mission assignment plan.

It's noteworthy that the assertion "Each subject can only be executed on one test aircraft" implies that each subject can be designated for missions on a single test aircraft or potentially reassigned to another test aircraft. However, the responsibility for executing the mission remains solely with one test aircraft.

The aforementioned considerations can be encapsulated in the model description as follows:

If there are n flight test mission modules and M flight test aircraft available, each module $i\left(i \in \{1,2,\cdots,N\}\right)$ contains $n_i\left(n_i \geq 1\right)$ subjects that must be implemented following a specified logic.

The subject $j\left(j \in \{1,2,\cdots,n_i\}\right)$ of the module i is represented by O_{ij}, and the set of test aircraft able to perform the subject j of the module i is represented by M_{ij}, and $M_{ij} \subseteq M$. Each subject O_{ij} can be implemented on any of the $\left|M_{ij}\right|$ test aircraft with implementation capabilities on test aircraft, $M_k \in M_{ij}$, $k \in \{1,2,\cdots,m\}$, and test aircraft M_k can perform multiple subjects of different flight test mission modules.

The mission execution time of O_{ij} in test aircraft M_k is represented by $t_{ij,k}$. During the implementation process, each subject O_{ij} is not allowed to be interrupted, and the same test aircraft can perform only one test flight at a time.

If the completion time of the planned mission is not consistent with the estimated completion time, the project is weighed in terms of cost based on the difference in magnitude and direction (positive for early completion and negative for delays), and the optimization target set as minimizing the cost of early/late completion.

The goal of using the FJSP model for flight test mission planning and optimization is to select an aircraft M_k for each subject O_{ij}, and to arrange the sequence of various modules or subjects assigned to the test aircraft M_k to determine the start time of their implementation and obtain the best solution that balances all decision-making indicators.

Assumptions are given:

a) All test aircraft are available from the initial moment of the first flight, $t=0$;

b) There are logical constraints between subjects regarding the order of implementation, with one subject having to be carried out only after the completion of a preset subject. Suppose that the subject b in module a, O_{ab}, is the prerequisite of the subject h in module g, O_{gh}. Thus, assume that the completion time of O_{ab} is C_{ab}, that of the O_{gh} is C_{gh}.

$$C_{gh}-C_{ab}\geq P_{ab,k}x_{ab,k} \tag{1}$$

c) Given any time t, each subject can only be carried out by one test aircraft simultaneously (test aircraft constraint I).

$$\sum_{k=1}^{m}x_{ij,k,t}=1 \tag{2}$$

d) Given any time t,simultaneously, a test aircraft can only perform one subject(test aircraft constraint II);

$$\sum_{i=1}^{n}\sum_{j=1}^{e_i}x_{ij,k,t}\leq 1 \tag{3}$$

e) The ongoing subject is not allowed to be interrupted (continuity constraint).

$$x_{ij,k,t_{ij,k}}+x_{ij,k,(t+1)_{ij,k}}\leq 1 \tag{4}$$

or

$$C_{ij,k}-t_{ij,k}=P_{ij,k} \tag{5}$$

f) As there is a logical relationship between the subjects, the flight test mission module follows the logical relationship of the subjects.

g) Some subjects can only be implemented within a limited time frame (window period restrictions).

4.2 Dynamic Programming for Flight Test Missions

Dynamic job scheduling entails the allocation and adjustment of flight test missions amidst changing conditions, including shifts in the implementation time of flight test subjects and equipment failures. In response to unexpected circumstances such as configuration changes, equipment malfunctions, or shortages of spare parts from suppliers, timely and appropriate redistribution and adjustments of flight test missions can be executed based on the mission capabilities of the test aircraft. This ensures the maintenance of an optimal or sub-optimal allocation plan throughout the execution of flight test missions.

Dynamic programming essentially involves reassigning the globally optimal allocation solution obtained under initial conditions. It primarily comprises event-driven rescheduling, periodic-driven rescheduling, and mixed rescheduling, which integrates both periodic and event-driven processes. Event-driven rescheduling entails immediate adjustment when an event occurs that alters the system's state, while periodic rescheduling occurs at regular production intervals. Mixed rescheduling incorporates predictive capabilities and the ability to address unexpected events [19,20]. Given the practical context of flight test projects, the problem addressed in this paper primarily focuses on event-driven rescheduling to tackle dynamic programming challenges.

4.3 Model Building

According to the problem description and assumptions of flight test mission planning, a objective function is formulated with logical relationship constraints, aircraft constraints, and continuity constraints.

The objective function is described as follows [18]:

a) The minimization of the maximum completion time:

$$\min f_1(x)=\min\{\max C_i\} \tag{6}$$

b) Minimum deviation in initial allocation:

$$\min f_2=\min\left\{\sum_{j\in N}p_j\left|\upsilon_j-\upsilon_j^*\right|+\sum_{k\in M}R_k\right\} \tag{7}$$

In the realm of dynamic scheduling, the deviation from the initial schedule serves as a metric to gauge the stability of the dynamic schedule. Considering the intricacies and complexities inherent in civil aircraft test flight projects, scheduling a specific subject to be conducted on a test aircraft entails a broad spectrum of factors encompassing modifications, airport facilities, equipment availability, and personnel allocation associated with that subject. When dynamic rescheduling occurs, the revised start time for the rescheduled subject triggers a redistribution of these aforementioned conditions and resources. Although these factors may not be explicitly accounted for at the planning stage, they are inherently linked to the flight test subject within the project and are adjusted in real-time as planning alterations unfold.

In dynamic scheduling, the objective is to retain the initial start time of the test subjects and the assigned test aircraft to the greatest extent possible, thereby minimizing the deviation from the initial scheduling plan. For each process J_i of the initially assigned program artifacts, the Subject Deviation Cost (SDC) is defined as [18]:

$$\mathrm{SDC}=\sum_{j\in N}p_j\left|\upsilon_j-\upsilon_j^*\right| \tag{8}$$

Where j represents each subject in the mission module that has not been implemented in the initial allocation plan; N represents the total number of subjects in the unimplemented module; p_j represents the penalty coefficient assigned to the subject j due to changes in implementation time, with $p_j\in[0,1]$; υ_j and υ_j^* respectively represent the start time of the initial allocation and re-allocation of the subject j.

