

## Article

# Flight Schedule Problem Optimization Based on Discrete Memory-Enhanced Restructured Particle Swarm Optimization Algorithm

Wei Gao <sup>1</sup>, Bingnan Wu <sup>2</sup>, Jianhua Liu <sup>2,\*</sup> and Daoming Tang <sup>2</sup>

<sup>1</sup> School of Artificial Intelligence, Fuzhou Technology and Business University, Fuzhou 350715, China

<sup>2</sup> School of Computer Science and Mathematics, Fujian University of Technology, Fuzhou 350118, China

\* Correspondence: jhliu@fjnu.edu.cn

## Abstract

Flight Schedule Problem optimization is a typical NP-hard combinatorial optimization problem that is challenging to solve using traditional algorithms, so metaheuristic algorithms are commonly adopted for such problems. This paper proposes a Discrete Memory-Enhanced Restructured Particle Swarm Optimization algorithm (DMERPSO) to address Flight Scheduling Problem optimization. Firstly, this paper designs a hybrid particle encoding scheme capable of simultaneously handling flight time adjustments (integer variables) and route selections (categorical variables) for the Flight Schedule Problem. Secondly, a new update equation of particle positions is provided based on probability selection within the three terms of the Memory-Enhanced Restructured Particle Swarm Optimization (MERPSO) algorithm, and the calculation of the selection probability is designed. Thirdly, the two strategies and perturbation terms of MERPSO are improved in order to be adapted to optimize the discrete Flight Schedule Problem. Finally, simulation experiments are conducted using DMERPSO based on real flight data from multiple Chinese airports with the objective of minimizing total flight delays, leading to better solutions that are faster than various benchmark algorithms. The DMERPSO algorithm exhibits significant advantages in reducing total delays, improving solution stability, and enhancing robustness, which validates that DMERPSO provides an effective new approach for solving Flight Schedule Problem optimization problems.

**Keywords:** flight schedule optimization; restructured particle swarm optimization; discrete algorithm; memory-enhanced



Academic Editor: Frank Werner

Received: 25 December 2025

Revised: 17 March 2026

Accepted: 18 March 2026

Published: 19 March 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

## 1. Introduction

Flight Schedule Problem optimization is aimed at efficiently allocating resources, such as aircraft and crews, in order to enhance overall efficiency, which represents a complex combinatorial optimization problem constrained by aircraft routes, runway slot capacities, and turnaround requirements [1]. Traditional methods for solving the Flight Schedule Problem rely primarily on mathematical programming and precise algorithms [2]. The classical paradigm of traditional Flight Schedule Problem optimization is centered on mathematical programming, particularly relying on Mixed-Integer Linear Programming (MILP) to construct precise deterministic models [3]. This paradigm rigorously formulates objective functions—such as maximizing the total operational profit or minimizing overall costs—alongside a set of hard constraints [4]. Because the Flight Schedule Problem is

an NP-hard problem, solving it primarily depends on exact algorithms such as branch-and-bound [5] and branch-and-price [6]. The branch-and-price algorithm decomposes the problem into a master problem (selecting optimal flight strings/crew pairings) and pricing subproblems (generating new feasible task columns within resource-constrained networks), thereby effectively handling large-scale combinatorial spaces. The seminal work of Barnhart et al. [7] systematically established this framework, laying the algorithmic foundation for theoretical research on core issues such as fleet assignment and crew scheduling. However, pure methods of this kind often face computational intractability when addressing ultra-large-scale problems, multi-stage integration or dynamic stochastic disruptions. In response, current research frontiers focus on enhancing the scalability and practicality of the traditional paradigm through several key avenues [8–10]. These innovations allow the traditional cornerstone of mathematical programming to sustain its vitality, offering interpretable and high-performance solutions for complex scheduling problems through integration with modern computational intelligence.

However, with the increase in the scale of the Flight Schedule Problem, these traditional algorithms have become computationally impractical. To overcome the computational bottlenecks of traditional methods, both academic and industrial communities commonly employ metaheuristic algorithms to obtain high-quality approximate optimal solutions, which demonstrate strong compatibility with the large-scale and high-dimensional characteristics of Flight Schedule Problem optimization problems [11,12]. Metaheuristic algorithms have emerged as a significant computational paradigm in modern combinatorial optimization and have been widely and successfully applied in the field of Flight Schedule Problem [13]. Their core advantages lie in their strong generality, low dependency on specific problem structures, and their ability to effectively explore vast solution spaces. For example, the Genetic algorithm approach was used to build feasible daily schedules and assign optimal aircraft for hub-and-spoke airlines [14]; Ant Colony Optimization mimics the pheromone communication of ants to find optimal paths through positive feedback, making it particularly suitable for flight string or aircraft routing optimization [15]; and Tabu Search utilizes memory structures to avoid cyclical searching, which has been successfully applied to flight schedule development, tail assignment, and crew scheduling [16].

The Particle Swarm Optimization (PSO) algorithm is a metaheuristic approach which is characterized by its conceptual simplicity, fast convergence speed, and strong global search capability, and has demonstrated significant potential in solving complex optimization problems [17]. However, when addressing Flight Schedule Problem optimization involving numerous discrete variables and complex multimodal search spaces, the standard PSO algorithm's inherent "global best" guidance mechanism often leads to premature loss of population diversity, thereby trapping the algorithm in local optima—a phenomenon known as "premature convergence". To address these limitations, researchers have explored algorithmic improvements through two primary approaches, for example, integrating PSO with neighborhood search [18].

In recent years, the Restructured Particle Swarm Optimization (RPSO) algorithm was developed based on the linear system theory of PSO, which kept the optimization performance alongside algorithmic simplicity [19]. RPSO simplified a standard PSO framework by designing a new particle position updating formulation with convergent term and perturbation term, introduced novel search mechanisms [20]. RPSO has relatively good convergence theoretical support and is simple and easy to use. However, RPSO loses the population's memory, including the historical positions and fitness information of the particles. A variant of RPSO was designed, the memory-enhanced reconfigured particle swarm algorithm (MERPSO), which uses an experience selection strategy and a block

search strategy to store the memory, constructing two new learning samples to replace the original learning samples so to balance local development and global exploration [21].

However, the original RPSO algorithm was primarily designed for continuous optimization problems, whereas Flight Schedule Problem optimization involves discrete decision variables. Its reliance on a single global best still risks premature convergence, particularly in highly multimodal problems such as Flight Schedule Problem optimization. This paper presents a discretized RPSO by transforming the variant of RPSO, the MERPSO algorithm, into a Discrete Memory-Enhanced Restructured Particle Swarm Optimization algorithm (DMERPSO), which is then applied to solve single-objective Flight Schedule Problem optimization. Experimental results demonstrate significant improvements in both computational efficiency and solution reliability. The main contributions of this work are as follows:

- (1) The deep analysis of the model of Flight Schedule Problem optimization is conducted so that a complete flight route can be mathematically described. For the Flight Schedule Problem optimization model to be solved by the RPSO algorithm, the decision variables affecting flights are obtained and the *equation of* particle expression is designed. A hybrid particle encoding scheme capable of simultaneously handling integer-type (adjustment time) and categorical-type (route selection) decision variables is designed.
- (2) A probability-based discrete position update formula is developed, enabling particles to probabilistically select between “exploitation” samples (*sg*), “exploration” samples (*mp*), and random perturbation terms (*dt*) to achieve balanced trade-offs between exploitation and exploration in discrete solution spaces.
- (3) The core learning strategies are restructured through discretization: the “mean” operation in continuous EDS was transformed into a “voting/majority rule” mechanism for discrete EDS; the “hyper-rectangular block” structure in continuous BSS was reformulated into adaptive “integer interval blocks” and “discrete set blocks” suitable for different variable types, while preserving its core concept of dynamic splitting and merging.

## 2. Memory-Enhanced Restructured Particle Swarm Optimization Algorithm

This section first introduces the standard Particle Swarm Optimization (PSO) algorithm and its significant variant—Restructured Particle Swarm Optimization (RPSO). Then, it elaborates on how the Memory-Enhanced Restructured Particle Swarm Optimization (MERPSO) algorithm significantly improves performance by incorporating the memory-enhancement mechanisms of three terms: “experience selection”, “block search” and “random perturbation”.

### 2.1. Restructured Particle Swarm Optimization (RPSO)

PSO was inspired by the foraging behavior of bird flocks in 1995 [22]; the classical equations for updating velocity and position are as the Equations (1) and (2):

$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 r_1 (pbest_{id}(t) - x_{id}(t)) + c_2 r_2 (gbest_d(t) - x_{id}(t)) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

where *pbest* is the particle’s own historically optimal position and *gbest* is the globally best-known position of the entire swarm;  $v_{id}(t)$  and  $x_{id}(t)$  represent the velocity and position of particle *i* in the *d*-th dimension during the *t*-th iteration, respectively; *w* denotes the inertia weight; *c*<sub>1</sub> and *c*<sub>2</sub> are learning factors; *r*<sub>1</sub> and *r*<sub>2</sub> are random numbers uniformly distributed

within  $[0, 1]$ . The PSO is highly susceptible to premature convergence toward a non-global optimal  $gbest$ , leading to diversity loss and entrapment in local optima [23].

