

Article

An Event-Driven and Feasibility-Audited Decision-Support Framework for Dynamic Rescheduling of Inland Container Depot Truck Operations

Shucheng Fan *  and Shaochuan Fu

School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China; shchfu@bjtu.edu.cn

* Correspondence: scfan@bjtu.edu.cn

Highlights

Please indicate how your work links to systems science via your contributions to systems practice, theory, and/or methodology.

- This study contributes to systems methodology by integrating offline E-ALNS planning, event-triggered local repair, six explicit feasibility audits, relational action ranking, discrete-event execution, and traceable logs into a closed ICD rescheduling loop.
- It advances systems practice by treating learned dispatch as a governed subsystem. Feasibility, plan stability, response latency, and interpretability are evaluated alongside completion and reward, while deterministic fallbacks and dispatcher oversight remain explicit.

What are the main findings and/or the implications of the main findings?

- Across 552 tested states and 4,968,000 task–vehicle action cells, the mask produced zero false positives or negatives against a separately coded oracle for the six implemented predicates; removing masking sharply reduced completion and reward.
- Transformer-PPO improved held-out completion, reward, and plan stability over the matched MLP, yet no method dominated completion, reward, stability, and latency; deployment should therefore retain governed ranking, deterministic fallbacks, and human review.

Abstract

Inland container depot (ICD) truck schedules must absorb new orders, service delays, appointment changes, congestion, and port cut-offs without destabilizing an already executed plan. This study asks whether event-triggered local repair can be separated into an explicit business-rule audit and a learned ranking of feasible task–vehicle actions. The proposed decision-support framework connects a static baseline, candidate task chains, six modeled hard-feasibility predicates, a Transformer encoder trained with proximal policy optimization (Transformer-PPO), and discrete-event execution logs. A five-seed, 120-episode confirmation gave Transformer-PPO a held-out online completion proxy (α_{online}) of 0.3226 and reward of 110.58, compared with 0.2581 and 61.87 for the matched multilayer perceptron (MLP); deterministic rules and search remained competitive. An independent audit of 4,968,000 action cells across 552 decision states found no disagreement with an independently coded oracle for the implemented hard predicates, while a reward-weight screen exposed the expected efficiency-stability trade-off. Together with a rolling-horizon comparator and a three-scale by three-disturbance stress test, the evidence supports an auditable system-integration contribution, not a new generic reinforcement learning (RL) algorithm



Academic Editor: Emilio Larrodé

Received: 13 July 2026

Revised: 13 August 2026

Accepted: 15 August 2026

Published: 20 August 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.

or universal performance superiority. Claims are limited to synthetic simulation-based decision support.

Keywords: inland container depot; dynamic rescheduling; logistics systems; reinforcement learning; Transformer; proximal policy optimization; action masking; discrete-event simulation; systems engineering

1. Introduction

Inland container depots (ICDs) support port-hinterland logistics by coordinating empty-container release, factory loading, heavy-container pickup, temporary storage, and port-bound gate-in operations. Such operations belong to the broader class of intelligent freight-transportation systems, where routing, scheduling, terminal capacity, real-time information, and operations research methods jointly shape system performance [1]. They also share important features with multimodal freight transportation planning, because local truck decisions are coupled with port access, container availability, and time-sensitive hinterland flows [2]. In daily operations, a depot dispatching plan is often generated before the execution period begins. This plan specifies which truck should serve which task, when key pickups and deliveries should occur, and how much time buffer is available before hard operational deadlines. However, the executed system is not static. Factory loading and unloading may take longer than expected, port appointments may be released or changed during the day, road conditions may alter transfer times, and urgent heavy-container tasks may enter the system with strict cut-off windows. These events can invalidate a previously feasible plan within minutes.

Static vehicle routing treats a route as a complete object. ICD rescheduling is different: the dispatcher observes a partially executed plan and must decide whether a limited repair is enough. Local container drayage studies show that port access appointments materially affect feasible truck planning [3], while empty-container reuse research shows that depot and port decisions are coupled across time [4]. The feasibility of a dispatch action depends on the truck's current location, the previous task in the same order, appointment readiness, remaining slack, resource compatibility and the effect on downstream committed tasks. Excessive replanning is also costly. If every disturbance triggers global rerouting, driver instructions and customer notifications become unstable, and the dispatcher may lose confidence in the decision support system. This is the plan-stability and rescheduling-frequency concern discussed in the rescheduling literature [5–8].

For this reason, a practical ICD rescheduling system should satisfy four requirements. First, it should react only when operationally meaningful events occur, rather than forcing decisions at fixed clock intervals when no useful action is available. Second, it should restrict changes to a controllable local neighborhood so that the baseline execution plan remains understandable. Third, it should guarantee that recommended actions satisfy hard business constraints before they reach the dispatcher or execution interface. Fourth, it should evaluate performance through both efficiency and stability, because a low-distance solution that frequently destroys the original plan may be unacceptable in practice.

Reinforcement learning (RL) is useful here because the value of a repair is observed over later events, not only at the instant of assignment. Recent surveys on machine learning and RL for combinatorial optimization show the potential of learning-based routing and scheduling, while also emphasizing constraint handling, representation, generalization, and evaluation [9,10]. A direct RL formulation is still inadequate for ICD operations. The action space contains many infeasible task–vehicle pairs, and the state has a variable

number of tasks, vehicles, and environmental conditions. If infeasible actions are handled only through penalties, early training can be dominated by illegal dispatches and sparse failure signals. A logistics policy intended for dispatcher use therefore needs an explicit feasibility audit before action selection.

