electronics

MDPI

Article

A Hybrid Periodic and Event-Driven Rolling Horizon Optimization Approach for Airport Logistics Vehicle Scheduling

Ran Feng [1], Zhihao Cai [1], Boyuan Li [1] and Qian-Qian Zheng [2,*]

[1] School of Computer Science and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China
[2] School of Management, Zhengzhou University, Zhengzhou 450001, China
* Correspondence: qianqianzheng@zzu.edu.cn

Abstract

The efficient scheduling of airport logistics vehicles is crucial for ensuring timely and cost-effective ground operations, particularly under dynamic disturbances such as flight delays, cancellations, and new task arrivals. With the increasing deployment of Internet of Things (IoT) technologies in airport environments, real-time data from sensors and connected devices enables efficient and adaptive scheduling. This paper considers a dynamic Airport Logistics Vehicle Scheduling (ALVS) problem that aims to minimize both vehicle usage and total task waiting time while satisfying task precedence and time window constraints. To address this problem, we propose a hybrid optimization framework, termed Periodic and Event-Driven Rolling Horizon Optimization (PERHO), which integrates periodic updates with event-driven rescheduling to adapt to real-time task variations in airport ground operations. Within PERHO, an Order-aware Adaptive Strategy Selection (OASS) algorithm is developed to dynamically select the most appropriate task sequencing heuristic from a candidate set based on recent performance and order relationships. Extensive experiments across various instance scales and dynamic scenarios demonstrate the effectiveness of the proposed PERHO-OASS approach. In experiments considering dynamic events, PERHO-OASS reduces vehicle usage and task waiting time by an average of 23.55% and 61.95%, respectively, over fixed heuristic algorithms, and by an average of 3.77% and 17.30% over adaptive selection methods, demonstrating strong robustness under uncertainty. The proposed approach can support airport operators in improving the efficiency and reliability of ground logistics operations.

Keywords: airport logistics vehicle scheduling; rolling horizon optimization; dynamic scheduling; adaptive system

Check for updates

Academic Editor: Subir Halder

Received: 27 March 2026
Revised: 30 April 2026
Accepted: 11 May 2026
Published: 18 May 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) license.

1. Introduction

With the rapid growth of the air transportation industry, airports face increasing pressure to manage large-scale logistics operations. Logistics vehicles play a critical role in airport ground logistics services, transporting baggage and cargo between terminals and aircraft. To ensure smooth airside operations and maintain flight punctuality, it is essential to develop efficient and adaptive scheduling methods for these logistics vehicles.

The Airport Logistics Vehicle Scheduling (ALVS) problem involves assigning a limited number of vehicles to time-constrained logistics tasks associated with flights. Each flight typically requires multiple logistics support tasks subject to strict temporal and precedence constraints. In addition, task execution is affected by uncertainties such as stochastic flight arrivals and cancellations, which further increase the complexity of scheduling decisions.

Existing research has investigated the ALVS problem using approaches such as mixed-integer programming and heuristic or meta-heuristic algorithms [1–3]. Some studies have also addressed scheduling for other airport ground support vehicles, such as refueling [4,5] and towing vehicles [6,7]. However, most of these methods are designed for static or semi-dynamic settings and often struggle to cope with frequent real-time disturbances.

With the increasing deployment of Internet of Things (IoT) technologies [8,9], real-time data from sensors and connected devices facilitates communication and information sharing. These data are transformed into structured information, including events such as task arrivals and cancellations, and system states such as vehicle locations and availability. These serve as inputs to the scheduling model and enable adaptive decision-making in response to dynamically changing airport conditions. Leveraging this capability, recent studies have applied Rolling Horizon Optimization (RHO) techniques, which iteratively solve a sequence of subproblems over a moving time horizon and update decisions as new information becomes available, thereby allowing rescheduling when new tasks or disruptions occur [10–12]. Nevertheless, key challenges remain, including balancing computational efficiency with optimization quality within each rolling interval and dynamically selecting or switching scheduling strategies in response to changes in task conditions.

Therefore, this study explores how airport logistics vehicles can be scheduled under dynamic disruptions while minimizing vehicle usage and task waiting time. Specifically, we first formulate a bi-objective ALVS optimization model that captures the operational workflow and practical requirements of airport logistics. The model accounts for constraints such as time windows and precedence relationships among tasks. We then propose a hybrid Periodic and Event-Driven Rolling Horizon Optimization (PERHO) framework, which integrates periodic updates with event-driven triggers to enable adaptive scheduling under dynamic disruptions. Within this framework, an Order-aware Adaptive Strategy Selection (OASS) algorithm is developed to dynamically select among three heuristic strategies, namely Earliest Serviceable Time Priority (ESTP), Arrival Time Priority (ATP), and Random Serviceable Time Priority (RSTP), based on recent performance and sequencing effectiveness. By combining adaptive strategy selection with rolling horizon decomposition, the proposed approach enhances decision flexibility under both static and dynamic conditions. Extensive experiments are conducted using instances derived from the benchmark in [13], where PERHO-OASS is compared with fixed heuristic algorithms (ESTP, ATP, and RSTP) as well as adaptive selection methods, including Adaptive Operator Selection (AOS) and Multi-Armed Bandit (MAB), to evaluate its effectiveness in solving the ALVS problem.

The main contributions of this paper are summarized as follows:

1. A bi-objective ALVS optimization model is formulated to minimize the number of vehicles used and the total waiting time of all tasks.
2. A hybrid PERHO framework is developed to integrate periodic scheduling with event-driven re-optimization for adaptive disruption handling.
3. An OASS algorithm is proposed to dynamically select task-ordering heuristics based on recent performance, thereby improving solution quality.
4. Extensive experiments under both static and dynamic scenarios demonstrate that the proposed PERHO-OASS approach outperforms baseline methods for the ALVS problem.

The remainder of this paper is organized as follows. Section 2 reviews research on airport ground vehicle scheduling and the application of rolling horizon optimization. Section 3 formalizes the ALVS problem. Section 4 presents the proposed PERHO framework and details the implementation of the OASS algorithm. Section 5 reports and analyzes experimental results under various parameter settings and dynamic scenarios. Finally, Section 6 concludes the study and outlines directions for future research.

2. Related Work

This section reviews the relevant literature on airport ground support vehicle scheduling problems and the RHO method, highlighting the key characteristics of airport ground vehicle scheduling and the advantages of RHO in addressing dynamic scheduling problems.

2.1. Airport Ground Support Vehicle Scheduling

Airport ground vehicle scheduling problems have attracted increasing attention in recent years. Generally, based on the number of service types, these problems can be classified into single-type and multi-type vehicle scheduling. Single-type vehicle scheduling focuses on the scheduling of a single category of vehicles, such as ferry vehicles [14], baggage vehicles [1,15], and other airport ground service vehicles [13,16], whereas multi-type vehicle scheduling involves the coordination of multiple types of vehicles with different operational characteristics [17–19]. Most of these studies consider static scheduling settings without explicitly accounting for operational disturbances or uncertainty. Several studies incorporate stochastic factors such as flight delays and emergency events. For example, Han et al. [20] propose a two-stage optimization algorithm for ferry vehicles considering flight time uncertainty, while Zhu et al. [21] develop a multi-objective mixed-integer programming model for ground-handling vehicle schedules under uncertain flight arrivals. These approaches extend static models by incorporating uncertainty, but they still rely on pre-planned scheduling structures and lack explicit mechanisms for adaptive decision-making in dynamically evolving environments.

In general, airport ground support vehicle scheduling can be formulated as a complex single-objective or multi-objective optimization problem. In practical settings, dynamic task arrivals, uncertainty, and strict operational constraints substantially increase both modeling complexity and computational difficulty. However, most existing studies rely on static or pre-planned scheduling frameworks, where uncertainty is primarily handled through stochastic modeling rather than adaptive decision-making, thereby limiting their ability to respond effectively to evolving operational conditions. These limitations highlight the need for adaptive scheduling mechanisms capable of maintaining high solution quality under dynamic conditions.

2.2. Rolling Horizon Optimization

RHO has been extensively applied across multiple domains to address dynamic decision-making problems under uncertainty. In the energy domain, RHO has been widely applied to scheduling and planning problems [22–26]. For instance, Servare et al. [24] developed a rolling horizon framework for large-scale industrial energy scheduling, while Brusco et al. [25] proposed a control model for renewable energy communities that leverages distributed storage to mitigate power imbalance. In production and manufacturing, RHO has been employed for dynamic production planning, scheduling, and inventory management under uncertainty [27–31]. Boujnah et al. [30] integrate a rolling horizon mechanism into aggregate production planning to enhance responsiveness, and Cruz et al. [31] apply a rolling horizon approach to perishable product systems, achieving improved service levels and reduced waste. In logistics and transportation, RHO facilitates dynamic vehicle routing and adaptive scheduling of transportation resources [10,32–35]. Abbaszadeh et al. [35] propose a rolling-horizon-based method for drone-assisted mobile blood collection, coordinating heterogeneous resources efficiently, and Zheng et al. [33] develop a rolling horizon strategy for port operations to handle dynamically arriving inter-terminal transport requests. These studies demonstrate the versatility of RHO in supporting adaptive decision-making in highly dynamic environments.