Based on the results of linear system theoretical analyses of PSO, the Restructured Particle Swarm Optimization (RPSO) algorithm was proposed [19,20], which employs only a single position update equation, as shown in Equation (3):

$$x_{i+1} = c_r \times pbest + (1 - c_r)gbest + G(t) \quad (3)$$

where  $c_r$  represents the acceleration coefficient, a random number within the interval  $[0, 1]$ , and  $G(t)$  denotes the perturbation term that can be designed by the user. For instance, the calculation method for  $G(t)$  was formulated as in Equation (4) [19]:

$$G(t) = (s \times r_3) \times (0.8 + \omega) \times (1 - \frac{t}{T}) \quad (4)$$

Here,  $s$  is set to one-quarter of the problem search range,  $r_3$  is a random number within the interval  $[-1, 1]$ ,  $\omega$  is similar to the inertia weight of the PSO algorithm, with its value linearly decreasing between 0.2 and 0.8,  $t$  denotes the current iterating generation, and  $T$  signifies the maximum iteration. The RPSO algorithm simplifies the parametric structure and directly “restructures” new particle positions, wherein each particle has a certain probability of either learning from the global best particle ( $gbest$ ) and the personal best particle ( $pbest$ ) to take on the exploitation, while  $G(t)$  controls the exploration which is set by the designer.

## 2.2. Memory-Enhanced Restructured Particle Swarm Optimization (MERPSO)

RPSO remains dependent on the single  $gbest$ . Therefore, if  $gbest$  is trapped in a local optimum of the searching area, the evolutionary trajectory of the entire particles will still be severely misguided. To address the issue of RPSO’s over-reliance on  $gbest$ , a Memory-Enhanced Restructured Particle Swarm Optimization (MERPSO) algorithm is proposed [21]. Effective guiding information should not be derived solely from the current single “champion”  $gbest$ , but rather from the collective wisdom crystallized by all “elites” of the entire search history.

### 2.2.1. Core Idea and Update Mechanism

In MERPSO, the concepts of  $pbest$  and  $gbest$  are completely abandoned, and the position update mechanism is transformed into a probabilistic selection-based learning approach. The position updates of particles are determined through probabilistic selection from three “learning samples”, representing distinct search strategies, instead of by velocity and displacement [21]. The position update formula is as shown in Equation (5):

$$x_{i,d}^{t+1} = \begin{cases} sg_d^t, & \text{if } rand() \leq P_{sg} \\ mp_{i,d}^t, & \text{if } P_{sg} < rand() \leq p_{sg} + p_{mp} \\ dt_{i,d}^t, & \text{otherwise} \end{cases} \quad (5)$$

where  $x_{i,d}^{t+1}$  denotes the new position of particle  $i$  in the  $d$ -th dimension at the  $(t + 1)$ th iteration. The three core learning samples are defined as follows: (1) The experience-guided sample ( $sg$ ), which represents exploitation of promising regions. This sample is generated from an “experience set” storing numerous historical optimal solutions. (2) The block exploration sample ( $mp$ ), representing exploration of uncharted regions. This sample is generated through a “block search strategy” designed to guide particles toward exploring less visited novel regions of the solution space. (3) The random perturbation term ( $dt$ ),

which introduces minor stochastic variations to the current particle position for maintaining population diversity and preventing premature algorithm stagnation.

### 2.2.2. Core Learning Strategies

The generating quality of *sg* and *mp* of the MERPSO algorithm is critical, which incorporates two sophisticated memory-enhanced strategies.

#### (1) Experience Selection Strategy (EDS)

The core of the EDS strategy lies in maintaining an external archive called the Experience Set (ES), which maintains the top  $Np$  solutions with the best fitness performance throughout the entire search history. The ES is updated after each iteration following fitness evaluation, preserving the most promising positions. The EDS strategy generates learning samples from the ES, denoted as SG. Each dimension of SG is independently determined according to Equation (6), meaning that every dimension of SG is computed as the average value of that dimension across  $K$  randomly selected solutions from the ES. In this algorithm, the value of  $K$  is set to 10.

$$sg_d = \frac{1}{K} \times \sum_{k=1}^K ES_d(r_k) \quad (6)$$

where  $ES_d(r_k)$  denotes the  $d$ -th dimension of the  $r_k$ -th solution in the ES, and  $r_k$  is a random integer within the interval  $[1, Np]$ . To enhance the algorithm's exploitation capability in the later stages,  $Np$  is gradually reduced to narrow the scope of solutions preserved in the experience set.  $Np$  is calculated according to Equation (7):

$$N_p = \left\lceil ps \times \left(1 - \frac{t-1}{T}\right) \right\rceil \quad (7)$$

where  $ps$  represents the population size,  $\lceil \cdot \rceil$  denotes the ceiling operation,  $t$  is the current generation number, and  $T$  is the maximum iteration limit.

ESD stores the most historically fit solutions discovered by the algorithm since its initiation. When generating *sg* samples to guide the entire population, the algorithm randomly selects multiple elite historical solutions from the ES and forms a "consensus" direction by calculating their mean.

#### (2) Block Search Strategy (BSS)

The block search strategy (BSS) dynamically partitions the entire multi-dimensional search space into a series of hyper-rectangular "blocks" and continuously monitors the frequency of particle visits to each block. The block search strategy (BSS) generates learning samples, denoted as MP. The method involves each particle  $X_i$  having its corresponding learning sample, labeled as  $mp_i$ . The vector  $mp_i$  is generated by randomly combining its personal best  $pb_i$  and a randomly selected position  $pr$  in the block, as determined by Equation (8):

$$mp_{i,d} = \begin{cases} pb_{i,d}, & rand > q_i \\ pr_d\{b\}, & else \end{cases} \quad (8)$$

where  $mp_{i,d}$  denotes the position of the  $d$ -th dimension in the  $i$ -th learning sample  $mp_i$ . For each dimension, it has a probability  $q_i$  of being  $pr_{i,d}\{b\}$  and a probability  $(1 - q_i)$  of being  $pb_i$ . Here,  $pr_{i,d}\{b\}$  represents a random position within the  $b$ -th block of the  $d$ -th dimension;  $b$  is the selection result from a roulette wheel choice among all blocks in that dimension. The selection weight for each block is given by  $W_b = M_b - n_p$ , where  $n_p$  is the number of particles within that block and  $M_b$  is the upper limit for the number of particles in a block. For the algorithm's exploration, BSS tends to generate exploratory samples *mp* from

those “less visited” (low-visit-frequency) blocks, thereby guiding particles to explore new search territories.

### (3) The random perturbation term (DT)

The learning sample SG generated by EDS enhances the algorithm’s exploitation capability, while the learning sample MP produced by BSS strengthens its exploration capability. The position update equation of particles in the MERPSO algorithm is given by Equation (9):

$$X_i^{t+1} = c'_s \times sg^t + c_s \times mp_i^t + D(t) \quad (9)$$

$$D(t) = (space \times r) \times (0.8 + \omega) \times (1 - \frac{t}{T})^{(C_m \times \frac{t}{T})} \quad (10)$$

where  $X_i^{t+1}$  represents the position of the  $i$ -th particle at generation  $t + 1$ .  $SG_t$  denotes the SG selected at generation  $t$ , and  $mp_i^t$  is the  $i$ -th MP at generation  $t$ .  $D(t)$  is a time-varying perturbation term, calculated according to Equation (10). The parameter  $space$  is set to 20% of the problem’s search space,  $r$  is a random number uniformly distributed in the interval  $[-1, 1]$ ,  $\omega$  is the inertia weight ranging from 0.8 to 0.2, and  $C_m$  is the perturbation coefficient.

## 3. Discrete MERPSO Algorithm for Solving Single-Objective Flight Schedule Problem Optimization

The MERPSO algorithm demonstrates competitive performance in continuous optimization problems. However, the Flight Schedule Problem constitutes a discrete optimization problem characterized by two functionally distinct decision variables: adjusted times and selected routes. So that the MERPSO algorithm can solve Flight Schedule Problem optimization, this section designs Discrete Memory-Enhanced Restructured Particle Swarm Optimization (DMERPSO), which involves the strategies of constructing particle representations, reformulating position update equations, designing learning strategies, incorporating perturbation terms, and implementing parameter discretization.