The objective is to design and test an auditable interface between a precomputed ICD truck plan and event-triggered local repair. The neural policy is only one ranking component inside this interface; static planning, business-rule validation, execution simulation, and dispatcher-readable logs remain explicit system modules.

The research question is as follows: can an ICD rescheduling system preserve hard task-chain, appointment, and time-window feasibility by construction while using learned relational ranking to select among feasible local repairs, and what stability, latency, and performance trade-offs arise relative to rule, heuristic and rolling-search methods?

Existing neural route-construction methods do not jointly represent a partially executed baseline, multi-stage container task chains, appointment release, port cut-offs, and plan-stability concerns. Exact and metaheuristic drayage models can encode many of these constraints, but they usually construct or re-solve a plan rather than provide a reusable event-level interface for feasible-action ranking and operational auditing. The unresolved issue is therefore how to integrate these functions for governed ICD runtime repair; the study does not claim that any prior method is intrinsically incapable of doing so.

Related Literature and Research Gap

The study draws on freight transportation systems, dynamic vehicle routing, rescheduling theory, and learning-based combinatorial optimization. Crainic et al. showed that intelligent freight-transportation systems depend on operations research, information flows, and real-time decision support [1]. SteadieSeifi et al. reviewed multimodal freight transportation planning and showed that operational decisions are embedded in tactical and strategic contexts [2]. ICD truck dispatching has a layered structure; an instruction to a single truck is local, but its feasibility depends on container flow, port appointment conditions, fleet availability, and downstream service commitments.

Container and drayage studies have established that appointment access [3], dynamic empty-container reuse [4], terminal-operation interactions [11], and multi-trip synchronization [12] materially shape truck schedules. Recent work reviews drayage operations and management [13], optimizes synchronized empty-container reuse [12], formulates multi-stage dynamic booking and routing [14], models stochastic truck appointments for up to 300 orders [15], and integrates appointments with separation-mode operations [16]. These models show that ICD dispatching cannot be reduced to distance minimization. They also clarify the boundaries of this paper; optimization builds strong plans, whereas the present study examines a governed interface for repairing an executing plan after an event.

Dynamic vehicle routing distinguishes static route construction from sequential decisions under information arrival [17–19]. Recent surveys and method studies identify a central difficulty; applying RL to large constrained combinatorial action spaces remains challenging [20–22]. ICD operations add multi-stage precedence, appointment release, cut-off protection, and baseline-continuity concerns. The framework therefore restricts RL to ranking an explicitly constructed feasible set rather than asking it to discover all business rules from penalties.

Rescheduling and dynamic scheduling literature provides the fourth layer. Vieira et al. proposed a framework of rescheduling strategies, policies and methods for manufacturing systems [5]. Ouelhadj and Petrovic surveyed dynamic scheduling in manufacturing [6], and Aytug et al. reviewed production schedule execution under uncertainty [7]. Herroelen and Leus reviewed robust and reactive project scheduling and emphasized that uncertainty

handling can be proactive, reactive, or predictive–reactive [8]. Although these studies are not specific to ICD operations, they provide two principles that are directly relevant here. First, the stability of the original plan should be evaluated, not ignored. Second, a rescheduling trigger should be meaningful; rescheduling too often can be as damaging as rescheduling too late.

Learning-based combinatorial optimization supplies policy learning and relational representations [9,10,23,24]. Recent work also shows that complex feasibility masks may be computationally difficult [22] and that dynamic-routing performance is sensitive to state-space design [25]. Accordingly, invalid-action masking [26], PPO [27] and self-attention [28] are treated here as established components. The research object is the ICD-specific system boundary, constraint audit and execution interface in which those components are embedded.

The recent literature published during 2022–2025 covers stochastic and dynamic routing learning [20,21,25,29–33], appointment and drayage optimization [12–16,34], and complex constraint handling [22]. Across this literature, no single method dominates in terms of exactness, real-time response, future-value estimation, constraint coverage, and operational traceability. This motivates a modular and explicitly bounded decision-support architecture rather than a claim of a universally superior algorithm.

Table 1 makes the contribution boundary explicit. Invalid-action masking [26], PPO [27], and self-attention [28] are established components; the claimed contribution is their integration with an executed ICD baseline generated by enhanced adaptive large neighborhood search (E-ALNS), task-chain and appointment audits, event-triggered local repair, and a dispatcher-readable audit trail.

Table 1. Methodological positioning matrix: the contribution is the ICD-specific integration rather than a new generic masking operator.