Despite its demonstrated effectiveness across multiple domains, applying RHO to airport ground logistics remains challenging due to the distinctive operational characteristics of airport environments. In particular, tight task deadlines and frequent dynamic events, such as flight delays and cancellations, make dynamic scheduling a critical requirement. In this context, RHO provides a suitable framework for handling dynamically evolving scheduling problems, as it enables iterative updates of decisions in response to newly available information. However, its application to airport ground logistics remains limited. This gap highlights the need for dedicated RHO-based scheduling frameworks that explicitly capture the dynamic and constrained nature of airport operations, thereby supporting more adaptive and efficient scheduling decisions.

Overall, existing studies are insufficient to fully address dynamic airport ground logistics scheduling, motivating the development of dedicated RHO-based approaches for airport operations that enable adaptive and efficient scheduling under dynamic conditions.

3. Problem Formulation

In the ALVS problem, each logistics support task $\tau \in \mathcal{T}$ is associated with a flight $f \in \mathcal{F}$, which arrives at time a_f. Executing a logistics support task (hereafter referred to as a task) can be regarded as a service process performed by a vehicle $v \in \mathcal{V}$, which includes dispatching the vehicle from the depot to the pickup location to perform the pickup subtask, transporting the cargo or baggage to the delivery location to complete the unloading subtask, and finally returning to the depot.

For each task τ, let s_τ and c_τ denote the start time and completion time of its pickup subtask at the pickup location, respectively. The pickup subtask must be executed within a specified time window $[E_\tau, L_\tau]$, where E_τ and L_τ denote the earliest and latest allowable service times. Moreover, due to aircraft cargo layout constraints, tasks belonging to the same flight must follow a predefined sequence. Specifically, the pickup subtask of τ can only start after that of its predecessor task τ' has been completed.

Furthermore, the following assumptions are adopted in this study. All logistics vehicles are homogeneous and travel within the airport at a uniform speed. Therefore, the travel time between different locations is determined by the corresponding travel distance. During operations, each vehicle is allowed to serve only one logistics task per trip. In addition, each logistics task is assumed to be a pre-defined service unit derived from airport operation planning, with its cargo load not exceeding vehicle capacity under the considered airport operational setting.

In practical airport logistics operations, the number of available vehicles is limited, and excessive task waiting time may lead to cascading delays and reduced operational efficiency. Therefore, it is essential to ensure timely and efficient task completion while maintaining effective vehicle utilization. To this end, a bi-objective optimization model is formulated as follows:

$$\min f_1 = \sum_{v \in \mathcal{V}} \alpha_v, \tag{1}$$

where α_v is a binary variable, and $\alpha_v = 1$ if vehicle v is activated for task execution; otherwise, $\alpha_v = 0$.

$$\min f_2 = \sum_{\tau \in \mathcal{T}} (s_\tau - r_\tau), \tag{2}$$

where r_τ is the release time of task τ, defined as

$$r_\tau = \begin{cases} a_{f_\tau}, & \text{if task } \tau \text{ has no predecessor} \\ c_{\tau'}, & \text{otherwise} \end{cases}, \tag{3}$$

where a_{f_τ} denotes the arrival time of flight f associated with task τ, and $c_{\tau'}$ denotes the completion time of the pickup subtask of the predecessor task τ'.

s.t.

$$r_\tau \leq s_\tau, \quad \forall \tau \in \mathcal{T}, \tag{4}$$

$$E_\tau \leq s_\tau \leq L_\tau, \quad \forall \tau \in \mathcal{T}, \tag{5}$$

$$\sum_{v \in \mathcal{V}} \beta_{\tau v} = 1, \quad \forall \tau \in \mathcal{T}, \tag{6}$$

$$\beta_{\tau v} \leq \alpha_v, \quad \forall \tau \in \mathcal{T}, \forall v \in \mathcal{V}, \tag{7}$$

where $\beta_{\tau v}$ is a binary variable, and $\beta_{\tau v} = 1$ if vehicle v is assigned to task τ; otherwise, $\beta_{\tau v} = 0$.

The objectives defined in Equations (1) and (2) aim to minimize the number of vehicles used and the total waiting time of all tasks, respectively. Equation (3) defines the release time of each task based on flight arrival time and predecessor pickup subtask completion time. Together with Equation (4), it ensures that each task cannot start before its release time, thereby ensuring the predefined task sequence. Equation (5) ensures that each pickup subtask is executed within its time window. Equation (6) guarantees that each task is assigned to exactly one vehicle. Equation (7) ensures that a vehicle is activated once it is assigned to any task.

Overall, the above formulation captures the essential scheduling objectives and fundamental constraints of the ALVS problem. To further analyze the computational complexity of the problem, we establish its NP-hardness in Theorem 1.

Theorem 1. *The ALVS problem is NP-hard.*

Proof. We prove the NP-hardness of the ALVS problem by reduction from the Flexible Job Shop Scheduling Problem (FJSP), which is a well-known NP-hard problem.

Consider a special case of the ALVS problem in which: (1) all travel times between locations are set to zero; (2) each logistics task corresponds to one operation; and (3) each flight corresponds to one job with a predefined task sequence.

Under this setting, each flight consists of a sequence of logistics tasks with precedence constraints, which is equivalent to the operation sequence of a job in the FJSP. Each logistics vehicle can process at most one task at a time, which corresponds to the machine capacity constraint in the FJSP. Assigning tasks to vehicles and determining their service order is therefore equivalent to assigning operations to machines and sequencing them in the FJSP.

Since the FJSP is NP-hard, and the ALVS problem generalizes the FJSP as a special case, the ALVS problem is also NP-hard. □

4. Algorithm Design

This section presents the proposed PERHO-OASS approach. First, we introduce the rolling horizon task categorization mechanism and describe the dynamic evolution of tasks across scheduling windows. Next, we detail the design of the hybrid PERHO optimization framework, which integrates periodic updates with event-driven rescheduling, as illustrated in Figure 1. Finally, we present the implementation of the OASS algorithm.

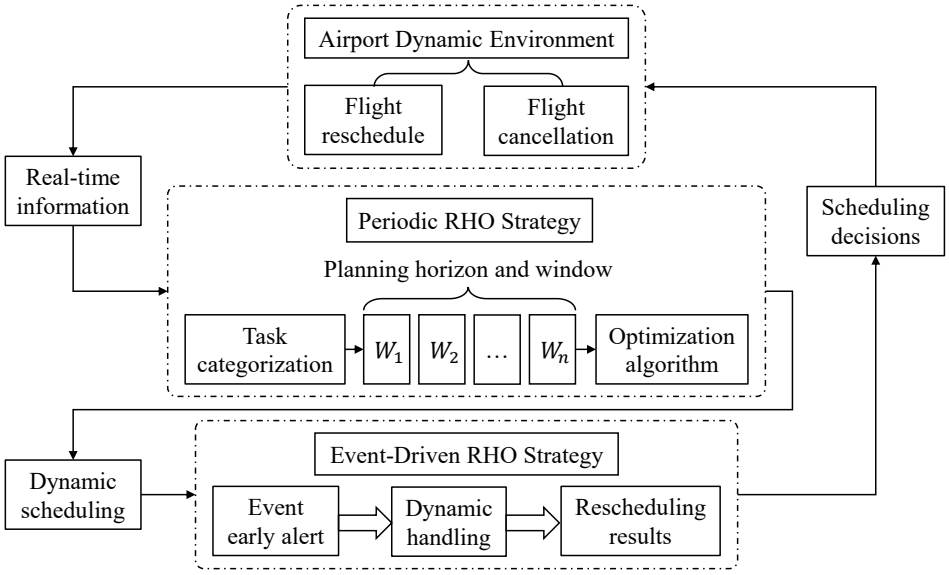


Figure 1. The flow chart of the Periodic and Event-Driven Rolling Horizon Optimization (PERHO).

4.1. Task Categorization

In the rolling horizon framework, tasks are categorized into three types according to their status relative to the current scheduling time: completed tasks, ongoing tasks, and waiting-for-service tasks, denoted by the sets $\mathcal{C}$, $\mathcal{G}$, and $\mathcal{W}$, respectively. Completed tasks have been fully executed in previous windows and are excluded from subsequent scheduling. Ongoing tasks are currently being processed but have not yet been completed, and may span multiple windows. Waiting-for-service tasks have not yet been assigned or executed and are considered for scheduling within the current window. As the time window rolls forward, tasks may transition from waiting-for-service to ongoing, and from ongoing to completed. However, in each window, only waiting-for-service tasks are involved in the scheduling and assignment process. Completed and ongoing tasks are retained for state tracking but do not influence scheduling decisions in subsequent windows.

Figure 2 illustrates the RHO process by depicting how tasks from multiple flights evolve across rolling windows, where Figure 2a,b correspond to the current and next rolling windows, respectively. The timeline is discretized by the rolling period Δt, and each window dynamically includes or excludes tasks based on their arrival and execution status. As the scheduling process advances from the current window (e.g., t_1) to the next window (t_2), newly arrived tasks such as T_{10} and T_{11} enter the scheduling window, while earlier tasks such as T_2 and T_3 have been completed. Meanwhile, tasks such as T_4, T_5, and T_6 transition from waiting-for-service to ongoing. This dynamic task categorization effectively reduces the scheduling scope and focuses computational effort on newly arrived tasks that require timely decision-making.