### 3.1. The Representation of Flight Schedule Problem Optimization for RPSO Algorithm

#### 3.1.1. A 4D Trajectory Model for the Flight Schedule Problem Optimization

The primary flight schedule information includes the number of flights, departure and arrival locations, scheduled departure times, estimated arrival times, and flight routes. Table 1 presents a basic flight schedule.

**Table 1.** An example of flight schedule.

| Flight Number | Departure Airport | Landing at the Airport | Estimated Time of Departure | Estimated Time of Landing | Flight Routes                                |
|---------------|-------------------|------------------------|-----------------------------|---------------------------|----------------------------------------------|
| $F_1$         | Xi'an             | Beijing                | 9:00                        | 11:20                     | $r^{(1,1)}, r^{(1,2)}, r^{(1,3)}$            |
| $F_2$         | Shanghai          | Shenzhen               | 10:15                       | 12:30                     | $r^{(2,1)}, r^{(2,2)}$                       |
| $F_3$         | Xi'an             | Shanghai               | 18:20                       | 21:15                     | $r^{(3,1)}, r^{(3,2)}, r^{(3,3)}, r^{(3,4)}$ |
| $F_4$         | Beijing           | Shenzhen               | 7:45                        | 9:10                      | $r^{(4,1)}$                                  |
| $F_5$         | Shenzhen          | Xi'an                  | 11:35                       | 15:00                     | $r^{(5,1)}, r^{(5,2)}, r^{(5,3)}$            |

Note: The Set of flights number are denoted as  $F = \{F_1, F_2, F_3, \dots, F_{N_f}\}$ , where  $N_f$  is the total number of flights;  $r^{(i,j)}$  denotes the  $j$ -th route of the  $i$ -th flight  $F_i$ .

The routes in flight scheduling are planned with one or more candidate routes, and different routes may exist for the same origin and destination. During the actual implementation of flight operations, the flight schedule often considers factors of adverse weather conditions and airspace congestion levels. While the departure location (*orig*) and landing location (*des*) remain unchanged, the actual departure time  $\tau^{\text{orig}}$  is dynamically adjusted according to real-time situations. The arrival time of a flight  $\tau^{\text{des}}$  is influenced by multiple

factors, including the route distance, flight speed  $v^f$ , and congestion level, at the destination airport. In practical flight execution, routes are selected based on operational requirements.

Evaluating the flight schedule requires real-time position information for each flight within the flight plan. The 4D trajectory model is a commonly used framework which encompasses three spatial dimensions (flight position coordinates) and one temporal dimension (time information). An aircraft's flight path along route  $r$  can be conceptualized as straight-line segments between waypoints ( $wp$ ) as navigation checkpoints. The connecting paths between waypoints are termed flight segments  $L$ . A complete flight route  $r^{(i,j)}$  can be mathematically described using either waypoints or flight segments using Equation (11):

$$r^{(i,j)} = (orig^i, wp_1^{(i,j)}, wp_2^{(i,j)}, \dots, des^i) = (L_1^{(i,j)}, L_2^{(i,j)}, \dots) \in \Gamma^{(i,j)} \quad (11)$$

where  $orig^i$  and  $des^i$  denote the origin and destination of flight  $F_i$ ,  $\Gamma^{(i,j)}$  represents the set of flight route, and Figure 1 illustrates a basic example of a flight route. Once the flight schedule parameter  $\tau_i^{orig}$ , the decision variable  $S_i$  for route selection, and the flight speed  $v_i^f$  are determined, the flight trajectory can be calculated accordingly.

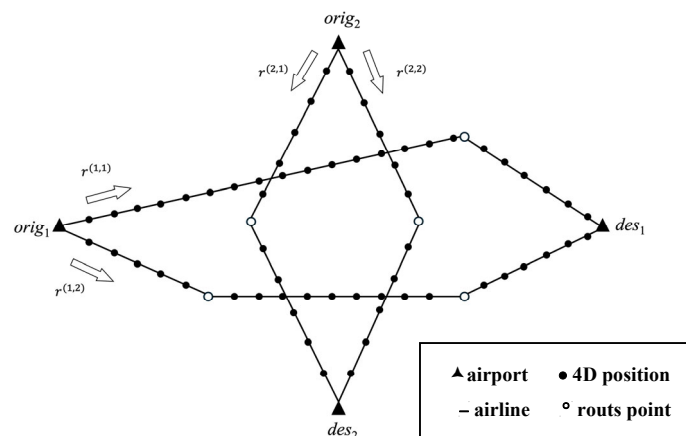


**Figure 1.** Flight route map.

### 3.1.2. Particle Representation and Flight Position Calculation in the RPSO Algorithm

For the Flight Schedule Problem optimization model, the decision variables affecting flights include the adjustment time of departure  $t^a$  and the selected route  $S^r$  for each flight. A particle  $X$  in the RPSO algorithm represents an adjustment scheme for the flight schedule, denoted as  $X \leq t^s$ . The dimension  $D$  of particle  $X$  equals  $2N_f$ , so the  $i$ -th particle  $X_i$  can be expressed by Equation (12):

$$X_i = \langle t_{(i,1)}^a, t_{(i,2)}^a, \dots, t_{(i,N_f)}^a, S_{(i,1)}^r, S_{(i,2)}^r, \dots, S_{(i,N_f)}^r \rangle \quad (12)$$

where  $t_{(i,1)}^a$  denotes the adjusted departure time (in integer minutes) for the first flight of the  $i$ -th particle. Flight departure times are typically set at 5 min intervals, where  $t^a = 3$  indicates a 15 min delay and  $t^a = -4$  corresponds to a 20 min advance.  $S_{(i,1)}^r$  represents the selected route for the first flight of particle  $i$ .

Since the altitude of flights is neglected and only their horizontal positions are considered, once the adjustment scheme  $X$  is determined, the position of flight  $j$  can be calculated using Equations (13)–(16):

$$T_j(wp_j^0) = \tau^{orig} + 5 \times t_{(i,j)}^a = T_j(orig_j) \quad (13)$$

$$T_j(wp_j^k) = T_j(wp_j^{k-1}) + \frac{\text{len}(wp_j^k - wp_j^{k-1})}{v_j^f}, j = 1, 2, \dots, Nf \quad (14)$$

$$\text{dis} = 2R \times \arcsin(\sqrt{\sin^2((\text{lat}B - \text{lat}A)/2) + \cos(\text{lat}A) \times \cos(\text{lat}B) \times \sin^2((\text{lon}B - \text{lon}A)/2)}) \quad (15)$$

$$\text{loc}_j(t^s) = \frac{t^s - T_j(wp_j^{z-1})}{T_j(wp_j^z) - T_j(wp_j^{z-1})} \times (wp_j^z - wp_j^{z-1}) \quad (16)$$

where  $T_j(wp_j^k)$  denotes the time when flight  $j$  passes the  $k$ -th waypoint,  $\tau_j^{\text{orig}}$  represents the scheduled departure time in the flight plan,  $T_j(\text{orig}_j)$  indicates the adjusted departure time of flight  $j$ , and the arrival time is denoted as  $T_j(\text{des}_j)$ ;  $v_j^f$  signifies the average flight speed of flight  $j$  provided by the dataset;  $\text{len}(A-B)$  is a function calculating the distance between two waypoints  $A$  and  $B$  using the Haversine formula as Equation (15), where  $R$  represents the Earth's radius (6371 km), and  $\text{lat}A$  and  $\text{lon}A$  respectively indicate the latitude and longitude coordinates of point  $A$  measured in radians;  $\text{loc}_j(t^s)$  specifies the position of flight  $j$  at timestamp  $t^s$ , with  $t^s$  being discrete time points separated at 5 min intervals.