Research Stream	Decision Setting	Constraint Treatment	Executed-Baseline Repair	Distinctive Scope
Generic invalid-action masking [26]	General constrained RL	Masks invalid actions	Not intrinsic	Generic policy-gradient constraint handling
Neural route construction [23,24]	Mostly one-shot construction	Problem masks during decoding	No partially executed ICD plan	Learned route construction
Dynamic-routing RL [20,30,32]	Dynamic requests/traffic	Model-specific feasibility rules	Varies	Dynamic vehicle-routing policies
Optimization-enriched machine learning [21]	Dynamic time-window routing	Optimization embedded with learning	Not the present ICD task-chain interface	Structured optimization-learning coupling
This study	Event-triggered repair of an executing plan	Pre-sampling task, appointment, vehicle, time-window and state audits	Yes: E-ALNS assignment, slack, and shadow-task reference	ICD task-chain repair plus action-level audit logs

The literature therefore leaves an integration problem: connecting static ICD planning, event-driven disturbances, hard feasibility checks, learning-based action ranking and simulation evaluation in one executable rescheduling loop. The dynamic policy is treated as one component of this system, rather than as a replacement for planning and feasibility control.

The paper makes three focused and verifiable contributions.

(1) It connects a static E-ALNS baseline to event-triggered local repair through candidate task chains, executed discrete-event transitions and dispatcher-readable logs. (2) It separates six explicitly enumerated ICD hard predicates from soft economic, stability, and role preferences, and audits both mask agreement and construction complexity. (3) It provides a documented, auditable evaluation comprising five-seed 120-episode learning curves

and paired tests, optimization and search baselines, a three-by-three robustness screen and reward-weight sensitivity. The contribution lies in system integration and accountable evaluation; masking, PPO, and self-attention themselves are established methods.

The remainder of this paper is organized as follows. Section 2 defines the operational setting and system boundary. Section 3 presents the event-driven Markov decision process (MDP), feasibility-audited action space, policy architecture, and simulation design. Section 4 reports the main results. Section 5 discusses system integration, deployment, and limitations. Section 6 concludes this study.

2. System Boundary and Problem Setting

2.1. ICD Truck Operations as a Dynamic Logistics System

The studied system includes an ICD, a port gate, customer or factory nodes, a truck fleet, and a pool of container-related tasks. A task may correspond to empty-container delivery, factory loading, heavy-container pickup, ICD temporary storage, port-bound gate-in actions, or empty-container recovery. Although these tasks can be represented as transport requests, their operational meanings differ. A port-bound gate-in task is usually governed by a hard cut-off time or appointment window, while a local empty-container move may have more flexibility. A dispatching system must therefore distinguish task roles rather than treating all moves as homogeneous jobs.

The simulated setting comprises one ICD, one aggregate port-gate interface, 50 factory/customer nodes in three industrial clusters, and a coordinated fleet of 60 trucks. The port-gate node is a scheduling abstraction, not a physical-lane count. All trucks belong to one resource pool; operational roles are soft reward preferences rather than hard eligibility constraints. Four independent contractors and a first-qualified-supplier preference are not modeled.

The system is dynamic because relevant information arrives during execution. A previously reserved task may become an executable real task when an appointment is released. A factory service delay may push a truck beyond its planned start time for the next task. A road congestion update may change the feasibility of a previously acceptable dispatch. These disturbances do not arrive at uniform intervals. They are event-triggered and state-dependent.

The model therefore uses event-driven decision epochs. At each event, the simulator freezes the operational state, enumerates candidate repairs, audits feasibility and passes the masked action set to the decision policy. After the selected action is executed, the simulator advances to the next event. Figure 1 summarizes the operational chain and identifies the buffer and local-repair points.

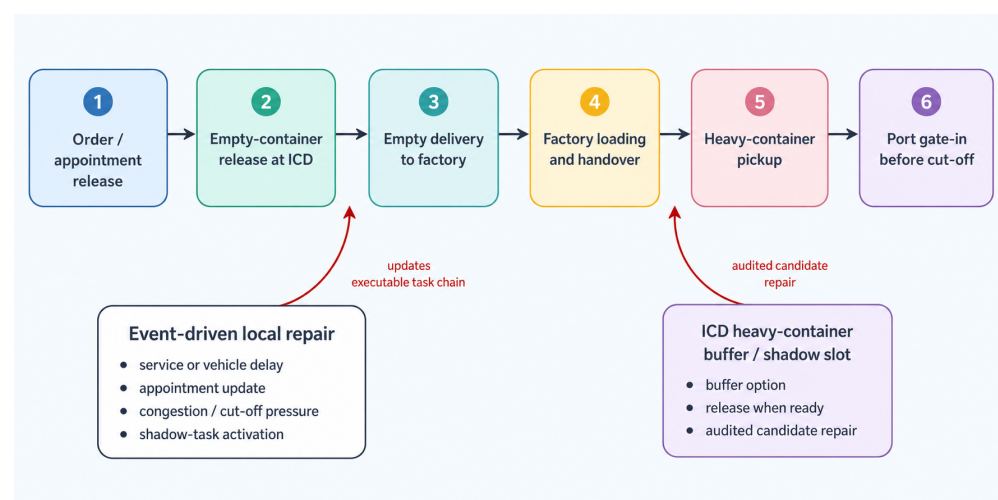


Figure 1. ICD container-handling process, buffer option, and event-driven local-repair points.

2.2. Static Baseline Plan and Dynamic Local Repair

The framework assumes that a baseline plan is available before the execution period. This plan can be generated by any appropriate static planning method, such as a mixed-integer programming model, a scheduling algorithm, a pickup-and-delivery heuristic, or an enterprise dispatching rule system [35,36]. In the present study, the baseline plan serves as the structured reference for runtime repair.