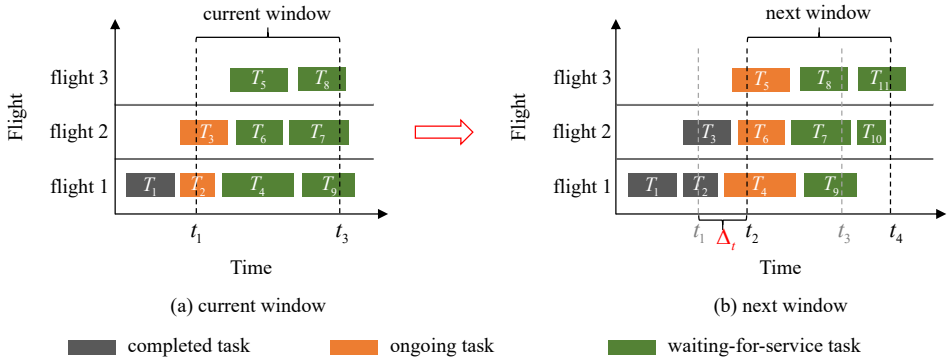
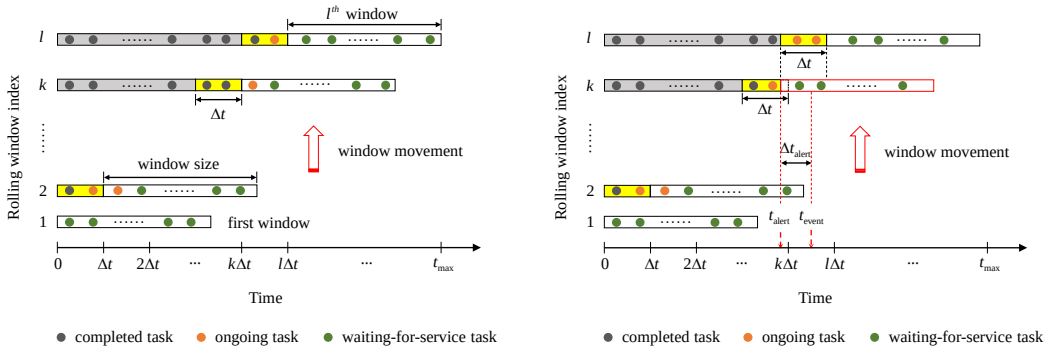


Figure 2. Task categorization in the rolling horizon optimization framework. The left part of the figure represents the current window (**a**), which transitions to the next window (**b**) as the time window moves forward by Δt, causing changes in task states.

4.2. PERHO Framework

The periodic RHO method, as shown in Figure 3a, ensures regular and stable scheduling by assigning tasks at fixed intervals. Although this method is simple and efficient, it cannot effectively handle dynamic disturbances, such as unexpected flight arrivals and flight cancellations, as random variations may disrupt the planned schedule and render it infeasible. In contrast, the event-driven approach dynamically adjusts the scheduling process in response to such events, enabling timely handling of disruptions and providing greater flexibility in highly dynamic environments. As illustrated in Figure 3b, when advance notice is received at the alert time t_{alert}, the scheduler first assigns all waiting-for-service tasks that arrived within the interval $[(k-1)\Delta t,\ t_{\text{alert}}]$. The rolling window is then shifted forward by Δt from t_{alert}, rather than waiting until $k\Delta t$ as in periodic RHO. Compared with fixed-period updates, event-driven triggers enable adaptive window shifting and dynamic rescheduling, which is particularly important in real-time changing airport environments. In practice, both modes are combined in a hybrid manner to balance stability and adaptability: the periodic mode is employed in the absence of events to maintain computational efficiency, while the event-driven mode is activated when dynamic changes occur to ensure timely and accurate scheduling updates.



(**a**) Schematic of the periodic RHO (**b**) Schematic of the event-driven RHO

Figure 3. Schematic of the PERHO framework illustrating both periodic and event-driven modes. The yellow box denotes the rolling period, and the upward red arrow indicates the direction of window movement.

Therefore, we propose the PERHO approach to solve the ALVS problem. In this framework, scheduling is performed only for tasks within the current window, which consists of tasks selected from those awaiting service. After each rolling period, completed tasks are removed, and newly arrived tasks are incorporated. This process is repeated

iteratively, ensuring that scheduling decisions are always based on tasks within the current window, until the end of the planning horizon.

In PERHO, some tasks assigned in the previous window may not have been executed in the current window. Two strategies can be adopted to handle such tasks: (a) retaining their previously assigned schedules, or (b) reassigning them in the current window. In strategy (a), the previously determined scheduling decisions are directly preserved, whereas strategy (b) allows both reassignment of tasks to different vehicles and adjustment of their execution order within the current window. Strategy (a) requires less computational effort but may perform poorly under unexpected disturbances, whereas strategy (b) provides greater flexibility at the cost of higher computational effort. Therefore, strategy (b) is adopted in response to dynamic events, while strategy (a) is used when no dynamic event occurs. The pseudocode of PERHO for dynamic ALVS is given in Algorithm 1, with two key parameters defined as follows:

Algorithm 1: PERHO for ALVS

Input: set of tasks $\mathcal{T}$, window size L, rolling period Δt
Output: Scheduling scheme

Initialize $t \leftarrow 0$, $\mathcal{C} \leftarrow \emptyset$, $\mathcal{G} \leftarrow \emptyset$, $\mathcal{W} \leftarrow \mathcal{T}$, and $\mathcal{V} \leftarrow \emptyset$;
while $\mathcal{C} \neq \mathcal{T}$ **do**
  $\mathcal{W}_t \leftarrow \{\tau \in \mathcal{W} \mid E_\tau \leq t + L\}$;
  if *an event occurs* **then**
    $t \leftarrow t_{\text{alert}}$;
    Categorize tasks in $\mathcal{W}_t$ using strategy (b);
  else
    Categorize tasks in $\mathcal{W}_t$ using strategy (a);
  Update $\mathcal{C}$ and $\mathcal{G}$;
  Remove tasks in $\mathcal{C} \cup \mathcal{G}$ from $\mathcal{W}_t$;
  Select a strategy $S \in \{\text{ESTP, ATP, RSTP}\}$ using OASS;
  Sort tasks in $\mathcal{W}_t$ according to strategy S;
  foreach *each* $\tau \in \mathcal{W}_t$ **do**
    Sort vehicles in $\mathcal{V}$ by their availability time ;
    foreach $v \in \mathcal{V}$ **do**
      if v *can complete* τ *within* $[E_\tau, L_\tau]$ **then**
        Assign v to τ;
        break;
  if τ *is not assigned* **then**
    Add a new vehicle v' to $\mathcal{V}$ and assign it to τ;
  Update vehicle set $\mathcal{V}$ and waiting task set $\mathcal{W}$;
  $t \leftarrow t + \Delta t$;
return *Scheduling scheme*

Window Size: The window size L determines the number of tasks considered at each decision stage. A larger window improves solution quality by incorporating more tasks but increases computational cost, whereas a smaller window reduces complexity at the expense of potential optimality.

Rolling Period: The rolling period Δt defines the update frequency of the scheduling process. A shorter Δt enables more frequent updates and improves responsiveness but may

reduce stability and increase computational overhead, whereas a longer Δt leads to more stable decisions with lower update frequency but reduced adaptability.

Therefore, the window size L and the rolling period Δt are typically determined empirically to balance solution quality, computational efficiency, and responsiveness.

4.3. Order-Aware Adaptive Strategy Selection

In the heuristic scheduling stage, three strategies (ESTP, ATP and RSTP) are employed to balance solution quality and computational efficiency. These strategies can be interpreted as Priority Dispatching Rules (PDRs), which determine task processing order based on predefined priorities and are widely used in scheduling problems [36]. Specifically, ESTP assigns tasks based on the earliest serviceable time and is suitable for tasks with strict time windows. ATP schedules tasks according to flight arrival times, favoring earlier arrivals. RSTP adopts a random assignment mechanism, which enhances diversity and helps avoid local optima. These strategies provide complementary advantages and enable flexible scheduling under different task characteristics.

AOS is a widely studied mechanism in metaheuristics for dynamically selecting among multiple heuristics based on their observed performance [37–39]. In many traditional approaches, operator performance is evaluated using a single credit vector, which often overlooks the sequential dependencies between operators. To address this limitation, we propose an OASS algorithm that explicitly considers the execution order of strategies. Specifically, OASS not only identifies effective strategies but also learns their optimal execution sequence. To this end, we model the transition probability between strategies using a 3×3 Guidance Matrix (GM), where each element $S_{i,j}$ represents the preference for selecting strategy S_j in the current rolling window given that strategy S_i was applied in the previous window.

$$GM = \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}. \tag{8}$$

Given the guidance matrix GM and the strategy S_i used in the previous window, OASS employs a roulette wheel selection mechanism to determine the strategy for the current window. The probability of selecting strategy S_j is defined as:

$$P(j) = \frac{S_{i,j}}{\sum_{k=1}^{3} S_{i,k}}. \tag{9}$$

After each rolling window, the matrix GM is updated to adapt to different search stages and improve solution quality. To ensure that all elements remain strictly positive while preserving their relative proportions, an exponential update mechanism is adopted. Specifically, the transition weight from S_i to S_j is updated based on a feedback value $\delta_{i,j}$, which is derived from the change in objective performance:

$$GM[i][j] = GM[i][j] \cdot \exp(\delta_{i,j}), \tag{10}$$

The feedback value $\delta_{i,j}$ is computed based on Pareto improvement as follows:

$$\delta_{i,j} = \begin{cases} \dfrac{|\mathcal{I}|}{|\mathcal{O}|}, & \text{if } \mathcal{I} \neq \emptyset \text{ and } \mathcal{D} = \emptyset \\ -\dfrac{|\mathcal{D}|}{|\mathcal{O}|}, & \text{if } \mathcal{D} \neq \emptyset \text{ and } \mathcal{I} = \emptyset, \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where $\mathcal{O}$ denotes the set of optimization objectives, $\mathcal{I} \subseteq \mathcal{O}$ the subset of improved objectives, and $\mathcal{D} \subseteq \mathcal{O}$ the subset of degraded objectives. $|\cdot|$ denotes the cardinality of a set.