### 3.1.3. The Objective and Constraint Conditions of Flight Schedule Problem Optimization

The objective of Flight Schedule Problem optimization takes minimizing flight delay time (Delay) as the optimization objective, which consists of total ground delay ( $T^{\text{dep}}$ ) and total arrival delay ( $T^{\text{arr}}$ ). The calculation method is shown in Equations (17)–(19):

$$\text{minDelay} = T^{\text{dep}} + T^{\text{arr}} \quad (17)$$

$$T^{\text{dep}} = \sum_{j=1}^{Nf} (5 \times t_j^a) \quad (18)$$

$$T^{\text{arr}} = \sum_{j=1}^{Nf} (T_j(\text{des}_j) - \tau_j^{\text{des}}) \quad (19)$$

The constraints of Flight Schedule Problem optimization primarily consider the utilization of airport runways, with the constraint conditions shown as Equations (20)–(23):

$$N_{ap}^{\text{dep}}(t^s) \leq Q_{ap}^{\text{dep}}, \forall ap \in \Omega, \forall t^s \in [0, T^{\text{max}}] \quad (20)$$

$$N_{ap}^{\text{arr}}(t^s) \leq Q_{ap}^{\text{arr}}, \forall ap \in \Omega, \forall t^s \in [0, T^{\text{max}}] \quad (21)$$

$$|t_{j,ap} - t_{j+1,ap}| \geq \tau_{\text{sep}}, i = 1, \dots, Nf - 1, \forall ap \in \Omega \quad (22)$$

$$t^a \in [T_{\text{min}}^a, T_{\text{max}}^a] \quad (23)$$

where all airports  $ap$  constitute the set  $\Omega$ . For each airport, the number of flights simultaneously taking off and landing within a given period is subject to upper limits, known as the airport's departure capacity  $Q_{ap}^{\text{dep}}$  and arrival capacity  $Q_{ap}^{\text{arr}}$ ; therefore, Equations (20) and (21) ensure that the number of departing and arriving flights at a given airport do not exceed its respective capacity. Furthermore, the interval between two consecutive departures at the same airport must exceed a specified threshold  $\tau_{\text{sep}}$ , so Equation (22) guarantees that the separation time between consecutive departures at the same airport adheres to the safety threshold  $T_{\text{sc}}$ , which is set to 5 min. Equation (23) constrains the adjustment time for an individual flight within the schedule to a reasonable operational range;  $T_{\text{min}}^a, T_{\text{max}}^a$  respectively represent the minimum and maximum values of the flight schedule adjustment.

### 3.2. Discrete MERPSO Algorithm

The MERPSO algorithm framework was originally proposed to address continuous optimization problems, whereas discrete optimization problems possess a different feasible solution space. This section extends the MERPSO framework to the Discrete MERPSO (DMERPSO) so to solve discrete optimization problems. Firstly, the methodology of the DMERPSO algorithm is designed; secondly, the discretization processing of each strategy of the algorithm is introduced; finally, the particle initialization method is presented.

#### 3.2.1. Position Update Formula of DMERPSO

The discretization method of the PSO algorithm employs a transfer function to map velocity values onto the interval  $[0, 1]$ . Subsequently, these mapped values within the  $[0, 1]$  interval are converted into binary values (0 and 1) through stochastic probability.

Since the MERPSO algorithm does not require particle velocity for position updates, the position update of particles in the DMERPSO algorithm is influenced by three components: the learning samples  $sg$  and  $mp$ , as well as the perturbation term  $dt$ . Specifically, sample  $sg$  enhances both the convergence speed and precision of the particle,  $mp$  strengthens the particle's exploration capability, while  $dt$  maintains particle diversity. These three components directly take on particle position updates while serving distinct functional roles. Therefore, MERPSO directly updates particle positions based only on probability. The corresponding particle update equation is presented in Equation (24):

$$x_{i,d}^{t+1} = \begin{cases} sg_d^t, & rand \leq P_{i,d}^{sg} \\ mp_{i,d}^t, & P_{i,d}^{sg} < rand \leq (P_{i,d}^{sg} + P_{i,d}^{mp}) \\ dt_{i,d}^t, & other \end{cases} \quad (24)$$

where the update value of each dimension of a particle is selected from the corresponding dimensions of  $sg$ ,  $mp$ , or  $dt$  with selection probabilities  $P^{sg}$ ,  $P^{mp}$ , and  $P^{dt}$  respectively, where the sum of these three probabilities equals 1. The probability of selecting each term is determined based on factors including particle characteristics and the population's search status.

#### 3.2.2. Calculation of Selection Probability

The DMERPSO algorithm determines the probability of updating particle positions by evaluating and scoring multiple scenarios, followed by normalizing the evaluated scores to obtain probabilities, thereby ensuring that the sum of the three probabilities equals 1. Each evaluation score can be expressed as  $SC = W$  (weight value)  $\times B$  (base score). The parameter values or ranges in this section are presented in Table 2. The three selection probabilities  $P^{sg}$ ,  $P^{mp}$ , and  $P^{dt}$  in Equation (25) are calculated with Equations (26)–(28).

$$P^l = \frac{SC^l}{SC^{sg} + SC^{mp} + SC^{dt}}, l \in \{sg, mp, dt\} \quad (25)$$

$$SC^{sg} = w_1 \times b_{sg} \quad (26)$$

$$SC^{mp} = w_1 \times b_{mp} \quad (27)$$

$$SC^{dt} = \frac{w_2}{w_1} \times B_3 \quad (28)$$

where the weight  $w_1$  is a parameter that linearly iterates with the number of generations within the interval  $[0.5, 2]$ , which makes particles to undergo random mutation with a higher probability during the early stages, thereby expanding the population's search scope while reducing the mutation probability in later stages so to enhance the convergence

precision. The weight  $w_2$  governs the maintenance of convergence in one-dimensional space, where particles remain identical to both learning samples  $sg$  and  $mp$  during the pursuit of convergence accuracy in later evolutionary stages, which is calculated using Equation (29).

$$w_2(i, d) = \begin{cases} 0, & x_{i,d} = sg_{i,d} = mp_{i,d} \text{ \& } t > \frac{T}{4} \\ 1, & \text{other} \end{cases} \quad (29)$$

**Table 2.** Relevant parameters of selection probability.

| Parameter | Size or Range |
|-----------|---------------|
| $w_1$     | [0.5, 2]      |
| $w_2$     | 0 or 1        |
| $B_1$     | 20            |
| $B_2$     | 100           |
| $B_3$     | 20            |

The scores  $b_{sg}$  and  $b_{mp}$  function similarly to offset parameters in continuous MERPSO algorithms, computed through Equations (30) and (31), which regulate particles to exhibit greater displacement probability during initial exploration phases while maintaining positional stability or enabling minor adjustments with increased probability in subsequent optimization stages. In the two equations,  $ps$  denotes the number of particles of each generation of the algorithm, and  $rank(f(X_i))$  is the sorting number of fitness of the objective function for the  $i$ th particle with the population.

$$b_{sg}(i, d) = \begin{cases} b_2 \times \frac{rank(f(X_i))}{ps}, & t < \frac{T}{2} \text{ \& } dis_s \geq dis_m \text{ \& } d \leq Nf \\ b_2 \times \left[1 - \frac{rank(f(X_i))}{ps}\right], & t \geq \frac{T}{2} \text{ \& } dis_s < dis_m \text{ \& } d \leq Nf \\ B_1, & \text{other} \end{cases} \quad (30)$$

$$b_{mp}(i, d) = \begin{cases} b_2 \times \frac{rank(f(X_i))}{ps}, & t < \frac{T}{2} \text{ \& } dis_s < dis_m \text{ \& } d \leq Nf \\ b_2 \times \left[1 - \frac{rank(f(X_i))}{ps}\right], & t \geq \frac{T}{2} \text{ \& } dis_s \geq dis_m \text{ \& } d \leq Nf \\ B_1, & \text{other} \end{cases} \quad (31)$$

Since the particle consists of two components, the adjustment time and trajectory, for the former  $dis_m(i, d) = |x_{i,d} - mp_{i,d}|$  and  $dis_s(i, d) = |x_{i,d} - sg_d|$ , larger values indicate greater temporal discrepancies between the particle's corresponding dimension and the learning samples, which carries practical physical significance; for the latter component, differences between these two values merely reflect distinct route selections, making comparative magnitude analysis meaningless. Consequently, the scoring mechanisms for these two particle components differ, with parameter  $b_2$  calculated using Equation (32).

$$b_2 = B_2 * \left| \frac{1}{1 + e^{0.01 \times (t - \frac{T}{2})}} - 0.5 \right| + B_1 \quad (32)$$

### 3.2.3. Improvement of Strategies and Perturbation Terms

Although the previous subsection adapted the position update formula of the MERPSO algorithm to Equation (24) and introduced a probability-based selection method to better solve discrete optimization problems, it still cannot guarantee that the resulting positions are integers, as is required for solutions to the Flight Schedule Problem optimization problem. Therefore, this section improves the experience selection strategy (ES), the block search strategy (BSS), and the algorithm's perturbation term, thereby transforming the

MERPSO algorithm into the discrete variant, the MERSPO algorithm, tailored for solving the Flight Schedule Problem optimization problem.