When conducting flight tests on a test aircraft, there are additional attributes to consider, such as aircraft configuration and test modifications. When adjusting the allocation scheme for flight test subjects, it becomes crucial to consider the implications of these adjustments. This entails evaluating whether to maintain the initial assignment of the test aircraft for conducting the flight test missions. This paper introduces the concept of "test aircraft anchoring" and proposes two allocation strategies for redistributing flight test missions on the test aircraft based on the requirements of flight test engineering: "arbitrary allocation of test aircraft" and "test aircraft anchoring."

For each flight test subject O_{ij}, its completion time C_{ij} and the assigned test aircraft M_k:

The function $Z(t,T_a,M_k)$ determines whether the specified subject O_{ij} has already been completed on the designated test aircraft M_k before the reallocation of the start time.

Where T_a denotes the redistribution start time (anchoring time), which is a specific moment in a test flight cycle. Flight test subjects completed prior to this point are in a frozen state and will no longer be considered for subsequent optimization. t denotes the completion time of the test flight subject O_{ij}.

The function $R_k(T_s, T_a)$ denotes the reordering of subjects to be allocated after the anchoring time, but it is confined to the initial allocation of the test aircraft. Here, T_s represents the current attributes of the flight test subject, which encompasses the subject itself, the assigned test aircraft, and the start and end times.

The function $R_k^*(T_s, T_a)$ denotes the re-sorting of subjects that require reallocation after the anchoring time, permitting it to be executed on any available test aircraft.

The determination of aircraft anchoring is indicated as follows:

$$Z(t, T_a, M_k) = \begin{cases} 1, & \text{if } C_{ij} \le T_a \\ 0, & \text{else} \end{cases}$$

The test aircraft anchoring rescheduling is expressed as:

$$R_k(T_s, T_a) = \begin{cases} 1, \text{if } C_{ij} \le T_a \text{ and the initially assigned trial aircraft is } M_k \\ 0, \text{else} \end{cases} \quad (9)$$

In the rescheduling of flight test missions, an appropriate aircraft allocation strategy can be chosen based on the demands of flight test engineering, ensuring alignment with the engineering requirements for rescheduling flight test missions.

4.4 PSO algorithm

In 1995, Kennedy and Eberhart introduced the Particle Swarm Optimization (PSO) algorithm, drawing inspiration from the collective behavior of birds in search of optimal habitats. This algorithm has subsequently undergone adaptations for navigating problem-solving spaces [21, 22].The PSO algorithm, rooted in population evolution algorithms, begins by initializing a set of random solutions known as particles. Each particle possesses a fitness value determined by the optimization function and a velocity dictating its direction and distance of movement [23, 24].Over iterative epochs, particles dynamically adapt their positions by tracking both their own best solution (individual extremum) and the best solution found by the entire population in the current iteration (global extremum) [23, 24].

Notably, the PSO algorithm stands out due to its parallel processing capability and robustness, enabling it to effectively achieve global optima with high probability while demonstrating superior computational efficiency in contrast to traditional stochastic methods. Moreover, its ease of implementation, rapid convergence, and solid theoretical foundation make it suitable for both theoretical inquiries and practical engineering applications [21, 25]. Fundamental concepts within particle swarm optimization algorithms encompass particle, population, particle velocity, fitness value, individual extremum, and global extremum.

The typical description of the particle swarm algorithm is as follows:

a) Particle

Particles are the fundamental units in PSO, representing a feasible solution in the search space. Given a solution vector with d dimensions, at the iteration t of the algorithm, the i^{th} particle $x_i(t)$ is represented as $x_i(t) = [x_{i1}(t), x_{i2}(t), \cdots, x_{id}(t)]$. $x_{ik}(t)$ represents the position of the i^{th} particle in the k^{th} dimension of the search space.

b) Population

The population represents a set of n particles, indicating n candidate solutions. The population formed after t iterations is denoted as $pop(t) = [x_1(t), x_2(t), \cdots, x_i(t)] \cdots, x_n(t)$, where $x_i(t)$ is the i^{th} particle in the population.

c) Particle velocity

Particle velocity represents the position change of particles during a single iteration, expressed as $v_i(t) = [v_{i1}(t), v_{i2}(t), \cdots, v_{id}(t)]$, where $v_{ik}(t)$ is the velocity of the i^{th} particle in the k^{th} dimension.

d) Fitness value

The fitness value corresponds to the objective function. The variable representing the optimal solution, possessing the highest fitness value at the conclusion of the iteration, is regarded as the optimal solution for the ongoing phase of the optimization search.

e) Individual extremum

The i^{th} particle I has achieved the optimal fitness value from the start of the search to the current iteration and is denoted as $p_i = (p_{i1}, p_{i2}, \cdots, p_{id})$.

f) Global extremum

The optimal solution for the fitness value of the entire population from the start of the search to the current iteration is denoted as $g_i = (g_{i1}, g_{i2}, \cdots, g_{id})$.

In the PSO algorithm, particles are initially assigned velocities and positions. During each generation, the fitness function values of the particles are computed, enabling the identification of optimal values for both individual particles and the entire population. Subsequently, the velocity and position of each particle i in the k^{th} dimension are updated according to equations (10) and (11), respectively.

Assume a search in an N-dimensional space where the information of i is represented by two N-dimensional vectors describing the particle's position and velocity, as defined in the previous text.

The speed and position of the particle are updated as follows:

$$\begin{aligned} v_{id}^{k+1} = \omega \cdot v_{id}^{k} + c_1 \cdot rand_1 \cdot \left(pbest_{id}^{k} - x_{id}^{k} \right) \\ + c_2 \cdot rand_2 \cdot \left(pbest_{id}^{k} - x_{id}^{k} \right) \end{aligned} \quad (10)$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \quad (11)$$

In equation (10) and equation (11), v_{id}^{k} represents the velocity of the particle i in the d^{th} dimension in the k^{th} iteration, x_{id}^{k} represents the position of the d^{th} dimension in the k^{th} iteration, ω represents the inertia weight, c_1 and c_2 represent the learning factor, $rand_1$ and $rand_2$ represent random numbers between [0,1], $pbest_{id}^{k}$ represents the position of the individual extremal point of the particle i in the d^{th} dimension, and $gbest_{d}^{k}$ represents the position of the particle swarm at the global extremal point of the d^{th} dimension.