All elements $S_{i,j}$ are initialized to 1, implying equal preference for all strategy transitions at the initial stage. The exponential update guarantees that all entries in GM remain strictly positive without requiring additional normalization.

Overall, the OASS algorithm selects a scheduling strategy at the beginning of each rolling window and applies it consistently within that window, while adaptively adjusting strategy preferences across consecutive windows based on their observed performance.

5. Experiments and Discussion

5.1. Experimental Setup

In this section, a series of experiments are conducted to evaluate the effectiveness of the proposed PERHO and OASS algorithms. The instances used for evaluation are derived from the benchmark (A3_ht) introduced in [13], which is constructed based on real-world airport flight schedule data. The time span of each instance is 8 h, and the number of tasks ranges from 100 to 400. Based on the distribution of task quantities in the benchmark, representative instances with 100, 200, 300, and 400 tasks are selected to evaluate the performance across different problem scales. To systematically investigate the impact of the rolling period and window size on solution quality, four different settings are considered for each parameter, selected according to the 8-h scheduling horizon of the benchmark instances to represent different update frequencies and planning horizons. Detailed configurations of these variables are provided in Table 1.

Table 1. Parameter settings used in the experiments.

Parameter	Values
Rolling period (min)	30, 40, 50, 60
Window size (min)	90, 120, 150, 180
Task quantity	100, 200, 300, 400

To assess the performance of the algorithm under different parameter combinations (rolling period and window size), each combination is run 500 times, with each run producing a single feasible solution, from which the non-dominated subset is extracted, along with the associated performance metrics. This procedure is repeated 20 times, and the final non-dominated solution set and the mean and standard deviation of the corresponding metrics for each parameter combination are obtained by aggregating the results across these repetitions.

We adopt the Hypervolume (HV) metric [40] and the objective function values (f_1 and f_2, corresponding to the two optimization objectives defined in Section 3) to evaluate solution quality. The HV metric is a widely used performance indicator in multi-objective optimization, measuring the volume dominated by the non-dominated solution set constructed from multiple independent runs in the objective space. A larger HV value indicates better performance. For the objective values reported in the following experiments, $f_1(f_2)$ denotes the solution with the minimum vehicle usage and its corresponding waiting time, while $(f_1)f_2$ denotes the solution with the minimum waiting time and its corresponding vehicle usage. The best results are highlighted based on the primary objective, with the secondary objective used for further comparison when multiple algorithms achieve the same primary value.

5.2. Parameter Analysis

To evaluate the impact of different rolling period and window size configurations across varying task quantities, we report the mean HV and its standard deviation (std) for instances with 100, 200, 300, and 400 tasks, as shown in Table 2. Values are rounded to two decimal places, with the best results highlighted in gray and bold. This analysis aims to identify parameter combinations that achieve both high solution quality and stable performance.

Table 2. Mean Hypervolume (HV) and standard deviation (std) for different rolling period and window size configurations across five problem instances for each task quantity.

Id	Rolling Period (min)	Window Size (min)	Task Quantity							
			100		200		300		400	
			Mean	Std	Mean	Std	Mean	Std	Mean	Std
1	30	90	0.73	0.03	0.66	0.11	0.71	0.10	0.81	0.05
2	30	120	0.76	0.03	0.67	0.07	0.67	0.06	0.79	0.06
3	30	150	0.77	0.01	0.68	0.11	0.71	0.07	0.81	0.04
4	30	180	**0.80**	0.06	0.66	0.08	0.69	0.08	**0.81**	0.02
5	40	90	0.71	0.08	0.72	0.06	0.66	0.09	0.79	0.05
6	40	120	0.75	0.05	0.72	0.04	0.62	0.09	0.80	0.05
7	40	150	0.73	0.06	0.72	0.05	0.66	0.11	0.80	0.02
8	40	180	0.70	0.07	0.70	0.06	0.66	0.15	0.78	0.04
9	50	90	0.72	0.07	0.78	0.07	0.70	0.06	0.76	0.03
10	50	120	0.70	0.06	0.78	0.08	0.75	0.12	0.77	0.04
11	50	150	0.71	0.07	**0.81**	0.05	0.69	0.07	0.76	0.02
12	50	180	0.74	0.06	0.78	0.08	**0.75**	0.10	0.77	0.01
13	60	90	0.68	0.07	0.72	0.06	0.68	0.09	0.73	0.04
14	60	120	0.69	0.02	0.74	0.02	0.65	0.04	0.73	0.03
15	60	150	0.68	0.05	0.72	0.06	0.67	0.07	0.76	0.04
16	60	180	0.69	0.06	0.72	0.05	0.65	0.10	0.71	0.05

Note: The best mean HV values for each task quantity are highlighted in gray and bold.

For instances with 100 tasks, the configuration with a rolling period of 30 min and a window size of 180 min achieves the highest mean HV (0.80), indicating superior optimization performance. Meanwhile, the configuration with a rolling period of 30 min and a window size of 150 min yields the lowest standard deviation (0.01), demonstrating strong stability. These two configurations are therefore recommended for small-scale instances, depending on whether solution quality or consistency is prioritized.

For medium-scale problems with 200 and 300 tasks, a rolling period of 50 min proves more effective. For 200 tasks, the configuration with a rolling period of 50 min and a window size of 150 min achieves the best performance, with a mean HV of 0.81. For 300 tasks, the configurations with a rolling period of 50 min and window sizes of 120 and 180 min both attain the highest mean HV (0.75). Among these, the configuration with a window size of 180 min demonstrates better robustness, as indicated by a lower standard deviation (0.10), making it more suitable in practice.

For instances with 400 tasks, the configurations with a rolling period of 30 min and window sizes of 150 and 180 min both achieve the highest mean HV (0.81). Notably, the configuration with a window size of 180 min exhibits the lowest standard deviation (0.02) across all experiments, indicating superior stability and effectiveness for large-scale instances.

Overall, the results indicate that optimal parameter selection is sensitive to problem scale. The effectiveness of the rolling period appears to depend on how tasks are distributed over time and how strongly they interact with each other. For small-scale instances, tasks are relatively sparse, making scheduling decisions more sensitive to local adjustments. For large-scale instances, the high task density increases competition for resources, which also makes the system more sensitive to scheduling changes. In both cases, shorter rolling

periods help maintain solution quality. In contrast, medium-scale instances tend to be more balanced, where a moderate rolling period provides a better trade-off between solution quality and computational effort. We also evaluated smaller rolling periods and larger window sizes, but these settings did not lead to significant improvements. Moreover, smaller rolling periods increased computational overhead, making them less efficient in practice.

5.3. Comparison of Different Algorithms

Table 3 compares ESTP, ATP, RSTP, AOS, MAB, and OASS across different task quantities, using the selected parameter settings (rolling period, window size). Experiments are conducted on five problem instances for each task quantity, and the results reveal the distinct characteristics of each algorithm while highlighting the superior performance of OASS. In addition to the fixed heuristic algorithms, two adaptive selection mechanisms are also included. Specifically, compared with OASS, the AOS used in this study adjusts the selection probabilities of candidate strategies according to their observed performance without modeling strategy transitions, where performance is evaluated based on the improvement or degradation in the objective values using the same criterion as in OASS, while MAB adopts an ϵ-greedy policy with $\epsilon = 0.1$ to balance exploration and exploitation. To further illustrate the performance of the different algorithms, a representative problem instance (instance id = 1) is selected from each task quantity category for detailed analysis.

Table 3. Performance comparison of six algorithms across different task quantities.