### (1) Experience Selection Strategy

In the discrete experience selection strategy, the storage principle and method of the Experience Set (ES) remain unchanged. The ES maintains an archive of the  $N_p$  best-performing solutions in terms of fitness throughout the entire search history, which is updated after each iteration's fitness evaluation. Each dimension of the learning sample  $sg$  is still selected independently; however, instead of calculating the average value of the selected solutions during the selection process,  $K$  solutions are randomly selected with replacement from the ES, and the most frequently occurring value among these selected solutions is chosen as the corresponding dimensional value for  $sg$ . In cases where multiple values have equal frequencies, one of these values is randomly selected as the dimensional value for  $sg$ . Here,  $K = 10$ , and  $N_p$  is calculated using Equation (33):

$$N_p = \left\lceil (ps - 5) \times \left(1 - \frac{t - 1}{T}\right) + 5 \right\rceil \quad (33)$$

### (2) Block Search Strategy

In the discrete block search strategy, the configuration of blocks varies according to particle dimensions. For the first to  $N_f$ -th dimensions of particles (i.e., flight adjustment time), blocks still maintain boundaries, where all integers on these boundaries can serve as selection ranges for  $mp$ . Taking Figure 2 as an example, which represents a block with a lower boundary of 1 and an upper boundary of 3, when selecting  $pr_d\{b\}$  according to Equation (15),  $mp$  can take any integer value within  $\{1,2,3\}$ . The splitting and merging operations continue to be applied to blocks in the first to  $N_f$ -th dimensions. An additional operation is introduced: when the difference between block boundaries is even and splitting becomes necessary, a new block will be formed by combining the original lower boundary with the integer value of  $(\text{upper boundary} + \text{lower boundary})/2$ , as illustrated in Figure 2. However, for the  $(N_f + 1)$ -th to  $2N_f$ -th dimensions of particles (i.e., route selection components), there exists no continuous practical significance between adjacent integers. Consequently, blocks in this dimension range are defined without boundaries—for instance, block  $\{5,8\}$  indicates that eight particles have selected route number 5 in this dimension. Therefore, neither merging nor splitting operations are performed on blocks within this dimensional range.

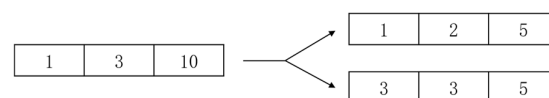


**Figure 2.** Discrete block splitting operation.

With the increase in particle dimensions, higher demands are placed on memory storage. To conserve memory and reduce the computational time, this section stores the data blocks of dimensions 1 to  $N_f$  and dimensions  $N_f + 1$  to  $2N_f$  separately in two matrices, BL1 and BL2. Two additional matrices are employed to record the starting positions of each dimension, which are referred to as boundary matrices (Edge Matrices) EM1 and EM2. The correspondence between BL and EM is illustrated in Figure 3.

| BL1 |     |     |     | EM1 |     | BL2 |     |     | EM2 |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1   | 2   | 4   | 10  | 1   | 1   | 1   | 1   | 4   | 1   | 1   |
| 2   | 4   | 7   | 5   | 2   | 5   | 2   | 2   | 7   | 2   | 8   |
| ... | ... | ... | ... | 3   | 10  | ... | ... | ... | 3   | 16  |
| 5   | -1  | 3   | 4   | ... | ... | 8   | 1   | 3   | ... | ... |
| ... | ... | ... | ... |     |     | ... | ... | ... |     |     |
| 10  | 4   | 5   | 2   |     |     | 16  | 1   | 5   |     |     |
| ... | ... | ... | ... |     |     | ... | ... | ... |     |     |

Figure 3. Correspondence between BL and EM.

### (3) Perturbation Term

The perturbation operation performs single-point mutation on the original particle basis, with a mutation probability of  $P^{dt}$  for each dimension. The parameter  $P^{dt}$  linearly decreases from 0.6 to 0.2 as the number of generations increases. The variation range for the first  $Nf$  dimensions is  $[R_{min}^{dt}, R_{max}^{dt}]$ , while the variation range for the last  $Nf$  dimensions corresponds to any selectable flight path, as formulated in Equations (33)–(36):

$$space_t^{dt} = \frac{(T_{max}^a - T_{min}^a)}{2} \times \left(1 - 0.8 \times \frac{t}{T}\right) \quad (34)$$

$$R_{min}^{dt} = \max\left(T_{min}^a, x_{i,d} - space_t^{dt}\right) \quad (35)$$

$$R_{max}^{dt} = \min\left(T_{max}^a, x_{i,d} + space_t^{dt}\right) \quad (36)$$

#### 3.2.4. The Methods for Initializing Particles and Satisfying Constraints

The adjustment time is categorized into five levels: slightly early, almost no adjustment, slightly delayed, moderately delayed, and severely delayed, defined as  $t^a = (-6, 0]$ ,  $(0, 3]$ ,  $(3, 8]$ ,  $(8, 18]$ ,  $(18, 36]$ . During population initialization, dimensions 1 to  $Nf$  of each particle randomly select one of these five intervals to generate adjustment times, while route selection dimensions are chosen randomly. This process forms the initial population  $p^0$ . After evaluating  $p^0$ , the top 1/3 particles with the best fitness performance are retained. These preserved particles undergo single-point mutation with probability  $P^{p0}$ , where each particle performs three mutation operations. All mutated particles constitute the new population  $p^1$ , which serves as the initialized population for the algorithm. This initialization approach ensures sufficient uniformity in particle distribution across the search space, enhances population diversity, and facilitates global exploration. Simultaneously, eliminating the worst-performing particles avoids excessive computational resource consumption on inferior solutions, thereby improving convergence precision.

During the optimization process, in order to ensure that the position of each particle in the population is always within the feasible region that satisfies the constraint limits (such as the capacity limits for runway takeoff and landing, and the minimum takeoff interval limit), the particle solution after each position update is always compared with the constraint conditions. If it does not meet the constraint condition limits, a random method is used to generate a new position, thereby repairing the particle solution to ensure that the search solution satisfies the operational constraints.

## 4. Simulation Experiment and Verification

### 4.1. Flight Data Collection and Processing

The dataset used in the experiments consists of real flight plans provided by an aviation data communication company. The flights originate from eight flight schedule instances from different operational periods across nearly 200 airports in China, ensuring

authenticity and generalizability [24]. Each dataset contains the following information: flight number, coordinates of departure airport, coordinates of arrival airport, scheduled departure time ( $\tau^{orig}$ ), scheduled arrival time ( $\tau^{des}$ ), route (including waypoints), actual trajectory, and average speed during actual flight. To simulate temporary adjustments to flight plans before execution and introduce additional scheduling difficulty, the departure and arrival times of certain flights were modified. The characteristics of these flight datasets are summarized in Table 3.

**Table 3.** Initial features of the experimental dataset.

| Dataset | Number of Flights | Number of Airports | Number of Waypoints | Number of Routes | Initial Delay (min) | Adjusted Initial Delay (min) |
|---------|-------------------|--------------------|---------------------|------------------|---------------------|------------------------------|
| M1      | 937               | 181                | 826                 | 2053             | 10,812              | -                            |
| M2      | 925               | 171                | 794                 | 2113             | 10,752              | -                            |
| M3      | 952               | 181                | 845                 | 2331             | 13,734              | 15,297                       |
| M4      | 932               | 164                | 790                 | 2374             | 18,392              | 20,752                       |
| M5      | 953               | 182                | 817                 | 2076             | 26,596              | 28,361                       |
| M6      | 920               | 171                | 792                 | 2036             | 48,905              | 51,802                       |
| M7      | 810               | 172                | 863                 | 1983             | 20,744              | 23,850                       |
| M8      | 809               | 164                | 854                 | 2130             | 26,744              | 29,501                       |

Among them, the initial delay refers to the sum of delay times for all flights when the flight schedule is actually executed, while the adjusted initial delay represents the sum of delay times under increased optimization difficulty during actual execution. Datasets M1 and M2 represent original flight schedules; M3 and M4 simulate mild delay scenarios by randomly selecting 10% of flights in the original schedules and simultaneously advancing or delaying both departure and arrival times by identical time intervals (ranging from 30 min early to 30 min late); M5 and M6 simulate moderate delay scenarios by randomly selecting 3% of flights in the original schedules with adjusted time windows (either advanced by 30–60 min or delayed by 30–60 min); M7 and M8 first implement mild delay simulations on 10% of randomly selected flights as in M3/M4, then additionally apply moderate delay adjustments to 3% of remaining unadjusted flights (representing 3% of total flights) with modified time windows (advanced by 30–60 min or delayed by 30–60 min).

#### 4.2. Experimental Performance

To validate the effectiveness of the DMERPSO algorithm, comparative experiments were conducted with RPSO [19], ADFPSO [25], SpadePSO [26], L-SHADE [27] and JSO [28] algorithms. For several continuous algorithms, in order to discretize them into integer encoding, the real-valued solutions of these algorithms underwent decimal rounding processing techniques to obtain integers. If their integer was greater than or less than the maximum and minimum values of each dimension integer within the given range of Flight Schedule Problem optimization problem, then the maximum and minimum values were taken, which ensured that the solutions of the comparison algorithms fell within the integer solution space of the Flight Schedule Problem optimization problem.