The article addresses the dynamic planning problem of test flight missions by employing a rescheduling method to optimize and update the initial allocation plan. Initially, the PSO algorithm is utilized for single-objective optimization, focusing on minimizing the maximum completion time, thereby yielding an optimal solution with a tendency towards stability. Subsequently, this solution serves as the initial population for the second round of PSO algorithm optimization, incorporating new constraints. Consequently, the multi-objective dynamic planning global optimal solution with minimum deviation from the initial solution is attained.

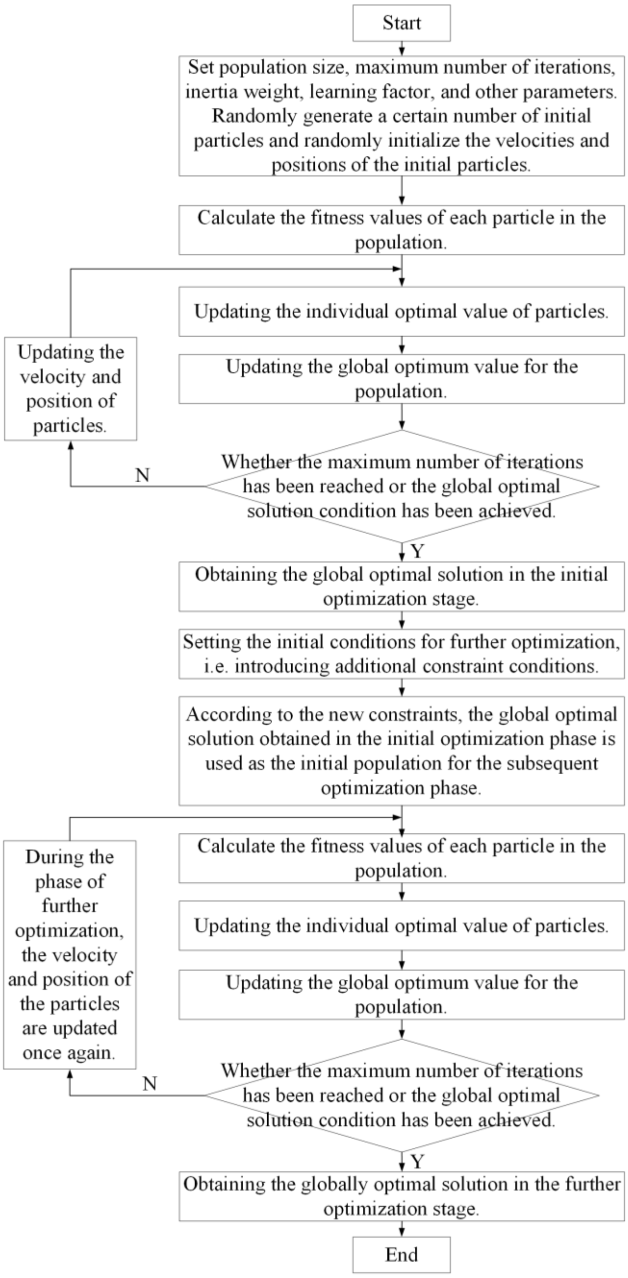

Fig 1. PSO flow chart for solving the dynamic planning problem of the flight test mission

The flowchart diagram depicting the PSO algorithm's application to solve the dynamic programming problem of the flight test mission is illustrated in Figure 1.

V. EXAMPLE DESIGN

5.1 Problem Description

During the implementation of flight tests for civil aircraft, it is essential to consider the logical relationships between various mission requirements. These relationships, derived or summarized from practical flight test engineering practice, adhere to the safety, accuracy, and efficiency of flight tests. Specifically, the logical relationship between flight test subjects discussed in this paper pertains to the mandatory execution sequence, known as pre-condition logic. This logic serves as a crucial constraint in determining the assignment order of flight test missions. Various flight test subjects exhibit interdependent relationships. For instance, before conducting flight tests for stall speed, maneuvering characteristics, lateral-directional stability and control, and handling qualities, it is imperative to first complete flight tests for airspeed system calibration and total air temperature calibration. Adhering to these dependencies ensures the formation of a planning scheme that aligns with the rules governing aircraft flight tests.

As an example, consider partial flight test subjects of a specific type of aircraft. Figure 2 illustrates the preconditions, where O21 represents the atmospheric data system calibration test item, serving as a prerequisite for numerous flight test items.

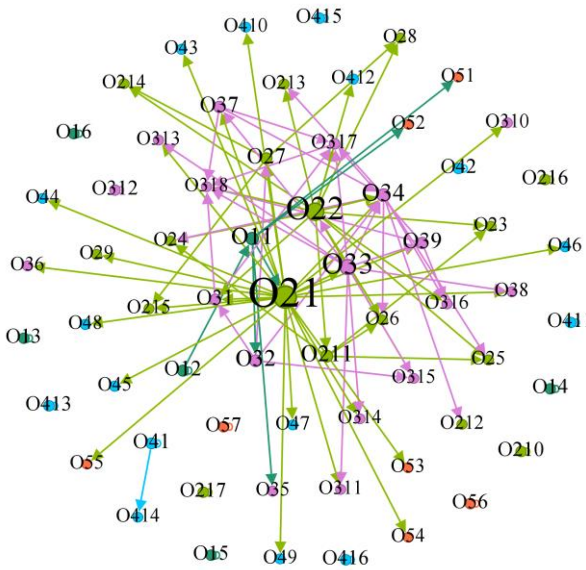

Fig 2. Example of the logical relationship between some test flight subjects of a certain type of aircraft

In the flight test mission planning scenario, five mission modules were chosen for a specific type of aircraft, encompassing a total of 64 test flight subjects distributed among five test aircraft for execution. For instance, flight test subjects O11 to O16 within the first flight module J1 denote the maiden flight and flight envelope expansion of the five test aircraft. The pre-relationships of each subject and the description of optional test aircraft are detailed in Table 1.