Task Quantity	Inst. ID	ESTP		ATP		RSTP		AOS		MAB		OASS (Ours)	
		f_1	f_2	f_1	f_2	f_1	f_2	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$
100 (30, 180)	1	6	453.73	6	423.37	7	373.57	6 (370.31)	(7) 246.21	6 (375.43)	(7) 222.89	**5** (528.09)	(7) **165.59**
	2	10	164.21	11	270.98	10	394.32	9 (283.08)	(10) 160.33	10 (305.83)	(11) 206.88	**8** (436.86)	(10) **154.25**
	3	9	569.96	8	814.11	8	477.80	**7** (568.66)	(11) 357.78	7 (592.58)	(12) 357.40	7 (571.67)	(10) **345.95**
	4	5	296.48	5	332.34	6	521.81	5 (299.96)	(11) 189.85	5 (299.96)	(10) 211.67	**5** (286.01)	(9) **172.97**
	5	9	191.35	6	266.09	7	249.83	5 (258.44)	(8) 120.73	5 (258.41)	(10) 158.48	**5** (258.19)	(9) **96.01**
200 (50, 150)	1	10	1829.15	9	1982.25	13	1729.49	8 (1113.84)	(10) 573.81	8 (1147.87)	(10) 976.06	**8** (1030.99)	(11) **438.22**
	2	8	1638.66	10	1642.98	14	1788.25	7 (797.72)	(12) 324.20	7 (812.76)	(9) 432.14	**7** (763.63)	(10) **277.67**
	3	8	1349.87	8	1735.97	12	2308.50	6 (1427.44)	(11) 410.22	6 (1216.35)	(9) 314.45	**6** (1205.16)	(12) **269.52**
	4	8	1406.80	8	1570.59	11	1965.58	6 (1183.13)	(9) 348.51	6 (1156.39)	(9) 351.35	**6** (1145.01)	(10) **339.11**
	5	9	1629.47	8	2010.23	11	2003.87	7 (943.42)	(9) 392.14	**7** (744.97)	(9) 384.98	7 (813.36)	(10) **298.32**
300 (50, 180)	1	17	2341.12	19	2917.70	25	2907.08	13 (1383.50)	(15) 1055.35	13 (1600.77)	(15) 1054.45	**12** (1797.29)	(16) **1022.17**
	2	18	2052.10	20	2287.19	26	2837.03	16 (1767.26)	(19) 1596.63	17 (1958.94)	(20) 1602.76	**15** (1969.36)	(19) **1347.38**
	3	18	1578.04	17	2422.83	24	2730.82	12 (1485.69)	(13) 1040.94	12 (1628.94)	(14) 1335.78	**12** (1100.47)	(13) **948.69**
	4	18	1680.18	16	2318.20	25	2262.81	12 (950.18)	(13) 923.60	11 (1483.40)	(16) 1298.04	**11** (1479.29)	(13) **667.76**
	5	17	1758.70	15	2750.90	22	2683.82	11 (1923.99)	(13) 828.48	**11** (1677.96)	(14) 1531.45	11 (1714.97)	(13) **757.59**
400 (30, 180)	1	16	1741.21	19	2062.64	24	2475.15	14 (2427.66)	(19) 981.86	14 (1849.86)	(21) 1075.34	**14** (1637.66)	(21) **843.58**
	2	16	1742.96	17	2459.50	23	2598.85	14 (2443.89)	(15) 1756.42	14 (2248.92)	(16) 1262.02	**14** (2066.16)	(16) **1260.67**
	3	16	1902.20	16	3026.49	21	2866.24	15 (1354.91)	(19) 831.63	15 (1416.99)	(22) 1077.03	**15** (1292.68)	(21) **789.11**
	4	14	2282.51	17	2494.72	24	2160.13	14 (2085.74)	(20) 990.13	14 (1731.15)	(20) 1018.11	**14** (1701.22)	(19) **884.39**
	5	15	2140.34	18	2309.78	24	2490.11	15 (1615.20)	(19) 960.21	14 (2290.53)	(20) 1322.36	**14** (2083.07)	(20) **791.01**

Note: The best results are highlighted in gray and bold based on the primary objective.

ESTP, ATP, and RSTP exhibit relatively limited performance compared with the adaptive selection methods, particularly as the task scale increases. ESTP achieves performance comparable to ATP in terms of vehicle usage but consistently requires more vehicles than OASS and results in higher waiting times. For example, in the first 300-task instance, ESTP uses 17 vehicles, whereas OASS requires only 12. In the first 100-task instance, ESTP yields

a waiting time of 453.73 min, significantly higher than the 165.59 min achieved by OASS. ATP performs similarly to ESTP in smaller instances but becomes less effective as the problem size increases. In the first 400-task instance, ATP requires 19 vehicles, compared to 14 for OASS. Additionally, ATP incurs higher waiting times than both ESTP and OASS in most cases. In the first 200-task instance, ATP results in a waiting time of 1982.25 min, which is higher than that achieved by other baseline methods. RSTP shows the weakest performance overall, requiring the largest number of vehicles across all task scales. Across all 400-task instances, RSTP requires more than 20 vehicles, and in several instances the number reaches 24, which is significantly higher than OASS. In terms of waiting time, RSTP increases substantially with task scale, ranging from 373.57 min for 100 tasks to 2475.15 min for 400 tasks. These results indicate that fixed heuristic strategies lack sufficient adaptability when handling large-scale dynamic scheduling problems.

AOS and MAB demonstrate better performance by dynamically selecting candidate strategies during the rolling optimization process. In terms of vehicle utilization, both methods consistently require fewer vehicles than the fixed heuristic algorithms in most instances. For example, in the first 300-task instance, both AOS and MAB achieve a minimum vehicle usage of 13 vehicles, outperforming all fixed heuristic algorithms. In terms of waiting time, both AOS and MAB also outperform ESTP, ATP, and RSTP in most cases. For instance, in the same 300-task instance, AOS and MAB reduce the waiting time to 1055.35 min and 1054.45 min, respectively, which are substantially lower than those achieved by the fixed heuristic algorithms. In several instances, AOS and MAB achieve the same minimum vehicle usage as OASS, but their corresponding waiting times are generally higher than those of OASS. In the third 100-task instance, however, AOS achieves the same minimum vehicle usage as OASS while obtaining a lower corresponding waiting time than other methods achieving the same vehicle usage, and is therefore highlighted in gray. Similar results are observed for MAB in the fifth 200-task instance and the fifth 300-task instance. Overall, although AOS and MAB significantly improve upon the fixed heuristic algorithms, their performance remains less stable than OASS across different problem instances.

In contrast, OASS demonstrates the best overall performance in most instances, achieving the minimum vehicle usage and waiting time across different problem settings. As shown in Table 3, even in cases where AOS or MAB attain the same minimum vehicle usage, OASS typically yields lower corresponding waiting times. Compared with the fixed heuristic algorithms, OASS reduces vehicle usage by an average of 26.42% and decreases task waiting time by 62.36%, demonstrating its substantial advantage in balancing resource utilization and service efficiency. Compared with the adaptive selection baselines AOS and MAB, OASS reduces vehicle usage by an average of 2.82% and decreases waiting time by 19.99%. These results indicate that OASS maintains superior performance across different problem instances.

Figure 4 presents the Gantt charts of all algorithms for the first 100-task instance. To ensure a fair comparison, solutions with similar vehicle usage are selected for visualization. It can be observed that, compared with the fixed heuristic algorithms, the adaptive methods generate more compact task schedules while using the same or fewer vehicles, thereby improving service efficiency and reducing task waiting time. This advantage is particularly evident for OASS.

In summary, OASS leverages dynamic strategy selection and sequencing based on real-time performance feedback and order relationships, enabling more efficient resource allocation and task scheduling. Compared with both fixed heuristic algorithms and conventional adaptive selection methods, OASS achieves more competitive and robust perfor-

mance, demonstrating the effectiveness of incorporating strategy sequencing into rolling scheduling decisions.

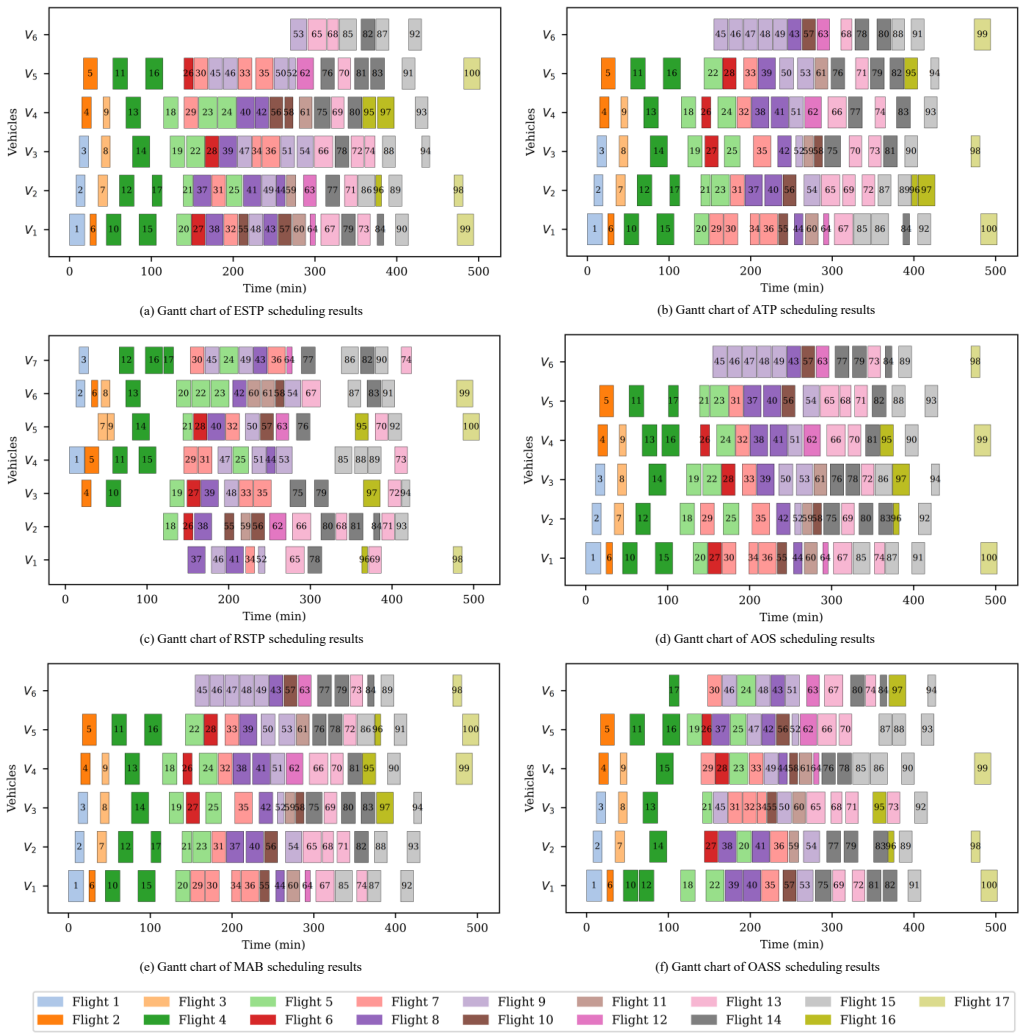


Figure 4. Gantt chart of logistics vehicle scheduling results for six algorithms in a 100-task instance.