Operationally, the baseline plan provides three types of information for runtime repair: task chains for the available vehicle fleet, slack resources that represent time buffers between committed tasks, and shadow tasks that reserve capacity for uncertain but likely future port-bound demand.

The minimum scheduling data are divided into master data and event data. Master data comprise task identifiers and types, order-chain predecessors, origins and destinations, distances or travel times, service-time distributions, release times, appointment windows, cut-offs, priorities, vehicle roles, and the baseline vehicle assignment. Event data comprise the current clock, vehicle locations and available times, task and shadow-task status, appointment readiness, observed delays, and the active congestion state. Raw audit fields are retained separately from the neural embedding so that each recommendation can be checked against the business rules.

The time buffer is not imposed as one fixed allowance for every task. It is computed from each baseline chain as the difference between the latest feasible start or finish and the scheduled time after precedence and travel are considered. Across seeds 42–51, the resulting baseline slack has a mean of 1.479 h and a median of 1.446 h. This task-specific slack is consumed when delays occur and enters local-repair and stability calculations.

Dynamic rescheduling is treated as controlled editing around the baseline plan. The model does not recompute a full daily schedule after each disturbance. It repairs the plan locally, prioritizes high-risk tasks, and records the cost of deviation from the baseline. In this structure, the static layer provides stability and the dynamic layer absorbs disturbances.

2.3. Disturbance Types

The current simulation focuses on disturbance types that directly affect ICD dispatching feasibility.

The disturbance model covers task activation after appointment or demand release, service delays in loading, unloading or handover, vehicle-availability updates after completion or delay, traffic or travel-time changes caused by congestion, and cut-off pressure when a port-bound task approaches its latest feasible completion time. Their operational causes are, respectively, late customer or port confirmation; variability in factory loading, unloading or handover; propagation of a preceding task delay into vehicle availability; observed road congestion or travel-time updates; and the progressive loss of remaining slack before a port cut-off. These causes alter either task executability, vehicle availability, or the time-window feasibility test.

These events determine when the policy is called and which candidate actions are available.

3. Materials and Methods

3.1. Event-Driven MDP Formulation

The dynamic rescheduling process is modeled as an event-driven Markov decision process with an operational state, a feasible repair action set, an event-driven transition rule, a reward function, and a discount factor. This formulation follows the standard MDP view of sequential decision-making [37] and the RL view of policy learning through state-action-reward feedback [38]. In the experiments, the discount factor is 0.99.

Let V be the set of vehicles, J the set of atomic tasks, and K the ordered set of event-generated decision epochs. A task j in J is represented by the following fields:

$$(q_j, o_j, d_j, p_j, r_j, b_j, \text{Pred}(j), z_j(k), \rho_j(k), h_j, g_j)$$

A vehicle i in V is represented by current node, available time, accumulated mileage, preferred operational role, and remaining baseline slack:

$$(l_i(k), u_i(k), \text{mileage}_i(k), \text{role}_i, \sigma_i(k))$$

The static baseline plan assigns tasks to vehicle chains before execution and provides the reference assignment used in the stability term.

At event epoch k , the observed state is:

$$s_k = (X_k^J, X_k^V, X_k^E)$$

The event may represent shadow-task activation, vehicle delay, vehicle availability update, appointment release, congestion update, or cut-off pressure. Candidate repairs are generated from pending task–vehicle pairs and candidate-chain scores. The audited feasible set is:

$$A_k^{\text{feas}} = \{a \in C_k : m_k(a) = 1\}$$

The selected action updates task status, vehicle location, vehicle availability, reward components, response latency, and plan-deviation records. The transition is written as follows:

$$s_{k+1} = F(s_k, a_k, e_{k+1}, \xi_k)$$

The disturbance term denotes stochastic service-delay and congestion elements already embedded in the simulator. An evaluation trajectory stops when all real tasks are completed or failed, the 24 h horizon is reached, or the evaluation step cap is reached.

Table 2 summarizes the formal objects used in the formulation and their implementation role.

Table 2. Formal notation used in the event-driven rescheduling model.

Symbol or Object	Definition	Implementation Role
V	Vehicle set, indexed by i	Vehicle tokens and vehicle-state records
J	Atomic task set, indexed by j	Task tokens, task pool and status machine
e_k	Event observed at decision epoch k	Triggers state freeze and rescheduling call
s_k	State tuple (task, vehicle and environment fields)	Input to the encoder and feasibility audit
C_k	Candidate repair set before auditing	Candidate-chain list built from pending task–vehicle pairs
$m_k(a)$	Binary feasibility indicator for action a	Action mask applied before policy sampling
A_k^{feas}	Feasible action set after auditing	Only these actions can be selected by the policy
π_0	Static baseline plan	Slack reference and baseline assignment for SDS
$\sigma_i(k)$	Remaining slack of vehicle i	Stability and local-repair resource indicator
$z_j(k)$	Shadow-task status	Prevents inactive reserved tasks from being dispatched
$\rho_j(k)$	Appointment-readiness indicator	Prevents premature port-bound gate-in actions
$ V $	60 coordinated trucks in the main evaluation	One pooled fleet; no four-operator contracting preference is modeled
N_F	50 factory/customer nodes in three industrial clusters	Synthetic service-node network around one ICD
N_P	One aggregate port-gate interface	Scheduling node, not a claim of one physical gate lane
$ J $	Mean 330 real and 40 reserved shadow atomic tasks	Generated from direct, relay, import, and shadow orders
Buffer	Task-specific latest-feasible minus scheduled time	Mean 1.479 h and median 1.446 h over seeds 42–51

3.2. State Representation

The state retains auditable task, vehicle, and environment fields. Task fields include type, activation, urgency, remaining time, priority, service duration, origin and destination; vehicle fields include location, availability, slack, mileage, and role; environment fields include time, congestion, appointment readiness, and candidate-chain summaries.