For each comparison algorithm, parameter settings followed the recommended values specified in their respective original papers. Each algorithm was independently executed 30 times on every dataset. To ensure fair comparison, all algorithms shared identical initial populations and employed the same stopping criterion based on maximum fitness evaluations (*MaxFEs*). The population size was set to 30 and *MaxFEs* was configured as 30,000 for all compared algorithms. All experiments were performed on a computer running the 64-bit version of Windows 11 Pro, with hardware specifications including an Intel Core i5 @3 GHz CPU and 32 GB RAM. All algorithms were implemented in MATLAB R2023a.

The optimization results of the DMERPSO algorithm and these comparative algorithms are presented in Table 4. The data in the table represent the mean and variance of total delays (in minutes) across 30 independent experiments. The symbols (+), (−), and ( $\approx$ ) indicate that the corresponding algorithm is significantly better, significantly worse, or approximately equivalent to the DMERPSO algorithm in terms of performance based on *t*-test results with  $p \leq 0.05$ . The minimum mean total delay for each dataset is highlighted in bold. Experimental results demonstrate that the DMERPSO algorithm can significantly reduce total flight delays and optimize flight schedules. It achieves superior performance over other comparative algorithms across all eight datasets, particularly showing the best performance on the more challenging M7 and M8 datasets.

**Table 4.** Total delay optimization results of the DMERPSO algorithm and the comparative algorithms (The bold data represents the best result).

| Dataset | Total Delay | DMERPSO                              | RPSO                   | ADFPSO                 | SpadePSO               | L-SHADE                | JSO                    |
|---------|-------------|--------------------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| M1      | mean        | <b><math>1.07 \times 10^3</math></b> | $7.52 \times 10^4$ (+) | $2.31 \times 10^4$ (+) | $8.88 \times 10^4$ (+) | $1.59 \times 10^4$ (+) | $9.43 \times 10^3$ (+) |
|         | std.        | 1029.6                               | 2761                   | 1847.3                 | 1331.8                 | 10360.0                | 3859.4                 |
| M2      | mean        | <b><math>2.15 \times 10^3</math></b> | $7.71 \times 10^4$ (+) | $2.27 \times 10^4$ (+) | $8.63 \times 10^4$ (+) | $1.44 \times 10^4$ (+) | $1.05 \times 10^4$ (+) |
|         | std.        | 1104.8                               | 2721.2                 | 2586                   | 1387.1                 | 4938.7                 | 1977.0                 |
| M3      | mean        | <b><math>3.16 \times 10^3</math></b> | $7.83 \times 10^4$ (+) | $2.20 \times 10^4$ (+) | $9.22 \times 10^4$ (+) | $2.03 \times 10^4$ (+) | $9.80 \times 10^3$ (+) |
|         | std.        | 745.27                               | 3820.6                 | 2271.8                 | 888.88                 | 13375.1                | 3325.8                 |
| M4      | mean        | <b><math>4.23 \times 10^3</math></b> | $8.29 \times 10^4$ (+) | $2.56 \times 10^4$ (+) | $9.02 \times 10^4$ (+) | $1.36 \times 10^4$ (+) | $1.45 \times 10^4$ (+) |
|         | std.        | 1097.3                               | 3818.9                 | 2632.3                 | 1560.7                 | 8303.3                 | 3917.3                 |
| M5      | mean        | <b><math>6.13 \times 10^3</math></b> | $8.05 \times 10^4$ (+) | $2.91 \times 10^4$ (+) | $9.73 \times 10^4$ (+) | $2.53 \times 10^4$ (+) | $1.34 \times 10^4$ (+) |
|         | std.        | 1061.1                               | 2250.6                 | 2107.7                 | 1611.6                 | 12115.3                | 3013.3                 |
| M6      | mean        | <b><math>2.16 \times 10^3</math></b> | $7.36 \times 10^4$ (+) | $2.50 \times 10^4$ (+) | $9.14 \times 10^4$ (+) | $1.88 \times 10^4$ (+) | $1.10 \times 10^4$ (+) |
|         | std.        | 1499                                 | 3405.5                 | 1627                   | 1522.4                 | 12011.1                | 3525.0                 |
| M7      | mean        | <b><math>3.27 \times 10^3</math></b> | $5.64 \times 10^4$ (+) | $2.29 \times 10^4$ (+) | $7.83 \times 10^4$ (+) | $1.85 \times 10^4$ (+) | $1.31 \times 10^4$ (+) |
|         | std.        | 1119.5                               | 2496.8                 | 2788.4                 | 1562.5                 | 9224.8                 | 3852.6                 |
| M8      | mean        | <b><math>1.89 \times 10^3</math></b> | $6.02 \times 10^4$ (+) | $1.65 \times 10^4$ (+) | $7.83 \times 10^4$ (+) | $8.21 \times 10^3$ (+) | $8.25 \times 10^3$ (+) |
|         | std.        | 667.87                               | 3513                   | 2584.3                 | 1257.8                 | 4020.4                 | 2882.0                 |

To better demonstrate the advantages of the proposed algorithm, the Friedman test was first applied to the mean values of the Total Delay Optimization Results from Table 4, followed by the Wilcoxon signed-rank test to compare the performance of the RMERSPO algorithm with other algorithms. Table 5 presents the rankings of algorithms based on the mean Total Delay. The last row of the table shows the Friedman test results; the *p*-value is very small, indicating a significant difference in algorithm performance.

The Wilcoxon signed-rank test was used to compare two algorithms in all pairwise comparisons. The DMERSPSO algorithm was compared with the other three algorithms using the Wilcoxon signed-rank test at a significance level of  $\alpha = 0.05$ . The summarized results are presented in Table 6, with the signs +,  $\approx$ , and −. The sign ‘+’ means that DMERSPSO performs significantly better than the compared algorithm; ‘−’ means that DMERSPSO performs significantly worse; while ‘ $\approx$ ’ means that there is no significant difference between the two algorithms. From the results in the table, it can be seen that DMERSPSO performs overwhelmingly better than the other algorithms.

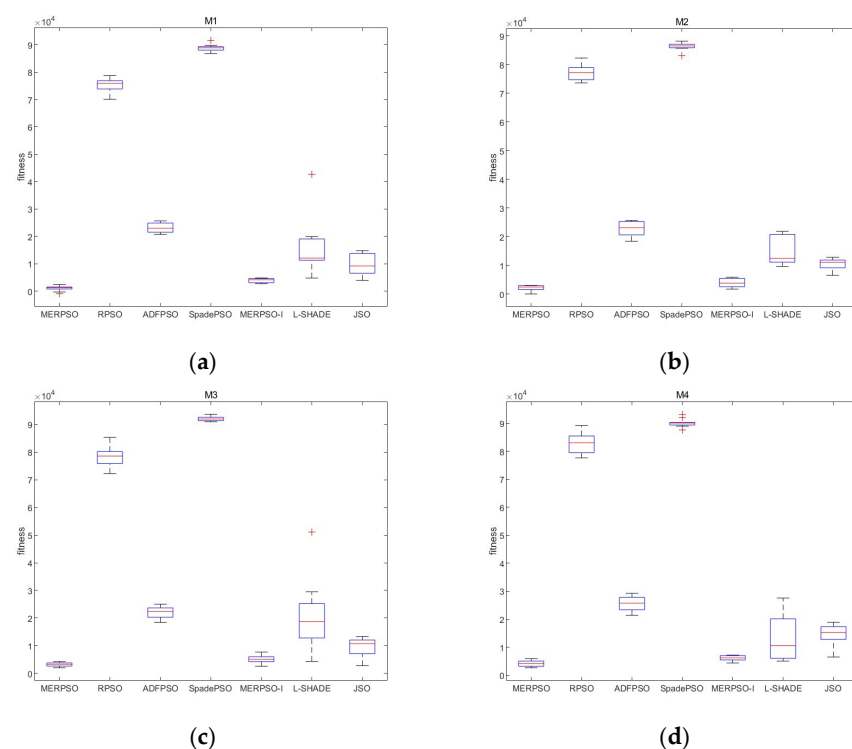
**Table 5.** The rank of algorithms based on the Total Delay Mean of dataset and Friedman test.

| Dataset       | DMERPSO                                  | RPSO | ADFPSO | SpadePSO | L-SHADE | JSO  |
|---------------|------------------------------------------|------|--------|----------|---------|------|
| M1            | 1                                        | 6    | 5      | 7        | 4       | 3    |
| M2            | 1                                        | 6    | 5      | 7        | 4       | 3    |
| M3            | 1                                        | 6    | 5      | 7        | 4       | 3    |
| M4            | 1                                        | 6    | 5      | 7        | 3       | 4    |
| M5            | 1                                        | 6    | 5      | 7        | 4       | 3    |
| M6            | 1                                        | 6    | 5      | 7        | 4       | 3    |
| M7            | 1                                        | 6    | 5      | 7        | 4       | 3    |
| M8            | 1                                        | 6    | 5      | 7        | 3       | 4    |
| Mean rank     | 1                                        | 6    | 5      | 7        | 3.75    | 3.25 |
| Friedman Test | $\chi^2 = 24, p = 1.5880 \times 10^{-8}$ |      |        |          |         |      |

**Table 6.** The Wilcoxon signed-rank results of DMERPSO versus other algorithms ( $\alpha = 0.05$ ).

| Algorithm | +/ $\approx$ /− |
|-----------|-----------------|
| RPSO      | 8/0/0           |
| ADFPSO    | 8/0/0           |
| SpadePSO  | 8/0/0           |
| L-SHADE   | 8/0/0           |
| JSO       | 8/0/0           |

To verify the stability of the algorithm's operation, box plots were generated to analyze the total delay obtained from 30 independent runs on eight datasets, with the results shown in Figure 4. The experimental results demonstrate that the DMERPSO algorithm achieves superior optimization performance and exhibits higher stability.