TABLE I
DYNAMIC PROGRAMMING PROBLEM DESCRIPTION FOR A 5*5 FLIGHT TEST MISSION (WHICH CAN BE CONDUCTED WITH THE SPECIFIC SUBJECT ON A DESIGNATED TEST AIRCRAFT).

Module	Subjects	Preceding Subject	Hours (h)	Test aircraft serial number				
				M1	M2	M3	M4	M5
J1	O11	O12	20	1	0	0	0	0
	O12	N/A	5	1	0	0	0	0
	O13	N/A	5	0	1	0	0	0
	O14	N/A	5	0	0	1	0	0
	O15	N/A	5	0	0	0	1	0
	O16	N/A	5	0	0	0	0	1
J2	O21	O11	20	1	1	1	1	1
	O22	O33	50	1	0	1	0	1
	O23	O22, O211	40	1	1	1	1	1
	O24	O22, O211, O34	50	1	0	0	0	0
	O25	O22, O211, O34	20	0	0	1	1	0
	O26	O22, O211, O34	10	1	1	1	1	1
	O27	O21, O32	45	0	0	0	0	1
	O28	O22, O27	15	0	0	1	1	1
	O29	O21	20	1	0	0	0	1
	O210	N/A	10	0	1	1	0	1
	O211	O22	20	1	1	1	0	0
	O212	O34	20	1	0	0	1	1
	O213	O22, O34	30	1	0	0	1	1
	O214	O22, O27	40	1	1	0	0	0
	O215	O21, O27	30	1	1	1	1	1
	O216	N/A	25	0	1	1	1	0
	O217	N/A	50	1	1	1	1	1
J3	O31	O21, O22, O32	60	1	1	1	1	1
	O32	O11	50	0	1	0	1	0
	O33	O21	60	0	0	1	0	0
	O34	O21, O22, O32, O33, O37	40	1	1	1	1	1
	O35	O11	30	1	1	0	1	0
	O36	O21	20	1	1	1	0	1
	O37	O21, O33	60	1	1	1	0	1
	O38	O21, O33	40	1	1	1	1	1
	O39	O21, O33	50	1	1	1	1	1
	O310	O21	30	1	0	0	1	0
	O311	O21, O33	45	1	1	1	1	1
	O312	N/A	20	0	1	0	0	1
	O313	O21, O33	40	1	1	1	1	1
	O314	O21, O33	40	1	1	1	1	1
	O315	O22, O32, O33	50	1	1	1	1	1
	O316	O22, O34, O39	70	1	1	1	1	1
	O317	O22, O31, O33, O37, O39	50	1	1	1	1	1
	O318	O22, O31, O33, O37, O39, O317	50	1	1	1	1	1
J4	O41	N/A	10	0	0	1	1	1
	O42	N/A	20	1	0	1	0	0
	O43	O21	20	1	1	0	1	0
	O44	O21	20	1	0	0	0	1
	O45	O21	15	1	1	1	0	0
	O46	O21	25	1	1	0	1	1
	O47	O21	35	1	1	0	0	1
	O48	O21	10	1	1	1	1	1
	O49	O21	25	1	0	1	0	1
	O410	O21	20	1	0	1	1	1
	O411	N/A	20	0	1	0	0	1
	O412	O22	30	0	1	1	0	1
	O413	N/A	20	1	1	1	1	0
	O414	O41	15	1	0	0	1	1
	O415	N/A	15	1	0	0	1	1
	O416	N/A	10	1	0	1	1	0
J5	O51	O11	10	0	1	0	1	1
	O52	O11	20	1	1	0	1	0
	O53	O21	40	1	1	1	1	1
	O54	O21	20	0	0	1	1	1
	O55	O21	20	1	1	1	1	1
	O56	N/A	5	1	0	1	0	0
	O57	N/A	10	0	0	0	1	1

In the table, the "Module" denotes categories for flight test subjects based on profession or system, such as the First Flight module, Performance module, Maneuverability and Stability module, Structural Strength module, and Overall Aircraft Environment module. The term "Subjects" refers to specific flight test subjects, including Flight Envelope Expansion, Air Data System (ADS) Calibration, Stall Speed, Takeoff Performance, Abuse Takeoff, Climb Performance, Simulated Icing, Three-axis Control, and Flutter, among others. "Preceding Subject" indicates flight test subjects that must be completed before a certain test, e.g., O22 requires completion of O33 beforehand. In actual civil aircraft flight tests, O22 represents the stalling speed for flight test subjects, while O33 represents the stall characteristics and stall warning for flight test subjects. "Hours (h)" signifies the time needed to complete each flight test mission. "Test aircraft serial number" represents the serial number of the test aircraft conducting the flight test; "1" denotes capability, while "0" indicates incapability.

In this study, we assume that five test aircraft conduct flight tests simultaneously. Past experience from previous model flight test projects suggests that the flight test intensity of a new aircraft is unevenly distributed throughout the test period. After the initial flight, the test aircraft M1 operates at a rate of 30 hours/month for the first 180 flight hours, and 40 hours/month thereafter (six months later). This forms an optimized flight test mission allocation plan with the goal of minimizing the maximum completion time (Scheme I).

Due to aircraft manufacturing capacity limitations, the second through fifth prototype aircraft need to be sequentially delayed by one month before commencing flight tests, necessitating dynamic scheduling adjustments to the initial plan. A series of optimal flight test mission allocation plans (Scheme II) is required to maximize completion time and minimize initial deviation.

5.2 Definition of Particle Swarm Optimization Parameters

It is advisable to determine the values of inertia weight, inertia weight decay coefficient, learning factor, and velocity upper and lower limits based on prior experiences.

The roles of these parameters are as follows:

a) Inertia weight: It dictates how much a particle should retain its current velocity during updates.

b) Inertia weight decay coefficient: This coefficient, applied to the inertia weights after each iteration, influences how much the inertia weights decrease over time.

c) Learning factor: This factor governs the impact of a particle's personal best position and the global best position on its velocity update.

Incorporating the inertia weight decay coefficient enhances both global and local search capabilities, preventing premature or delayed convergence and dynamically adjusting the search process.