Across the ten-seed baseline package, the generated plan contains on average 46.0 emergency and 37.1 urgent trunkline tasks. Factory service duration follows a normal distribution with mean 3.0 h and standard deviation (SD) 0.8 h, truncated to [1.5, 6.0] h. Task, vehicle, and environment tokens are embedded for ranking, while the raw fields remain outside the encoder for deterministic audit and explanation.

3.3. Candidate Repair Actions and Feasibility Auditing

The action space is built around local repairs rather than unrestricted global replanning. Candidate actions include task insertion, vehicle reassignment, local chain adjustment, and priority-based repair. In the implementation, these actions are represented through task–vehicle assignments, and candidate-chain objects. A candidate chain records the task level, vehicle index, task index, order index, slack, travel time, appointment status, cut-off risk, and priority score.

Before the policy samples an action, an action mask removes infeasible candidates. For a task–vehicle action $a = (i, j)$, the earliest physically reachable time is defined before release-time and appointment-readiness checks are applied. For appointment-bound tasks, the appointment-release value for task j at epoch k is recorded; if the task is not appointment-bound, this value is set to zero.

$$t_{ij}^{\text{arr}}(k) = u_i(k) + \tau(l_i(k), o_j, k)$$

$$t_{ij}^{\text{start}}(k) = \max \left\{ t_{ij}^{\text{arr}}(k), r_j, a_j^{\text{appt}}(k) \right\}$$

$$f_{ij}(k) = t_{ij}^{\text{start}}(k) + p_j$$

Waiting time is defined as follows:

$$w_{ij}(k) = t_{ij}^{\text{start}}(k) - t_{ij}^{\text{arr}}(k)$$

The mask can therefore be expressed as follows:

$$\begin{aligned} m_k(i, j) = & I[\text{status}_j(k) = \text{pending}] I[z_j(k) = 0] \\ & \times I[\text{Pred}(j) \text{ completed}] I[\rho_j(k) = 1 \vee j \text{ not appointment-bound}] \\ & \times I[f_{ij}(k) \leq b_j] \times I[\text{index}(i, j) \text{ valid}]. \end{aligned}$$

The implemented mask is complete only with respect to six declared hard predicates: valid task/vehicle indices, pending status, shadow activation, within-order predecessor completion, GATEIN appointment release, and reachable completion by the latest time. Role preference, economic cost, and baseline consistency are soft ranking terms. Operator contracts, loading-bay capacity, explicit swaps, and corrupted input records are outside the present mask; finite schema-valid input is an upstream precondition.

With J task atoms and V vehicles, the current implementation builds a pending-task list and, for each candidate task–vehicle pair, rescans the task collection when reconstructing within-order predecessor status. Its worst-case time is therefore $O(J^2V)$; the separate $O(J^2)$ pending-list membership scan is dominated when $V \geq 1$. Mask memory is $O(J_{\max} V_{\max})$. If pending membership is stored as a set and order-level predecessor completion is pre-indexed once per decision state, each pair can be checked in $O(1)$, reducing construction

to $O(JV)$ time with $O(J)$ auxiliary state. For the fixed 150×60 action array, the mask contains 9000 Boolean slots. Implementation consistency was tested with a separately coded predicate oracle at every visited state over five seeds and three disturbance regimes; mask time was measured separately from policy-scoring latency.

3.4. Reward Design

The reward function reflects four management objectives: cut-off safety, economy, plan stability and role consistency. For a feasible action $a = (i, j)$, let f_{ij} be the simulated finish time, b_j the latest feasible finish time, d_{ij} the empty travel distance from the vehicle's current node to the task origin, and w_{ij} the waiting time before service. The immediate reward is:

$$R_k = 0.70R_{\text{safe}} + 0.15R_{\text{econ}} + 0.15R_{\text{stab}} + R_{\text{role}}$$

The safety term is applied most strongly to port-bound heavy-container and gate-in tasks. For tasks of type GATEIN or PICK_L_ICD, slack is defined as follows:

$$s_{ij} = b_j - f_{ij}$$

$$R_{\text{safe}} = \begin{cases} -140, & s_{ij} < 0, \\ -24(0.25 - s_{ij}) - 4, & 0 \leq s_{ij} < 0.25, \\ -8(0.75 - s_{ij}) + 2, & 0.25 \leq s_{ij} < 0.75, \\ \min(12, 3 + 2.5s_{ij}), & s_{ij} \geq 0.75. \end{cases}$$

For non-port-bound supporting tasks, $R_{\text{safe}} = 0.4$. A cut-off miss is also recorded as a violation and the affected task is marked failed in the simulator.