**Figure 4.** Cont.

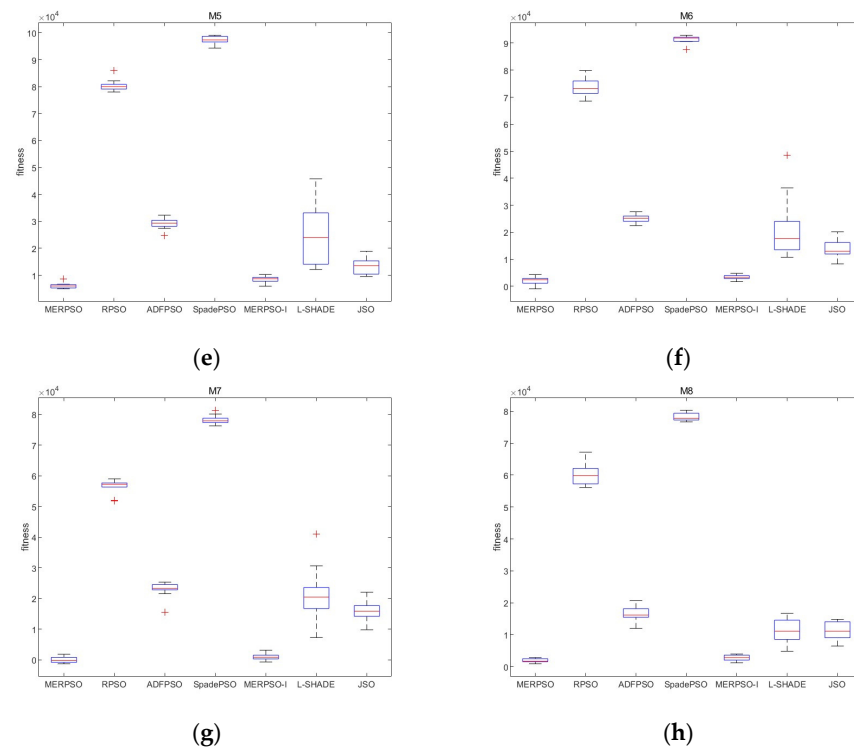


**Figure 4.** Box plots of the results on the eight datasets. (a) M1; (b) M2; (c) M3; (d) M4; (e) M5; (f) M6; (g) M7; (h) M8.

#### 4.3. Effectiveness Analysis of the Initialization Method

The DMERPSO algorithm introduces a population initialization method that incurs the computational cost of one generation. To analyze the effectiveness of this strategy, convergence curves were plotted by comparing the average best fitness per generation across 30 independent experiments between the DMERPSO algorithm, the DMERPSO-I variant (without the initialization strategy), and three baseline algorithms. The convergence curves of these algorithms on eight datasets are presented in Figure 5. As observed from the convergence results, the DMERPSO algorithm demonstrates strong sustained convergence capability. In the middle to late iteration stages, when other algorithms become trapped in local optima, DMERPSO exhibits superior ability to escape local optima through continuous exploration and exploitation of the solution space to seek better solutions. Although the DMERPSO-I variant without the initialization strategy achieves better performance than DMERPSO in early iterations, the population diversity introduced by the initialization strategy enables DMERPSO to achieve better convergence accuracy in later stages. This validates the effectiveness of the proposed initialization strategy.

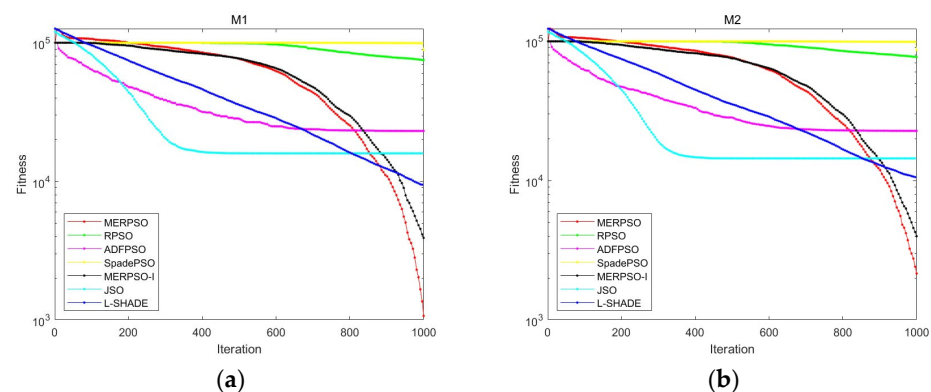


**Figure 5.** Cont.

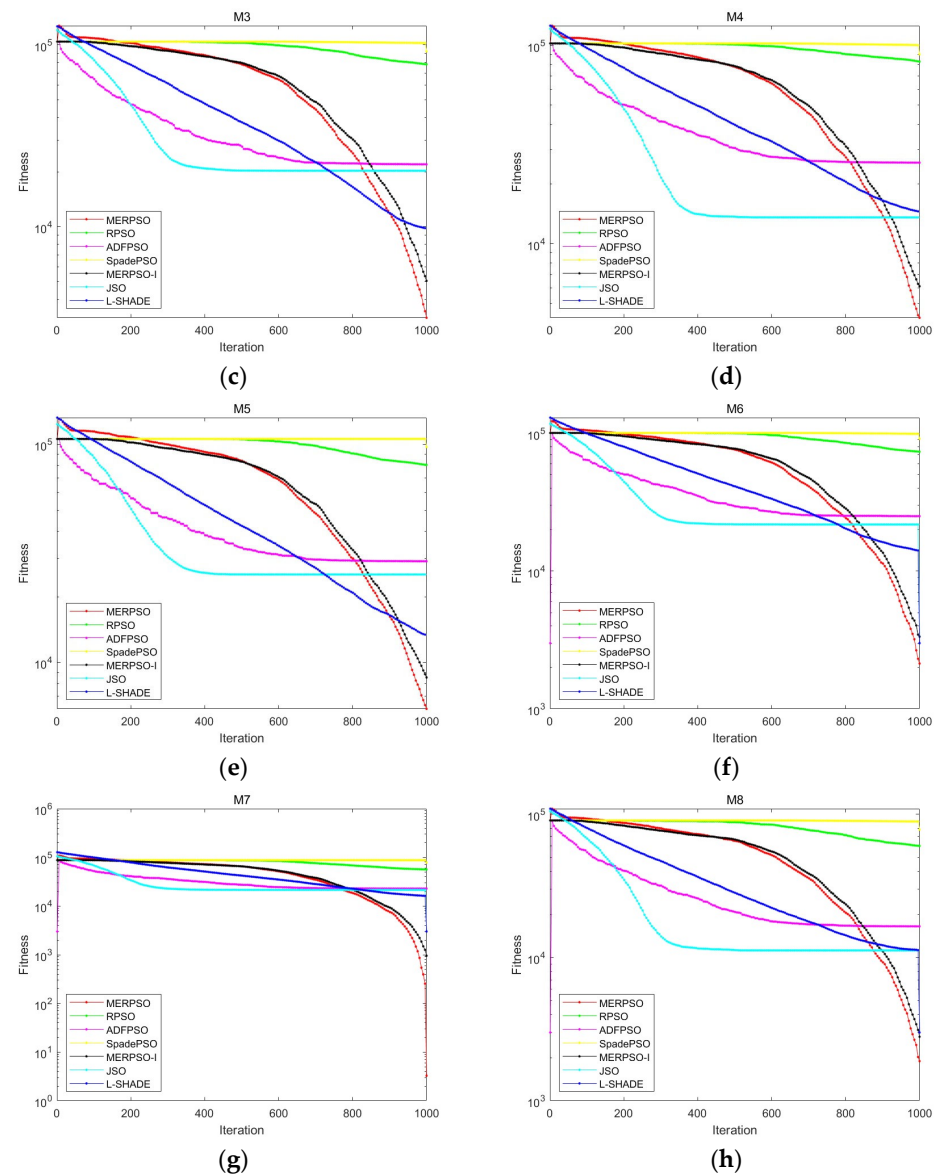


**Figure 5.** Convergence curves of the DMERPSO algorithm and the comparison algorithms. (a) Curves of convergence on M1; (b) convergence curves on M2; (c) curves of convergence on M3; (d) curves of convergence on M4; (e) curves of convergence on M5; (f) curves of convergence on M6; (g) curves of convergence on M7; (h) curves of convergence on M8.