Utilizing the inertia weight decay coefficient allows for the adoption of various search strategies throughout the optimization process. Initially, it reinforces global search efforts, transitioning to local search as the optimization progresses. This adaptive strategy enables PSO to thoroughly explore discovered solution regions while maintaining efficiency, thus increasing the likelihood of discovering high-quality solutions.

5.3 Initializing Populations

Based on the above encoding mechanism, an initial population is generated. In this paper, we set the number of particles to P=50, which means we generate 50 feasible solutions as the initial population. We use an inertia weight of $w=1$, inertia weight decay coefficient $w_{damp}=0.99$,learning factors $c_1=1.5$ and $c_2=2.0$, and conduct a total of 500 iterations.

5.4 Encoding and Decoding Mechanism

Opting for Particle Swarm Optimization (PSO) for the Flexible Job Shop Scheduling Problem (FJSP) and encoding particle positions is a crucial step. This paper addresses the correlation between test subjects and aircraft in the encoding process, drawing inspiration from the correlation between artifacts and processes. Consequently, the encoding utilizes two Z-dimensional vectors, denoted as x and y, where Z equals the total number of flight test subjects (given as 64 in the input), thus Z = 64, and m represents the number of test aircraft.

The encoding and decoding rules for x are:

a) Encodings

x is a vector containing real values where the real values are between 0 and 1. These values implicitly represent the priority of the test mission, with smaller values indicating higher priority.

Here, "implicit" means that the priority of test flight missions is not defined by explicit labels or assigned numbers, but rather by real values in the vector x. These real values fall between 0 and 1. When sorting the vector x, smaller real values are placed at the beginning, indicating that test flight missions associated with these smaller values will be executed earlier.

Although no priority number is explicitly assigned to each flight test mission, an execution sequence is implicitly defined by sorting the real values of the x-vector, which in turn determines their priority.

The mathematical expression is:

$$x=\left[x_1,x_2,\cdots,x_n\right]$$

Where, floating-point encoding $x_i\in\left[0,1\right]$, Z is the number of test missions.

b) Decoding

1) The sort function is used to sort the values in x to obtain the execution sequence of the flight test. The sort function sorts based on the magnitude of the values in x and returns the index that indicates the mission order based on these values.
2) By sorting, we have obtained a sequence of missions, in which the missions placed at the beginning have higher priority and should be executed earlier.
3) Finding the next mission whose predecessors have all been completed, ensures that any selected mission is immediately available to start, and also means that a mission whose predecessors have not yet been completed is never started, thus ensuring that the logical constraints between the flight test missions are satisfied.
4) Through the above steps, it is possible to decode the actual execution sequence of the test flight mission from the original x encoding.

The encoding and decoding rules for y are:

a) Encoding

The vector y is composed of integers, where each integer denotes the aircraft number designated for particular flight test missions. Each value in y corresponds to the aircraft assigned to carry out its respective mission.

The mathematical expression is $y=\left[y_1,y_2,\cdots,y_n\right]$, where, $y_i\in\left\{1,2,\cdots,m\right\}$, m represents the number of test aircraft and Z represents the number of test flight missions.

b) Decoding

Each element y_i of y represents the aircraft identifier associated with the corresponding mission x_i in x. This identifier determines which aircraft will execute the corresponding test flight mission. Therefore, when decoding y based on the sorting order of x, the following steps are followed:

1) For each flight test mission x_i, check that the aircraft with its associated number y_i meets the requirements of the mission.
2) For each flight test mission x_i, verify whether the aircraft associated with the code y_i meets the requirements of the mission.
3) If y_i meets the requirements of x_i, then y_i remains unchanged.
4) If y_i does not meet the requirements of x_i, then find an aircraft that can meet the requirements of x_i and update y_i to the identification number of this aircraft.
5) Through the above steps, the decoding process of y ensures that each mission x_i is associated with an aircraft numbered y_i that meets its requirements.

5.5 Adaptive Fitness Function

The adaptive function serves to modify the search direction and velocity of particles. For the FJSP, heuristic evaluation functions are commonly employed to assess the fitness of each particle. Typically, this evaluation function comprises both the objective function and constraints. In this paper, the adaptive fitness function is defined as minimizing both the minimum-maximum completion time and the initial allocation deviation. Consequently, particles with lower fitness values are closer to optimal solutions.

5.6 Neighborhood Operations

The particles generated during initialization comprise two components, x and y. In each iteration, neighborhood operations are applied separately to x and y Three types of neighborhood operations are performed randomly for x.

a) Swap

This operation randomly selects two distinct positions within the solution vector and exchanges the elements at these positions. It involves a straightforward element exchange process.

The mathematical expression for this operation is as follows:

$$x_{new}=\begin{cases}x_i,\text{if } j=k\\ x_j,\text{if } i=k\\ x_k,\text{else}\end{cases}$$

Where i and j are randomly selected two positions, while K is a position traversed from the solution vector x. The specific operation is to randomly select two positions i and j in the solution x. In the new solution x_{new}, the element x_i at position i is replaced with x_j, and the element x_h at position j is replaced with x_i, while all other elements remain unchanged.

b) Reversion

This operation randomly chooses two positions within the solution vector and then reverses the order of all elements between these positions, inclusive of the selected positions themselves. It can be viewed as the reversal of a subsequence.

The mathematical expression for this operation is as follows:

$$x_{new}=\begin{cases}x_{i+l},\text{if } k\in\left[i,l\right]\text{and } l=j-k\\ x_k,\ \text{else}\end{cases}$$

Where $\left[i,j\right]$ is a randomly selected contiguous subsequence, and k is the position traversed from the solution vector x. The specific operation is to select a random contiguous subsequence $\left[i,j\right]$ in the solution x. In the new solution x_{new}, the elements in this subsequence are reversed. That is, the element at position i becomes the element at position j in x_{new}, the element at position $i+1$ becomes the element at position $j-1$, and so on.

c) Insertion

This operation randomly selects two distinct positions within the solution vector. Subsequently, the element at the first position is removed and inserted after the second position, with the order of the remaining elements adjusted accordingly.