The economic term is normalized from travel and waiting costs:

$$R_{\text{econ}} = -\frac{3.5d_{ij} + 120w_{ij}}{1000}$$

The stability term compares the selected vehicle with the baseline vehicle in π_0 . If the assignment matches π_0 , $R_{\text{stab}} = 0.6$. If the task exists in π_0 but is reassigned to another vehicle, $R_{\text{stab}} = -0.6$. If the task was not present in the baseline, $R_{\text{stab}} = -0.1$. When no baseline assignment is available, $R_{\text{stab}} = 0$.

$$R_{\text{role}} = 0.5I[\text{preferredrole matchestaskrole}] - 0.35I[\text{currentroleswitches}]$$

This term discourages avoidable switching between trunkline and branch-line work. In the no-mask ablation only, an invalid sampled action receives a penalty of -120 .

The nominal weights (0.70, 0.15, 0.15) encode an explicitly declared priority: cut-off safety receives the majority weight, while economy and plan stability are secondary and symmetric. The weights sum to one and apply only after hard feasibility screening; the role term remains separate. They are operational calibration choices, not derived optimal constants.

Economic reward denotes normalized empty-travel and waiting cost, not payment or contracting preference; consistency denotes assignment and role stability relative to the baseline. A local simplex screen retrains the policy under safety-heavy (0.80,0.10,0.10), nominal (0.70,0.15,0.15), balanced (0.60,0.20,0.20), economy-heavy (0.60,0.30,0.10), and stability-heavy (0.60,0.10,0.30) profiles for 60 episodes over seeds 42–44; the final 30 episodes use a fixed 100-active-atom/60-vehicle local-repair window within the 150-task-slot tensor (the remaining 50 task slots are padding). Because profiles change the reward scale, comparisons use α_{online} , mileage, schedule deviation score (SDS), and violations rather than cross-profile total reward.

3.5. Transformer-PPO Architecture

The policy uses an Actor–Critic architecture trained with proximal policy optimization (PPO).

The policy encoder is evaluated in two configurations: an MLP baseline that flattens or aggregates token features and environment summaries, and a Transformer-small encoder that projects task and vehicle tokens to a shared representation, processes them with self-attention [28] and fuses them with environment features.

The compact Transformer-small model uses $d_{\text{model}} = 64$, two attention heads, and one encoder layer. The Actor outputs logits over candidate task–vehicle actions. The Critic estimates the state value. Action masking is applied to Actor logits before sampling.

PPO is selected because it limits destructive policy updates through a clipped surrogate objective and is widely used for stable policy-gradient learning [27]. With probability ratio $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, the clipped surrogate objective is:

$$L^{\text{CLIP}}(\theta) = E_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon) \hat{A}_t \right) \right]$$

where $\hat{A}_t$ is the advantage estimate and $\varepsilon = 0.2$ in the experiments.

3.6. Curriculum Learning and Training Configuration

Curriculum learning is used to reduce early training failure. The system begins with smaller and simpler task sets, then increases disturbance complexity and scale after the recent training window satisfies progression criteria. This helps the policy learn basic feasibility and dispatching preferences before facing denser event combinations.

Table 3 summarizes the simulation and training protocol, making the synthetic environment, baseline plan, and learning budget explicit.

Table 3. Simulation and training protocol.

Component	Setting	Value Used in the Evaluation
System scale	Orders and fleet	Static source: 100 orders/60 vehicles; dynamic evidence: a fixed 100-active-atom local-repair window within the 150-task-slot tensor.
Order mix	Direct, relay, import and shadow orders	30, 40, 20 and 10 orders
Spatial network	ICD, port and factories	ICD-port distance 80 km; 50 factories around three industrial clusters
Travel speed	Normal and congested speed	60 km/h and 30 km/h
Time horizon	Simulation horizon	24 h, with 18:00 as time zero
Loading time	Factory loading duration	Normal (3.0, 0.8) h, truncated to 1.5–6.0 h
Cut-off waves	Vessel wave hours	12, 18, 24 and 30 h with normal(wave, 1.5) h perturbation
Direct-order release	Tight release window	Cut-off minus uniform(4, 7) h
Other-order release	General release window	Uniform(0, max(cut-off—10, 1)) h
Port appointment	Gate-in release and close	Release = max(0, b_j —min(6, max(2, 0.35 b_j))); close = b_j
Disturbance A	Shadow-task activation	$t = 16$ h, activating 5 shadow tasks
Disturbance B	Vehicle delay	$t = 20$ h, vehicle 12, delay normal(1.0, 10/60) h truncated at zero
Disturbance C	Congestion	$t = 22.5$ h for 2 h
Baseline plan	Static reference	E-ALNS, 80 iterations, same seed as the instance
Screening benchmark	Online evaluation	Seeds 42–51; 40 training episodes for learned policies; 128 event steps
PPO update	Learning hyperparameters	$\gamma = 0.99$, clip = 0.2, $\lambda_{\text{GAE}} = 0.95$, lr = 0.0003, epochs = 4, batch = 64
Transformer-small	Encoder hyperparameters	$d_{\text{model}} = 64$, heads = 2, layers = 1, feedforwarddim = 128

Table 3. Cont.