5.4. Dynamic Event Analysis

To evaluate the performance of the proposed PERHO framework under dynamic events, new flight arrivals and flight cancellations are randomly introduced during the scheduling process. The details of the instance settings are reported in Table 4. Values in parentheses indicate the number of tasks added or removed during each event. The alert lead time Δt_{alert} is set to 30 min to provide sufficient buffer for rescheduling. These settings reflect the dynamic characteristics of airport operations and provide a basis for evaluating the algorithms under varying scenarios. Table 5 reports the comparative results for instances with task quantities ranging from 100 to 400.

Under dynamic event scenarios, the fixed heuristic algorithms exhibit limited adaptability, particularly when facing large-scale disruptions. ESTP generally maintains relatively stable vehicle usage under minor disturbances, but its rigid prioritization of serviceable time often leads to cascading delays when new flights are introduced or existing flights are canceled. ATP performs worse than ESTP under dynamic conditions, especially in terms of waiting time, due to its strict adherence to arrival order. RSTP shows the least reliable performance among the fixed heuristic algorithms, frequently requiring excessive vehicles and incurring high waiting times across different instances. As task scale increases, the limitations of fixed heuristics become more evident. For example, in Ins8, ESTP, ATP,

and RSTP require 18, 23, and 17 vehicles, respectively, while incurring waiting times of 2539.97, 3320.02, and 2717.28 min. In contrast, OASS requires only 12 vehicles and achieves a waiting time of 984.90 min. These results indicate that fixed heuristic algorithms struggle to effectively respond to dynamic disruptions and resource reallocation requirements.

Table 4. Details of instance configurations for dynamic event analysis.

Ins.	Rolling Period (min)	Window Size (min)	Task Quantity	Events
Ins1	30	180	100 (−3)	cancel a flight
Ins2	30	180	100 (+4)	schedule a new flight
Ins3	30	180	100 (−3, +4)	cancel a flight and schedule a new flights
Ins4	50	150	200 (−8)	cancel three flights
Ins5	50	150	200 (+9)	schedule four new flights
Ins6	50	150	200 (−8, +9)	cancel three flights and schedule four new flights
Ins7	50	180	300 (−25)	cancel five flights
Ins8	50	180	300 (+15)	schedule three new flights
Ins9	50	180	300 (−25, +15)	cancel five flights and schedule three new flights
Ins10	30	180	400 (−25)	cancel five flights
Ins11	30	180	400 (+10)	schedule five new flights
Ins12	30	180	400 (−25, +10)	cancel five fligths and schedule five new flights

Table 5. Performance comparison of six algorithms under different dynamic event scenarios.

Ins.	ESTP		ATP		RSTP		AOS		MAB		OASS (Ours)	
	f_1	f_2	f_1	f_2	f_1	f_2	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$	$f_1(f_2)$
Ins1	5	420.10	6	312.76	7	334.78	5 (489.75)	(8) 176.91	6 (289.90)	(9) 176.25	**5** (401.86)	(7) **131.73**
Ins2	6	460.10	6	435.01	7	402.97	6 (391.73)	(8) 252.16	6 (381.79)	(8) 238.05	**6** (318.29)	(8) **227.75**
Ins3	5	426.46	6	324.40	7	396.17	5 (437.51)	(7) 148.49	6 (387.07)	(10) 219.92	**5** (418.05)	(7) **113.99**
Ins4	10	1782.47	9	1602.27	13	1742.06	8 (1013.99)	(10) 565.35	8 (1076.82)	(11) 889.56	**7** (1510.13)	(10) **460.77**
Ins5	11	2029.05	10	2149.52	13	2251.34	8 (1371.25)	(10) 710.12	8 (1397.94)	(11) 1088.76	**8** (1366.84)	(10) **694.53**
Ins6	9	1906.04	10	1963.75	12	2023.98	8 (1116.08)	(9) 708.22	8 (1125.39)	(9) 1052.16	**8** (995.92)	(9) **697.97**
Ins7	15	1970.10	17	2721.17	17	1587.60	11 (1506.14)	(17) 791.37	12 (1712.85)	(16) 944.73	**11** (1436.39)	(18) **726.95**
Ins8	18	2539.97	23	3320.02	17	2717.28	13 (1474.24)	(16) 1064.49	13 (1667.23)	(15) 1102.7	**12** (1951.32)	(17) **984.90**
Ins9	17	2118.52	20	2879.36	17	1786.78	11 (1754.30)	(15) 861.71	12 (2056.57)	(15) 1101.75	**11** (1578.29)	(17) **810.11**
Ins10	15	1332.25	17	1956.24	20	2240.48	14 (1724.07)	(19) **927.31**	14 (1695.92)	(20) 1001.27	**14** (1669.78)	(19) 975.72
Ins11	16	2095.94	18	2640.46	23	2688.29	15 (1699.55)	(23) 1038.29	15 (1786.13)	(18) 938.76	**15** (1385.34)	(21) **761.83**
Ins12	15	1506.80	18	2354.20	20	2875.07	**14** (2179.51)	(19) 1001.43	14 (2447.58)	(17) 1052.98	14 (2232.57)	(18) **953.86**

Note: The best results are highlighted in gray and bold based on the primary objective.

AOS and MAB exhibit stronger adaptability than the fixed heuristic algorithms under dynamic event scenarios by dynamically adjusting strategy selection according to scheduling performance. Both methods achieve competitive results in several instances. For example, in Ins10, AOS achieves the best waiting time among all algorithms. In Ins12, AOS further achieves the best vehicle utilization while maintaining a competitive waiting time. MAB also improves upon the fixed heuristic algorithms in most dynamic instances, though its performance is slightly inferior to AOS. Specifically, MAB achieves the same minimum vehicle usage as OASS in 6 instances, whereas AOS matches OASS in 10 instances. Nevertheless, in most cases, their corresponding waiting times remain higher than those of OASS. Overall, while both AOS and MAB provide notable improvements over the fixed heuristics, their results are generally less consistent than OASS across the set of evaluated instances.

OASS consistently outperforms all other algorithms across the evaluated dynamic event instances, achieving the lowest vehicle usage and waiting times in most cases. On average, across all instances with task quantities ranging from 100 to 400, OASS reduces vehicle usage by 23.55% and waiting time by 61.95% compared with the fixed heuristic strategies, and, when compared to the adaptive baselines AOS and MAB, it achieves an additional reduction of 3.77% in vehicle usage and 17.30% in waiting time. These results indicate that OASS not only achieves superior efficiency and service quality but also maintains stable performance across a wide range of dynamic scheduling scenarios.

Its effectiveness in simultaneously optimizing vehicle allocation and task scheduling under varying disruptions highlights its robustness and practical applicability for airport ground logistics.

Figure 5 shows the Gantt chart of scheduling results for six algorithms on Ins2, in which four newly arrived tasks are considered. For better visual comparability, solutions with comparable vehicle usage are selected for illustration. It can be observed that OASS generates more compact task schedules than the baseline methods, leading to lower task waiting times under dynamic events.

Figure 5. Gantt chart of scheduling results for six algorithms on Ins2.

In summary, OASS outperforms all baseline strategies under dynamic events by dynamically selecting and sequencing strategies based on real-time conditions. These findings highlight the critical role of OASS in real-world airport logistics, where adaptability to frequent disruptions is essential.

5.5. Comparison with Global Scheduling

To further evaluate the effectiveness of the proposed PERHO-OASS approach, we compare its performance with global scheduling approaches based on ESTP and ATP. This setting can be regarded as an idealized scenario, where global methods have full prior knowledge of all tasks and dynamic events, providing a reference for comparison to assess whether the proposed PERHO-OASS approach can achieve comparable or even

superior performance. In the experimental setup, both global ESTP and ATP are applied to the same task sets under identical vehicle and time constraints as those used in the PERHO experiments described in Section 5.2. The results reported in Table 6 reveal several important findings.

Table 6. Performance comparison of PERHO-Order-aware Adaptive Strategy Selection (OASS) and global scheduling using Earliest Serviceable Time Priority (ESTP) and Arrival Time Priority (ATP) algorithms.

Task Quantity	ESTP		ATP		PERHO-OASS (Ours)	
	f_1	f_2	f_1	f_2	f_1	f_2
100 (−3)	**5**	439.53	7	517.72	**5**	**401.86**
100 (+4)	**6**	485.06	8	488.08	**6**	**318.29**
100 (−3, +4)	**5**	445.89	7	529.36	**5**	**418.05**
200 (−8)	10	1822.42	10	1730.03	**7**	**1510.13**
200 (+9)	10	2060.45	10	2050.89	**8**	**1256.37**
200 (−8, +9)	10	**1797.71**	10	1925.26	**7**	1996.41
300 (−25)	16	1890.09	17	2603.21	**11**	**1436.39**
300 (15)	19	2586.98	21	3331.36	**12**	**1951.32**
300 (−25, +15)	18	2128.62	19	3065.92	**11**	**1578.29**
400 (−25)	**16**	**1154.14**	19	2030.37	**16**	1669.78
400 (+10)	17	**2013.58**	18	3014.16	**15**	2081.82
400 (−25, +10)	16	**1166.02**	19	2379.93	**14**	2232.57

Note: The best results are highlighted in gray and bold based on the primary objective.