The mathematical expression for this operation is as follows:

$$x_{new}=\begin{cases}x_i,\quad \text{if } k=j+1\\ x_{k+1},\text{if } i<k<j\\ x_k,\quad \text{else}\end{cases}$$

Where i is the randomly selected position, while j is the new position after the insertion of element i, and k is the position traversed from the solution vector x. The specific operation is to randomly select a position i in the solution x, and then choose a position j as the new position for element x_i. In the new solution x_{new}, the element x_i is moved to position $j+1$ (i.e., after x_j), and all elements between positions i and j are shifted one position forward.

These neighborhood operations are frequently utilized in various heuristic search and optimization algorithms, particularly when addressing combinatorial optimization problems like the FJSP.

For the portion related to y, execute the mutation operation.

d) Mutation

Mutation is a technique employed to introduce novel solution spaces for exploration and uphold diversity within a population. By introducing minor, random alterations to individuals or solutions, mutation aids algorithms in escaping local optima and potentially discovering superior global optima.

The mathematical description of this operation is as follows:

Let y be an integer vector of length Z, where Z represents the number of flight test missions, and each element of y takes values in the range $[1,m]$, with m being the number of aircraft. The mutation operation is applied to the vector y by modifying two randomly selected elements.

1) Random Selection: Two independent positions, ind_1 and ind_2, are randomly selected from the set of integers $[1,Z]$. This is mathematically represented as $ind_1, ind_2 \sim U(1,Z)$, where $U(a,b)$ represents a uniform distribution over the interval $[a,b]$.
2) Random Modification: For each selected position ind_i, replace its corresponding element $y[ind_i]$ with a new random integer in the range $[1,m]$. Mathematically, it is represented as $y[ind_i] \sim U(1,m)$, $i = 1,2$.

After the procedure mentioned above, the original y-vector has now undergone mutations, with two of its elements randomly altered.

5.7 Velocity and Position of a Particle

In Particle Swarm Optimization (PSO), individual particles serve as representations of potential solutions, where each particle's position reflects the current state of a solution. Additionally, the particle's velocity embodies both the direction and magnitude of the solution's search process.

In the specific context of allocating subjects to test aircraft, each particle encapsulates a feasible allocation scheme. Here, the particle's position denotes the aircraft allocation and the implementation sequence for each subject within the scheme. Concurrently, the particle's velocity indicates its traversal speed and direction during the search process, representing the trajectory of improvement for the current solution. In FJSP, the velocity typically consists of two components: the global best solution and the individual best solution. The global best solution signifies the historically optimal scheduling solution among all particles, while the individual best solution represents the historically optimal scheduling solution for the current particle. The formula for updating velocity often incorporates these factors along with adjustment parameters such as the learning factor and inertia factor.

When evaluating the fitness function, if the current value surpasses the historical best for an individual particle (pBest), an update to pBest occurs. Similarly, if the current fitness function value exceeds the historical best across all particles (gBest), a global update to gBest is executed.

5.8 The "Boundary Checking and Velocity Reflection" Strategy

When tackling optimization problems, encountering bounded solution spaces is common, wherein solutions cannot exceed specified boundaries. In the PSO algorithm, particles update their positions based on their velocities. However, these new positions may extend beyond the boundary, leading to exploration of invalid or infeasible solutions.

To mitigate this issue, a strategy called "boundary checking and velocity reflection" is employed to ensure particles remain within the valid search space. Through this strategy, when a particle approaches or surpasses the boundary, it is not only brought back within the boundary but also its velocity is adjusted to "reflect" back into the search space.

This approach offers the advantage of confining particles within the effective search space while preserving their exploration capability, represented by their velocity. For positions beyond the boundary, the opposite of their velocity is considered. Thus, if a particle was initially on the verge of exceeding the boundary, it would now move in the opposite direction.

The "boundary check and velocity reflection" strategy enhances the effectiveness and robustness of the PSO algorithm, ensuring particles consistently explore within the valid search space and fully leverage their exploration dynamics.

The mathematical expression of the "boundary check and velocity reflection" strategy is described as follows.

a) Identifying which positions exceed the boundaries.

Given a particle location X and the search space boundaries *LowerBound* and *UpperBound*, which denote the lower and upper bounds of the solution space, respectively. It is possible to determine which elements or dimensions are beyond the boundaries. Define an indicator function I:

$$I_i = \begin{cases} 1, \text{if } X_i < LowerBound \text{ or } X_i > UpperBound \\ 0, \text{otherwise} \end{cases}$$

Where X_i is the i^{th} element of the position vector X, where $I_i = 1$ indicates that the i^{th} dimension is out of bounds.

b) Velocity reflection

For positions that exceed the boundary, take the negative value of their velocity. The velocity V of the particle can be updated as follows:

$$V_i^{'} = \begin{cases} -V_i, \text{if } I_i = 1 \\ V_i, \ \text{if } I_i = 0 \end{cases}$$

Where V_i is the i^{th} element of velocity vector V, and $V_i^{'}$ is the updated velocity.

c) Adjust the position to the boundary

For dimensions beyond the boundary, they are adjusted back to the boundary. The position x of the particle can be updated as

$$X_i^{'} = \begin{cases} LowerBound, \text{if } X_i < LowerBound \\ UpperBound, \text{ if } X_i > UpperBound \\ X_i, \qquad \text{otherwise} \end{cases}$$

Where $X_i^{'}$ is the updated position.

The mathematical expressions provided above offer a detailed explanation of how the "boundary check and velocity reflection" strategy ensures that particle positions remain within the specified boundary and adjusts particle velocity to achieve a "reflection" effect when surpassing the boundary. This strategy proves highly beneficial in optimization problems, particularly when the solution space features clearly defined boundaries.

5.9 Example Analysis

5.9.1Initial Planning Scenario (Scheme I)

In the research on flight test mission assignment schemes, we initially considered a scenario where five test aircraft are simultaneously available at the start of the flight test period. This study compares the shortened flight test period between the original ordering obtained from the initial population and the optimized ordering using the Particle Swarm Optimization (PSO) algorithm.

In the preliminary arrangement without optimization, the entire duration of the flight test period is 680 hours. This duration encompasses the entire period of flight testing, from the initiation of the first flight test mission to the completion of the last one.