Component	Setting	Value Used in the Evaluation
Robustness scale	Atomic tasks/fleet	Approximately 40/20, 70/40 and 100/60; E-ALNS baseline rebuilt for every scale and seed
Robustness disturbance	Low/nominal/high	2/5/8 shadow activations; mean truck delay 0.5/1.0/2.0 h; congestion duration 1/2/4 h
Online search baseline	Rolling-horizon lookahead (RH-2)	Two first-stage candidates, two executed lookahead steps, event/periodic reoptimization, no training
Long-training confirmation	Paired seeds and budget	Seeds 42–46; three-stage warm-up plus 60 episodes on a 100-atom/60-vehicle window; held-out evaluation every 20 episodes
Mask audit	Scope and sample	Five seeds $\times$ three disturbances; 552 states; 4,968,000 action cells
Reward sensitivity	Profiles and budget	Five profiles; seeds 42–44; 30 warm-up plus 30 episodes on a 100-atom/60-vehicle window

3.7. Discrete-Event Simulation Environment

The evaluation uses a discrete-event simulation environment representing the port-ICD-customer system. Discrete-event simulation is appropriate because truck completions, appointment releases, delays and congestion updates are naturally represented as event-calendar updates rather than continuous-time differential processes [39,40]. The simulator maintains vehicle states, task states, appointment states, event logs, accumulated cost, and plan deviation. After each selected action, the simulator updates the execution trajectory and advances to the next event, so metrics are computed from executed trajectories rather than from isolated one-step choices.

Table 4 defines the reported metrics. Two distinctions are important. First, α_{train} and α_{online} are computed on different trajectories and should not be compared across tables as if they were the same population statistic. Second, stability is reported in two forms: training-tail stability for learned policies and cross-seed reward stability for the online benchmark.

Table 4. Metric definitions and interpretation.

Metric	Formula or Computation	Interpretation	Direction
Training proxy α_{train}	Tail mean of episode-level α_{train} during training	Used only to compare learned policies under the same training protocol	Higher is better
Online proxy α_{online}	Completed counted real atomic tasks divided by all counted real atomic tasks at evaluation end	Counted real atomic tasks include initially non-shadow tasks and shadow tasks activated before termination; not a field compliance rate	Higher is better
Reward	Training tables report tail mean reward; online tables report total trajectory reward	Multi-objective value induced by the reward function	Higher is better
Selector/update latency	Policy/rule scoring plus assignment-state update after mask construction	Online computational response time	Lower is better
SDS	Mean mismatch ratio between executed task–vehicle assignments and π_0 among compared tasks	Plan deviation from the baseline schedule	Lower is better
Training-tail stability	1—CV of tail training rewards, clipped by finite-sample behavior	Stability of learned-policy training tail	Higher is better
Cross-seed stability	1—CV of online total reward across seeds	Robustness of online performance across scenario seeds	Higher is better
Invalid action rate	Invalid sampled actions divided by executed decision steps	Feasibility failure indicator, mainly for the no-mask ablation	Lower is better

All results are summarized as means and standard deviations across seeds where applicable.

3.8. Experimental Scenarios, Baselines and Reporting Protocol

The controlled simulation evaluated four aspects of the framework: the scope and cost of hard-feasibility screening, the relational encoder effect, the behavior of learned and non-learned rankings under a fixed executed scenario, and sensitivity to scale, disturbance, and reward weights. This is not a production-readiness test. Because no proprietary ICD event logs, dispatcher decision histories, or live operational key performance indicators (KPIs) are used, the results establish internal behavior within the declared simulator rather than external validity at an operating depot.

The principal baseline is an MLP encoder with the same downstream PPO structure, masking logic, curriculum settings and random seed set as the Transformer-small policy. This baseline isolates the effect of the state encoder. Because both models receive the same feasibility mask, any observed difference is not caused by one model being allowed to take a different class of actions.

The ablation design includes a no-mask configuration to test the feasibility layer directly. This variant is used only as a simulation control. In live operations, removing the mask would allow illegal dispatch recommendations. A no-curriculum configuration is also included to test whether gradual scenario progression improves learned dispatching behavior.

The ten-seed screening experiment compared six policies over seeds 42–51 after 40 learning episodes. A separate convergence confirmation trained matched MLP and Transformer-PPO models for 120 episodes over seeds 42–46; three 20-episode warm-up levels are followed by 60 episodes on a fixed 100-active-atom/60-vehicle local-repair window. Both models were evaluated every 20 episodes on the same fixed nominal window per seed. The fixed evaluation avoids interpreting curriculum-level changes as convergence. Rule and chain baselines were evaluated on the same five instances.

Figure 2 presents the controlled event timeline. The baseline is initialized at $t = 0$ h, followed by stochastic task and appointment releases and service completions. Five shadow tasks activate at $t = 16$ h; a one-hour delay affects only vehicle 12 at $t = 20$ h; a two-hour congestion window begins at $t = 22.5$ h; and the scenario ends at $t = 24$ h or the 128-step evaluation cap. Thus, 20 h denotes the event time, not the duration of the delay.



Figure 2. Controlled event timeline for the nominal 24-h scenario.

For the five-seed long run, paired mean differences used 20,000-resample bootstrap 95% intervals, two-sided Wilcoxon signed-rank tests, and paired Cohen's d_z ; episode 120 was also compared with episode 100 as a limited plateau diagnostic. These tests are reported jointly because they target different aspects of the paired distribution. The two-step rolling-horizon lookahead comparator (RH-2) used the same audited feasible set, cloned the executed discrete-event simulation (DES) state at disturbance/periodic triggers, and scored two-step candidate continuations without learned parameters.