## 5. Conclusions

To effectively address the discrete optimization problem of single-objective Flight Schedule Problem optimization, a Discrete Memory-Enhanced Restructured Particle Swarm Optimization algorithm (DMERPSO) is proposed. Experimental results demonstrate that the DMERPSO algorithm can effectively reduce flight delays, optimize air traffic capacity allocation, and achieve superior performance compared to benchmark algorithms.

The core innovation of the DMERPSO algorithm is designing the calculation of the selection probabilities from three components. More importantly, the DMERPSO algorithm improves the three terms—the experience selection strategy (ES), the block search strategy (BSS), and the algorithm’s perturbation term—to directly transform them so as to be tailored to solve the integer code problem of the Flight Schedule Problem. The proposed integer coding scheme for Flight Schedule Optimization is significant and can be extended to other related problems. In the experiments conducted in this study, continuous algorithms were employed as comparisons. These algorithms performed decimal rounding on real-number

solutions to obtain integers, which makes it difficult to maintain the favorable optimization characteristics of the solution space search, thereby introducing certain discrepancies in the experimental comparison.

**Author Contributions:** Conceptualization, J.L., W.G. and B.W.; methodology, B.W., W.G. and J.L.; software, B.W. and D.T.; validation, W.G., D.T., J.L. and B.W.; formal analysis, W.G., B.W., D.T. and J.L.; investigation, W.G., B.W., D.T. and J.L.; resources, J.L. and W.G.; data curation, B.W., D.T., J.L. and W.G.; writing—original draft preparation, B.W. and W.G.; writing—review and editing, W.G., J.L. and B.W.; visualization, B.W. and D.T.; supervision, J.L.; project administration, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Provincial Natural Science Foundation of Fujian, China (2023J01349).

**Data Availability Statement:** Data is contained in <https://github.com/buaaguotong/ATFM-Benchmark> (accessed on 10 March 2026).

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

## References

- Bertsimas, D.; Patterson, S.S. The Air Traffic Flow Management Problem with Enroute Capacities. *Oper. Res.* **1998**, *46*, 406–422. [CrossRef]
- Hu, H.; Sun, J.; Du, B. Air Traffic Management in Dense Airspace via Network Flow Optimization. *J. Aerosp. Inf. Syst.* **2025**, *22*, 433–446. [CrossRef]
- Eltoukhy, A.E.E.; Chan, F.T.S.; Chung, S.H. Airline schedule planning: A review and future directions. *Ind. Manag. Data Syst.* **2017**, *117*, 1201–1243. [CrossRef]
- Wang, D.; Wu, Y.; Hu, J.Q.; Liu, M.; Yu, P.; Zhang, C.; Wu, Y. Flight schedule recovery: A simulation-based approach. *Asia-Pac. J. Oper. Res.* **2019**, *36*, 1940010. [CrossRef]
- Yan, S.; Tu, Y.P. Multifleet routing and multistop flight scheduling for schedule perturbation. *Eur. J. Oper. Res.* **1997**, *103*, 155–169. [CrossRef]
- Soykan, B.; Erol, S. A branch-and-price algorithm for the robust airline crew pairing problem. *Savun. Bilim. Derg.* **2014**, *13*, 37–74.
- Barnhart, C.; Belobaba, P.; Odoni, A.R. Applications of operations research in the air transport industry. *Transp. Sci.* **2003**, *37*, 368–391. [CrossRef]
- Mercier, A.; Cordeau, J.F.; Soumis, F. A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Comput. Oper. Res.* **2005**, *32*, 1451–1476. [CrossRef]
- Reiners, T.; Pahl, J.; Maroszek, M.; Rettig, C. Integrated aircraft scheduling problem: An auto-adapting algorithm to find robust aircraft assignments for large flight plans. In *Proceedings of the 2012 45th Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2012*; IEEE: Piscataway, NJ, USA, 2012; pp. 1267–1276.
- Cadarso, L.; Vaze, V. Passenger-centric integrated airline schedule and aircraft recovery. *Transp. Sci.* **2023**, *57*, 813–837. [CrossRef]
- Delahaye, D.; Odoni, A.R. Airspace Congestion Smoothing by Stochastic Optimization. In *Evolutionary Programming VI, Proceedings of the 6th International Conference*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 163–176.
- Xiao, M.; Hong, C.; Cai, K. Optimizing the safety-efficiency trade-off on nationwide air traffic network flow using cooperative co-evolutionary paradigm. *Sci. Rep.* **2025**, *15*, 20377. [CrossRef] [PubMed]
- Cecen, R.K.; Durmazkeser, Y. Meta-Heuristic Algorithms for Aircraft Sequencing and Scheduling Problem. In *Progress in Sustainable Aviation*; Springer International Publishing: Cham, Switzerland, 2022; pp. 107–118.
- Kablan, A.A.; Elberkawi, H.R.K.; Eldharif, E.A. Airline scheduling model using genetic algorithm. In *Proceedings of the 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Tripoli, Libya, 6–8 May 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 718–722.
- Deng, G.F.; Lin, W.T. Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Syst. Appl.* **2011**, *38*, 5787–5793. [CrossRef]
- Li, M.; Hao, J.K.; Wu, Q. Learning-driven feasible and infeasible tabu search for airport gate assignment. *Eur. J. Oper. Res.* **2022**, *302*, 172–186. [CrossRef]
- Abualigah, L.; Sheikhan, A.; Ikotun, A.M.; Zitar, R.A.; Alsoud, A.R.; Alshourbaji, I.; Hussien, G.; Jia, H. Particle Swarm Optimization Algorithm: Review and Applications. *Metaheuristic Optim. Algorithms* **2024**, 1–14.
- Ma, J.; Zhang, X.; Wang, C. Dynamic Optimization of Flight Schedules Based on Deep Reinforcement Learning. *Comput. Integr. Manuf. Syst.* **2021**, *27*, 2697–2708.

19. Zhu, J.; Liu, J.; Wang, Z.; Chen, Y. Restructuring Particle Swarm Optimization Algorithm Based on Linear System Theory. In *Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 1–7.
20. Zhu, J.; Liu, J.; Chen, Y.; Xue, X.; Sun, S. Binary Restructuring Particle Swarm Optimization and Its Application. *Biomimetics* **2023**, *8*, 266. [[CrossRef](#)] [[PubMed](#)]
21. Wu, B.; Liu, J.; Li, S.; Li, M. Memory-Enhanced Reconstructive Particle Swarm Optimization Algorithm. *Comput. Eng. Appl.* **2025**, *61*, 116–127.
22. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In *Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995*; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 1942–1948.
23. Van den Bergh, F.; Engelbrecht, A.B. A Study of Particle Swarm Optimization Particle Trajectories. *Inf. Sci.* **2006**, *176*, 937–971. [[CrossRef](#)]
24. Guo, T.; Mei, Y.; Tang, K.; Du, W. A Knee-guided Evolutionary Algorithm for Multi-objective Air Traffic Flow Management. *IEEE Trans. Evol. Comput.* **2023**, *28*, 994–1008. [[CrossRef](#)]
25. Yu, F.; Tong, L.; Xia, X. Adjustable Driving Force Based Particle Swarm Optimization Algorithm. *Inf. Sci.* **2022**, *609*, 60–78. [[CrossRef](#)]
26. Wu, X.; Han, J.; Wang, D.; Gao, P.; Cui, Q.; Chen, L.; Liang, Y.; Huang, H.; Lee, H.P.; Miao, C.; et al. Incorporating Surprisingly Popular Slgorithm and Euclidean Distance-based Adaptive Topology into PSO. *Swarm Evol. Comput.* **2023**, *76*, 101222. [[CrossRef](#)]
27. Tanabe, R.; Fukunaga, A. Improving the search performance of SHADE using linear population size reduction. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014*; IEEE: Piscataway, NJ, USA, 2014; pp. 1658–1665.
28. Brest, J.; Maučec, M.S.; Bošković, B. Single objective real-parameter optimization: Algorithm jSO. In *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1311–1318.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.