From the convergence curve, it's evident that a notable characteristic of the PSO algorithm is its clear signs of convergence after approximately 50 iterations. Subsequently, after 500 iterations, the optimized solution results in a significantly reduced flight test period, now at 430 hours. This indicates a considerable enhancement in flight test efficiency through the application of the PSO optimization method, as depicted in Figure 3-5.

It's worth noting that despite the presence of unused time gaps between flight test missions across multiple aircraft, these gaps do not hinder the achievement of overall optimal results. These gaps are inherent to the complexities of flight testing and have become less significant in light of substantial improvements in overall optimization, resulting in a notable reduction in the total duration of flight testing, as emphasized.

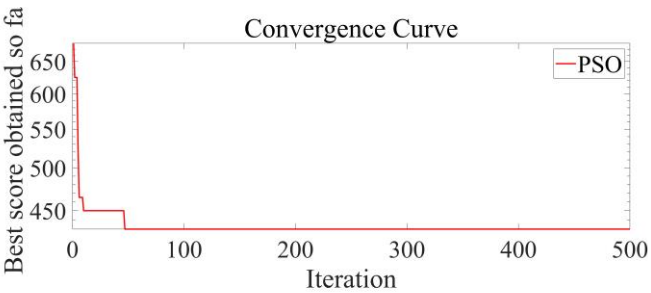

Fig. 3. Scheme I iterative convergence curve

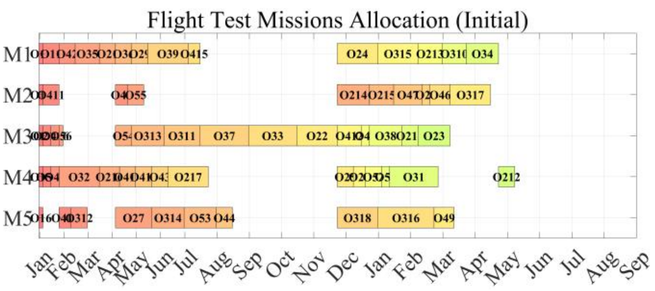

Fig. 4. Initial scheme under test flight conditions with 5 test aircraft starting simultaneously

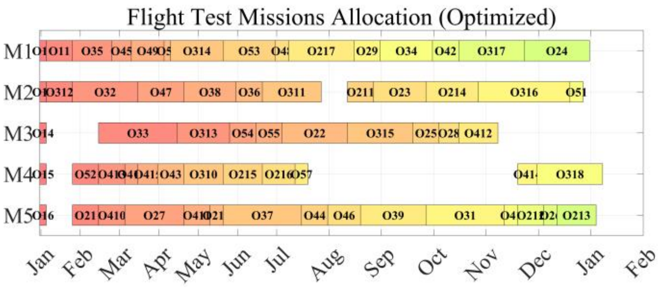

Fig. 5. The optimized scheme under the conditions of the simultaneous start of test flights of 5 test aircraft (Scheme I).

5.9.2Planning Schemes with Varying Factory Interval Scenarios (Scheme II)

In the actual process of civilian aircraft development, due to limitations in aircraft manufacturing capabilities, it is assumed that only the first aircraft is available for flight testing initially. Subsequently, starting from the first flight of the first aircraft, one test aircraft is added for flight testing every month thereafter. By the end of the fourth month, all five aircraft can be utilized for executing flight test missions.

Taking into account the real conditions observed during civil aircraft flight tests, this case study applies dynamic programming to optimize civil aircraft flight testing missions and obtain a new optimized solution with additional constraints. Considering the differing manufacturing dates of each aircraft in the initial stage, there are intervals in the first flight times of the five aircraft.

Without any optimization, the entire flight test period lasted for 635 hours, representing the overall duration of the entire flight test period, from the start of the first test flight mission to the completion of the last mission.

Through the analysis of the results of the PSO algorithm, as depicted in Figures 6-8, the optimized solution significantly reduces the overall duration of the flight test period to 430 hours. The optimized flight test period is not only shorter but also greatly reduces idle windows between different missions, indicating that the optimized solution is more efficient and reasonable.

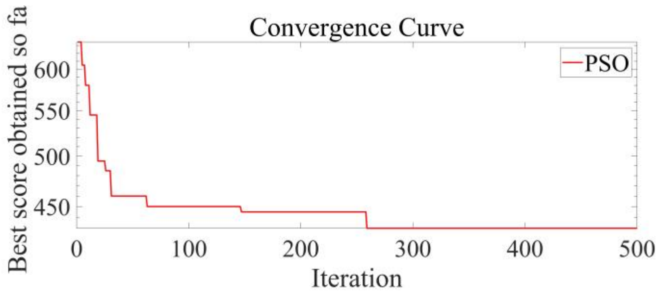

Fig. 6. Scheme II Iterative convergence curve

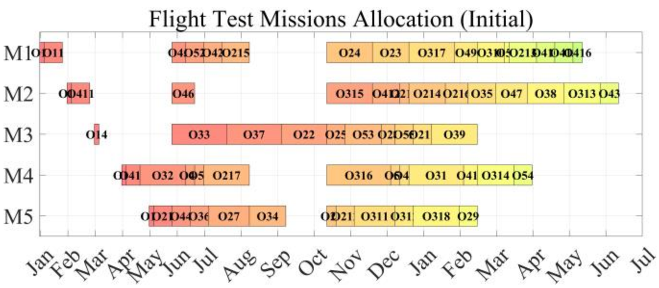

Fig. 7. Initial scheme under different factory interval conditions

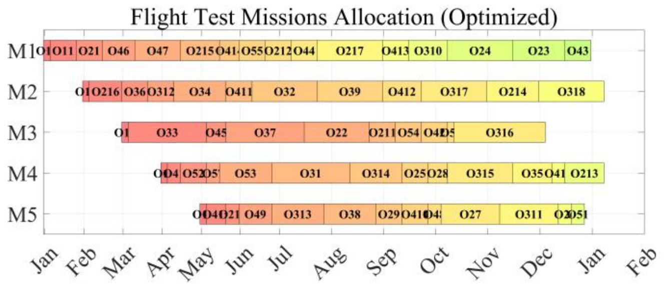

Fig. 8. Optimized scheme under different factory interval conditions (Scheme II)

As evidenced by real-life flight test data from civil aircraft, flight test project decisions necessitate robust flight test planning support. In practical scenarios, utilizing the PSO comprehensive algorithm enables optimization for both static conditions of civil aircraft test flight mission allocation and reallocation with added constraints. This leads to the development of a reasonable planning scheme aligned with assumed conditions and engineering experience, requiring relatively few iterations to achieve convergence.