In terms of vehicle usage, PERHO-OASS consistently demonstrates superior performance in minimizing the number of vehicles required. For small-scale instances with 100 tasks, PERHO-OASS matches the best result achieved by ESTP, requiring only five vehicles, and outperforms ATP by reducing the vehicle count by two in cases such as task cancellations. As the problem scale increases, the advantage of PERHO-OASS becomes more pronounced. For large-scale scenarios involving 400 tasks, PERHO-OASS requires only 14 to 16 vehicles, whereas ESTP and ATP require 16–17 and 18–19 vehicles, respectively. A particularly notable case is observed for 300 tasks with 25 cancellations, where PERHO-OASS uses only 11 vehicles compared to the 17 vehicles required by ATP, representing a 35% reduction. This improvement suggests that the PERHO-OASS approach is able to achieve competitive performance even when compared to global ESTP and ATP, which have full prior knowledge of dynamic events but rely on static scheduling decisions.

Regarding waiting time optimization, PERHO-OASS achieves the best or highly competitive results across most scenarios. For example, in the case of 100 tasks with four new arrivals, PERHO-OASS attains a total waiting time of 318.29 min, outperforming both ESTP (485.06 min) and ATP (488.08 min). In the case of 100 tasks with three cancellations, as shown in Figure 6, PERHO-OASS achieves performance comparable to that of ESTP under global scheduling, while achieving a waiting time of 401.86 min, which is lower than the 517.72 min obtained by ATP. There are occasional exceptions. For instance, in the 200-task scenario with combined disruptions, PERHO-OASS results in a waiting time of 1996.41 min, which is higher than the 1797.71 min achieved by ESTP. However, this is accompanied by a significant reduction in vehicle usage, with PERHO-OASS requiring seven vehicles compared to ten for ESTP, reflecting a deliberate trade-off between objectives. Similar exceptions can also be observed in the instances with 400 tasks. These results suggest that the proposed PERHO-OASS approach is able to maintain competitive performance while balancing trade-offs between objectives.

PERHO-OASS also demonstrates remarkable robustness under combined disruptions. For example, in the scenario with 300 tasks, 25 cancellations, and 15 new arrivals, PERHO-OASS maintains 11 vehicles, compared with 18 for ESTP and 19 for ATP, while achieving

a total waiting time of 1436.39 min, which is less than half of that for ATP, 3065.92 min. In the 400-task scenario with 25 cancellations and 10 new arrivals, PERHO-OASS utilizes 14 vehicles and attains a waiting time of 2232.57 min, outperforming ATP, which requires 19 vehicles and incurs 2379.93 min of waiting. These results indicate that PERHO-OASS maintains both stability and efficiency even as the complexity of disruptions increases, whereas global scheduling approaches degrade significantly in both objectives. Notably, PERHO-OASS occasionally accepts slightly longer waiting times in exchange for substantial reductions in vehicle usage, reflecting a practical trade-off aligned with operational constraints in airport logistics.

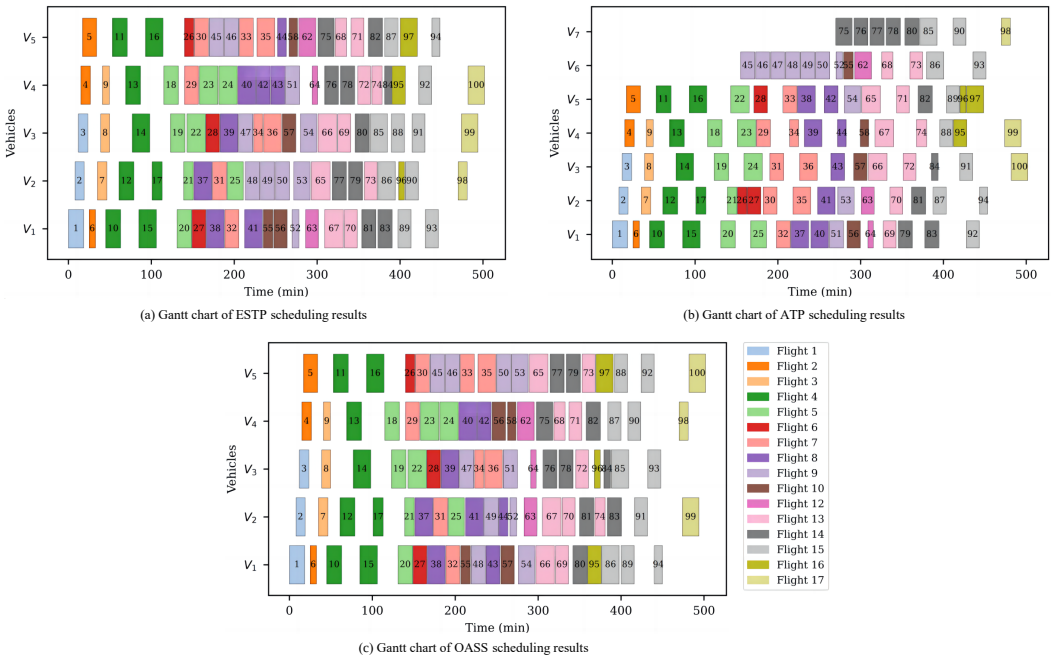


Figure 6. Gantt chart of scheduling results for three algorithms under a flight cancellation event (removing three tasks) in a 100-task instance.

The superior performance of PERHO-OASS can be attributed to several key factors. Its adaptive strategy selection mechanism enables dynamic switching among ESTP, ATP, and RSTP based on real-time feedback, whereas global ESTP and ATP are constrained by static priorities and cannot respond to instantaneous changes due to their fixed rules. The rolling window mechanism in PERHO-OASS prevents overcommitment to outdated plans and facilitates timely responses to newly arising events. Moreover, as problem scale increases, global scheduling approaches become computationally prohibitive, while the decomposition of the problem into smaller subproblems within PERHO-OASS ensures both scalability and tractability.

In summary, the advantages of PERHO-OASS, demonstrated in dynamic event scenarios (Section 5.4), also extend to static global benchmarks. This highlights the generalizability and robustness of the proposed approach and indicates that traditional global scheduling methods may struggle to effectively handle dynamic airport logistics, whereas PERHO-OASS provides a practical and effective alternative.

6. Conclusions

In this paper, we propose a hybrid PERHO framework that integrates periodic and event-driven RHO to address the ALVS problem. The approach dynamically categorizes tasks into completed, ongoing, and waiting-for-service sets, enabling adaptive scheduling

under real-time disruptions. Three heuristic strategies are incorporated, together with an OASS algorithm to balance solution quality and computational efficiency.

Experimental results under both static and dynamic scenarios demonstrate that the proposed method consistently outperforms fixed heuristic algorithms and adaptive selection methods in terms of vehicle utilization and task waiting time, validating its effectiveness in dynamic airport logistics scheduling. The proposed approach can support airport operators in improving the efficiency and reliability of ground logistics operations.

Despite these promising results, several limitations remain. First, the current study simplifies real-world airport operations and does not fully capture complex uncertainties, such as weather conditions and air traffic control disruptions. Second, the experiments are conducted on benchmark instances derived from a specific airport logistics setting, and further validation is needed to assess the scalability and generalizability of the proposed method across more diverse airport environments.

Future research will incorporate predictive models based on machine learning or time-series forecasting methods to better anticipate task variations and support proactive scheduling decisions under more complex operational disruptions. The proposed approach will also be further validated across airports with different operational characteristics to enhance its practical applicability.

Author Contributions: Conceptualization, R.F. and Q.-Q.Z.; methodology, R.F. and Z.C.; software, R.F. and Z.C.; validation, Z.C. and B.L.; formal analysis, B.L.; resources, B.L.; data curation, Z.C.; writing—original draft preparation, R.F.; writing—review and editing, B.L. and Q.-Q.Z.; supervision, Q.-Q.Z.; funding acquisition, R.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

ALVS	Airport Logistics Vehicle Scheduling
IoT	Internet of Things
RHO	Rolling Horizon Optimization
PERHO	Periodic and Event-Driven Rolling Horizon Optimization
OASS	Order-aware Adaptive Strategy Selection
ESTP	Earliest Serviceable Time Priority
ATP	Arrival Time Priority
RSTP	Random Serviceable Time Priority
AOS	Adaptive Operator Selection
MAB	Multi-Armed Bandit
FJSP	Flexible Job Shop Scheduling Problem
PDRs	Priority Dispatching Rules
GM	Guidance Matrix
HV	Hypervolume
std	standard deviation