For scenarios involving conditional reassignment, Scheme I is developed based on multi-objective optimization for reassignment. Under the given conditions, Scheme II is generated, which is deemed reasonable and feasible in terms of both results and iteration count. These schemes effectively meet the requirements of flight-testing planning for civil aircraft.

VI. Conclusion

The planning of flight test missions for civil aircraft is a sophisticated mission allocating a set of flight test subjects to a group of test aircraft using a combination allocation scheme. This process must adhere to a series of constraints on flight test activities. This paper develops an allocation method based on the FJSP model to address the complexity of planning flight test missions and the significant differences between this application and the standard conditions of the FJSP. A tailored model that aligns with the specificities of the flight test engineering problem is established. The PSO algorithm used in this paper solved the multi-objective flight test mission planning problem under both static and dynamic conditions. Through practical examples, the research demonstrates that the number of iterations and precision offered by the proposed approach can meet the engineering requirements. The results affirm the method's efficacy in addressing engineering-level challenges and lay the groundwork for future applications in dynamically allocating larger and more complex flight test missions.

References

[1] Feng, Kang, and X. Zhang. "Maintenance and Support Management for Type Certification Flight Test of Large Civil Aircraft." *Springer Berlin Heidelberg* (2014).

[2] Garey, M. R., and J. R. Sethi. "The Complexity of Flowshop and Jobshop Scheduling." *Mathematics of Operations Research* 1.2(1976):117-129.

[3] Liang Gao, Guohui Zhang, Wang Xiaojuan. "Intelligent Algorithms and Applications for Flexible Job Shop Scheduling". Huazhong University of Science and Technology Press (2012). pp. 3-26.

[4] Stéphane Dauzère-Pérès, W. Roux, and J. B. Lasserre. "Multi-resource shop scheduling with resource flexibility." *European Journal of Operational Research* 107.2(1998):289-305.

[5] Wang, Chun, Z. Ji, and Y. Wang. "A Novel Memetic Algorithm Based on Decomposition for Multiobjective Flexible Job Shop Scheduling Problem." *Mathematical Problems in Engineering*. 2017.

[6] Wang Chun. "Research on Models and Evolutionary Algorithms of Multi-objective Flexible Job Shop Scheduling". Jiangnan University (2018). pp. 1-16.

[7] Mastrolilli, Monaldo, and Luca Maria Gambardella. "Effective neighbourhood functions for the flexible job shop problem." *Journal of Scheduling* 3.1 (2000): 3-20.

[8] Pezzella, F., G. Morganti, and G. Ciaschetti. "A genetic algorithm for the Flexible Job-shop Scheduling Problem." *Computers & Operations Research* 35.10(2008):3202-3212.

[9] Wang, Ling, et al. "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem." *Computers & Industrial Engineering* 62.4(2012):917-926.

[10] Hmida, Abir Ben, et al. "Discrepancy search for the flexible job shop scheduling problem." *Computers & Operations Research* 37.12(2010):2192-2201.

[11] Xia, Weijun, and Z. Wu. "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems." *Computers & Industrial Engineering* 48.2(2005):409-425.

[12] Gao, Jie, et al. "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems." *Comput. Ind. Eng*. 53(2007):149-162.

[13] Zhang, Guohui, et al. "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem." *Computers & Industrial Engineering* 56.4(2009):1309-1318.

[14] Abdellah, Aissou, A. Daamouche, and M. Hassan. "Components Assignment Problem for Flow Networks Using MOPSO." *IAENG International Journal of Computer Science* 48.1(2021):96-108.

[15] Chen, Gonggui, et al. "Quasi-oppositional cuckoo search algorithm for multi-objective optimal power flow." *IAENG International Journal of Computer Science* 45.2(2018):255-266.

[16] Tamura, Keiichi, and H. Kitakami. "A New Distributed Modified Extremal Optimization using Tabu Search Mechanism for Reducing Crossovers in Reconciliation Graph and Its Performance Evaluation." *IAENG International Journal of Computer Science* 41.2(2014):131-140.

[17] Shen, Xiao-Ning, and X. Yao. "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems." *Information Sciences* 298(2015):198-224.

[18] Wang Xiaojuan, "Research on Methods for Multi-objective Flexible Job-shop Scheduling Problem". Huazhong University of Science and Technology (2011). pp. 68-82.

[19] Raman, Narayan, and F. B. Talbot. "The job shop tardiness problem: A decomposition approach." *European Journal of Operational Research* 69.2(1993).

[20] Sabuncuoglu, Ihsan, and S. Karabuk. "Rescheduling frequency in an FMS with uncertain processing times and unreliable machines." *Journal of Manufacturing Systems* 18.4(1999):268-283.

[21] Kennedy, J., and R. Eberhart. "Particle Swarm Optimization." *Icnn95-international Conference on Neural Networks* IEEE, 1995.

[22] Eberhart, R., and J. Kennedy. "A new optimizer using particle swarm theory." *Mhs95 Sixth International Symposium on Micro Machine & Human Science IEEE*, 2002.

[23] Tutuko, Bambang, et al. "Route Optimization of Non-holonomic Leader-follower Control Using Dynamic Particle Swarm Optimization." *IAENG International Journal of Computer Science* 46.1(2019):1-11.

[24] Castillo, Oscar, et al. "Optimization of the Fuzzy C-Means Algorithm using Evolutionary Methods." *Engineering Letters* 20.1(2012):1-5.

[25] Lv, Sheng, and Lihong Qiao. "Process planning and scheduling integration with optimal rescheduling strategies." *International Journal of Computer Integrated Manufacturing* 27.7 (2014): 638-655.