References

1. Zhang, Z.; Che, Y.; Liang, Z. Split-demand multi-trip vehicle routing problem with simultaneous pickup and delivery in airport baggage transit. *Eur. J. Oper. Res.* **2024**, *312*, 996–1010. [CrossRef]
2. Feng, R.; Gao, W.; Li, Y.; Cai, Z.; Chen, G.; Xu, M. Airport ground cargo transportation scheduling problem: Simultaneously considering logistics vehicles and handling facilities under strict time restriction. *Appl. Intell.* **2025**, *55*, 1097. [CrossRef]
3. Cai, Z.; Gao, W.; Feng, R.; He, S.; Jin, Y.; Li, Y.; Xu, M. A Clustering-based Single-parent Genetic Algorithm for Airport Logistics Vehicle Scheduling. *Mach. Intell. Res.* **2026**, *23*, 692–720. [CrossRef]
4. Hu, J.; Yi, K.; Han, X.; He, D. Optimal Scheduling of Airport Refueling Vehicle Based on Improved NSGA-II. In Proceedings of the 2023 China Automation Congress (CAC), Chongqing, China, 17–19 November 2023; pp. 146–151. [CrossRef]
5. Wang, Y.; Zhang, Z.; Quan, W. A Multi-strategy Enhanced Memetic Algorithm for Airport Refueling Vehicle Scheduling Problem. In Proceedings of the 2024 43rd Chinese Control Conference (CCC), Kunming, China, 28–31 July 2024; pp. 1897–1902. [CrossRef]
6. van Oosterom, S.; Mitici, M.; Hoekstra, J. Dispatching a fleet of electric towing vehicles for aircraft taxiing with conflict avoidance and efficient battery charging. *Transp. Res. Part C Emerg. Technol.* **2023**, *147*, 103995. [CrossRef]
7. Zoutendijk, M.; Mitici, M. Fleet scheduling for electric towing of aircraft under limited airport energy capacity. *Energy* **2024**, *294*, 130924. [CrossRef]
8. Ali, K.; Ioannou, G.; Lovieris, P.; Cosmas, J.; Taylor, M.; Dimitrakakis, G.; Chome, E.; Mete, S.; Kizil, T.; Celik, S.; et al. IoT Data Mining of Air Cargo for Autonomous Artificial Intelligent Raising of Import Duties by Revenue and Customs. In Proceedings of the 2025 6th International Conference on Computer Vision and Data Mining (ICCVDM), London, UK, 12–14 September 2025; pp. 365–369. [CrossRef]
9. Gao, X.; Peng, Y.; Chen, L.; Zhu, Z.; Ding, J.; Zhou, X.; Deng, W. MRCFSSA: A Multi-strategy Sparrow Search Algorithm for Robust Airport Gate Allocation in Dynamic IoT Environments. *IEEE Trans. Consum. Electron.* 2026, *early access*. [CrossRef]
10. Zheng, H.; Xu, W.; Ma, D.; Qu, F. Dynamic rolling horizon scheduling of waterborne AGVs for inter terminal transportation: Mathematical modeling and heuristic solution. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 3853–3865. [CrossRef]
11. Xie, M.; Wang, H.; Gao, Y.; Wang, Y. A rolling-horizon framework for managing shared parking and electric vehicle charging. *Sustain. Cities Soc.* **2023**, *98*, 104810. [CrossRef]
12. Tian, Q.; Lin, Y.H.; Wang, D.Z.; Yang, K. Toward real-time operations of modular-vehicle transit services: From rolling horizon control to learning-based approach. *Transp. Res. Part C Emerg. Technol.* **2025**, *170*, 104938. [CrossRef]
13. Cai, Z.; Gao, W.; Feng, R.; Li, Y.; Xu, M. Benchmark for the scheduling problems of airport ground support operations and a case study. *Appl. Soft Comput.* **2025**, *169*, 112555. [CrossRef]
14. Lv, L.; Deng, Z.; Shao, C.; Shen, W. A variable neighborhood search algorithm for airport ferry vehicle scheduling problem. *Transp. Res. Part C Emerg. Technol.* **2023**, *154*, 104262. [CrossRef]
15. Zhang, X.; Wang, X.; Dong, W.; Xu, G. Adaptive large neighborhood search for autonomous electric vehicle scheduling in airport baggage transport service. *Comput. Oper. Res.* **2025**, *182*, 107086. [CrossRef]
16. Ahmadi, S.; Akgunduz, A. Airport operations with electric-powered towing alternatives under stochastic conditions. *J. Air Transp. Manag.* **2023**, *109*, 102392. [CrossRef]
17. Liu, Y.; Wu, J.; Tang, J.; Wang, W.; Wang, X. Scheduling optimisation of multi-type special vehicles in an airport. *Transp. B Transp. Dyn.* **2022**, *10*, 954–970. [CrossRef]
18. Luo, Q.; Liu, H.; Liu, C.; Deng, Q. Multi-strategy cooperative scheduling for airport specialized vehicles based on digital twins. *Sci. Rep.* **2024**, *14*, 15533. [CrossRef]
19. Fu, W.; Li, J.; Liao, Z.; Fu, Y. A bi-objective optimization approach for scheduling electric ground-handling vehicles in an airport. *Complex Intell. Syst.* **2025**, *11*, 209. [CrossRef]
20. Han, X.; Zhao, P.; Kong, D. Two-stage optimization of airport ferry service delay considering flight uncertainty. *Eur. J. Oper. Res.* **2023**, *307*, 1103–1116. [CrossRef]
21. Zhu, S.; Sun, H.; Guo, X. Cooperative scheduling optimization for ground-handling vehicles by considering flights' uncertainty. *Comput. Ind. Eng.* **2022**, *169*, 108092. [CrossRef]
22. Cuisinier, É.; Lemaire, P.; Penz, B.; Ruby, A.; Bourasseau, C. New rolling horizon optimization approaches to balance short-term and long-term decisions: An application to energy planning. *Energy* **2022**, *245*, 122773. [CrossRef]
23. Abedi, S.; Kwon, S. Rolling-horizon optimization integrated with recurrent neural network-driven forecasting for residential battery energy storage operations. *Int. J. Electr. Power Energy Syst.* **2023**, *145*, 108589. [CrossRef]
24. Junior, M.W.S.; de Oliveira Rocha, H.R.; Salles, J.L.F. A smart energy scheduling under uncertainties of an iron ore stockyard-port system using a rolling horizon algorithm. *Comput. Oper. Res.* **2024**, *164*, 106518. [CrossRef]
25. Brusco, G.; Menniti, D.; Sorrentino, N. A rolling horizon management model to reduce imbalance in real-time for a renewable energy community. *Sustain. Energy Grids Netw.* **2025**, *43*, 101828. [CrossRef]

26. Yan, D.; Wen, B.; Zhu, B.; Li, Y.; Tang, L.; Hou, J. Rolling Horizon Scheduling Optimization Method for Vehicle-to-Grid Interaction Oriented to Multi-Regulation Requirements of Power Grids. In Proceedings of the 2025 15th International Conference on Power and Energy Systems (ICPES), Chongqing, China, 6–8 December 2025; pp. 50–55. [CrossRef]
27. Raphael, H.; Lars, M. A rolling horizon planning approach for short-term demand supply matching. *Cent. Eur. J. Oper. Res.* **2023**, *32*, 865–896. [CrossRef]
28. Zeiträg, Y.; Figueira, J.R. A novel machine-learning rolling horizon heuristic for dynamic lot-sizing and job shop scheduling problems. *Int. J. Prod. Res.* **2025**, *63*, 4563–4589. [CrossRef]
29. Larizadeh, R.; Tosarkani, B.M. A novel data-driven rolling horizon production planning approach for the plastic industry under the uncertainty of demand and recycling rate. *Expert Syst. Appl.* **2025**, *263*, 125728. [CrossRef]
30. Boujnah, I.; Tlili, M.; Korbaa, O. A modified particle swarm optimization algorithm in a rolling horizon framework for the aggregate production planning problem: Pharmaceutical industry case. *Ann. Oper. Res.* **2025**, *355*, 3081–3098. [CrossRef]
31. Cruz, J.A.d.; Salles-Neto, L.L.d.; Schenekemberg, C.M. A Rolling Horizon Approach for a Production Planning and Inventory Management Problem Under Stochastic Demand. *Process Integr. Optim. Sustain.* **2025**, *9*, 2021–2046. [CrossRef]
32. Shalaby, A.A.; Shaaban, M.F.; Mokhtar, M.; Zeineldin, H.H.; El-Saadany, E.F. A dynamic optimal battery swapping mechanism for electric vehicles using an LSTM-based rolling horizon approach. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 15218–15232. [CrossRef]
33. Zheng, W.; Zhou, H. Dual-Sourcing Inventory Routing Problem with Route-Dependent Lead Times in Rolling Horizon Framework. *Appl. Sci.* **2023**, *13*, 2229. [CrossRef]
34. Baldouski, D.; Krész, M.; Dávid, B. Scheduling truck arrivals for efficient container flow management in port logistics. *Cent. Eur. J. Oper. Res.* **2025**, *33*, 791–818. [CrossRef] [PubMed]
35. Abbaszadeh, A.; Doulabi, H.H. Drone-aided mobile blood collection problem: A rolling-horizon-based matheuristic. *Comput. Oper. Res.* **2025**, *185*, 107253. [CrossRef]
36. Li, Y.; Wang, Q.; Li, X.; Gao, L.; Fu, L.; Yu, Y.; Zhou, W. Real-time scheduling for flexible job shop with AGVs using multiagent reinforcement learning and efficient action decoding. *IEEE Trans. Syst. Man Cybern. Syst.* **2025**, *55*, 2120–2132. [CrossRef]
37. Fialho, Á.; Schoenauer, M.; Sebag, M. Toward comparison-based adaptive operator selection. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, OR, USA, 7–11 July 2010; pp. 767–774. [CrossRef]
38. Fialho, Á.; Da Costa, L.; Schoenauer, M.; Sebag, M. Analyzing bandit-based adaptive operator selection mechanisms. *Ann. Math. Artif. Intell.* **2010**, *60*, 25–64. [CrossRef]
39. Maturana, J.; Lardeux, F.; Saubion, F. Autonomous operator management for evolutionary algorithms. *J. Heuristics* **2010**, *16*, 881–909. [CrossRef]
40. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **2002**, *3*, 257–271. [CrossRef]

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.