The robustness extension used a three-by-three design across task/fleet scales of approximately 40/20, 70/40, and 100/60 and low, nominal, and high disturbance levels over seeds 42–44. The three disturbance levels activated 2/5/8 shadow tasks, applied a 0.5/1.0/2.0 h mean delay to one truck, and used 1/2/4 h congestion windows.

The event times were 20/16/10 h for shadow release, 22/20/12 h for the truck delay, and 23/22.5/16 h for congestion. A scale-specific E-ALNS baseline was regenerated for every seed. Transformer-PPO was trained for 30 episodes per seed using travel speeds of 60 km/h normally and 30 km/h during congestion, then compared with the rule and chain baselines in all nine cells; RH-2 was also evaluated in the large-scale nominal cell. This design measures interpolation within the modeled workload and disturbance ranges. It does not cover a new depot topology, task ontology, constraint set, or empirical ICD distribution, so it is a controlled stress test rather than field validation.

The reporting protocol distinguishes training proxies from operational KPIs. The training proxy α_{train} is computed on learning trajectories. The online proxy α_{online} is computed under the simulator's task-status accounting at the end of a stress-test trajectory. Its numerator is the number of completed counted real atomic tasks, and its denominator is the number of counted real atomic tasks at evaluation end. Counted real atomic tasks include initially non-shadow tasks and shadow tasks activated by the event calendar before termination. For example, in seed 42 under the earliest-cutoff rule, 29 completed counted real atomic tasks are divided by 335 counted real atomic tasks after shadow activation. The resulting value is therefore a normalized stress-test proxy under a 24 h horizon and 128-step cap, not an enterprise completion rate. A field compliance rate would require a validated data pipeline, real task distributions, operational acceptance records, and enterprise-specific service definitions.

3.9. Implementation Details and Baseline Construction

The candidate-chain heuristic and the learned policies used the same task–vehicle candidate set and the same feasibility mask, except for the no-mask diagnostic ablation. For each feasible candidate c , the chain generator computes a base score:

$$S_c = B_{\text{trunk}} + B_{\text{branch}} + B_{\text{backhaul}} + B_{\text{service}} + B_{\text{enable}} + B_{\text{continuation}} \\ + 6S_{\text{slack}} + 4U + 1.8h_j + 2.5M_{\pi_0} + 2.2M_{\text{pref}} + 1.2M_{\text{current}} \\ - 0.35\tau - 0.40\text{idle}.$$

where U is inverse deadline remaining, h_j is task priority, M_{π_0} indicates a baseline-vehicle match, M_{pref} and M_{current} indicate preferred-role and current-role matches, τ is travel time, and idle is vehicle availability delay. The slack score is negative when slack is below 0.25 h and otherwise increases up to a capped positive value. The online candidate-chain heuristic ranks candidates by priority tier first and then by the effective score $S'_c = S_c - P_c$, where P_c includes a role-switch penalty, repeated-vehicle penalties, recent-vehicle penalties, work-in-process penalties, and a trunkline-reserve penalty. The heuristic selects the highest positive effective score, with conflicts removed by task, vehicle, and order. The earliest-cutoff rule uses the same feasible candidates but sorts them by cut-off level, appointment readiness, trunkline stage, latest time, travel time, baseline match, and vehicle ID.

The static baseline plan π_0 is generated by E-ALNS. The initial solution is a greedy priority and time-window construction. The neighborhood search then applies four operator families: relay recombining, street-turn discovery, triangular backhaul, and temporal slack shifting. The acceptance rule uses simulated annealing with $T_0 = 100$, cooling factor 0.995, and terminal temperature 1.0; operator weights are updated every 50 iterations, and the evaluation uses 80 iterations for each baseline plan. Table 5 summarizes implementation constants and baseline-plan diagnostics over the ten benchmark seeds.

Table 5. Implementation details and baseline-plan diagnostics.

Item	Value or Formula	Notes
Discrete action space	$150 \times 60 = 9000$	Invalid or padded actions are removed by the mask
Token dimensions	Task 8, vehicle 8, environment 10	Padding caps are 150 tasks and 60 vehicles
Candidate-chain selection	priority tier, then $S'_c = S_c - P_c$	Same feasible candidate set as masked learned policies
PPO optimizer	Adam, lr = 0.0003	PPOclip = 0.2, $\gamma = 0.99$, $\lambda_{\text{GAE}} = 0.95$
PPO loss terms	clipped policy loss + 0.5 Huber value loss—0.01 entropy	Gradient norm clipped at 0.5
Curriculum levels	20/10, 40/20, 70/40, and 100/60 tasks/vehicles	Progression uses recent α_{train} and reward-window criteria
E-ALNS operators	relay recombining, street-turn discovery, triangular backhaul, temporal slack shift	80 iterations in the evaluation
E-ALNS objective	initial $42,188.85 \pm 865.11$, best $34,294.24 \pm 1038.08$	Mean $\pm$ SD over seeds 42–51
Baseline travel distance	$10,239.87 \pm 254.07$ km	Replayed over static vehicle chains
Baseline scheduled atomic tasks	330.0 real tasks and 40.0 shadow atomic tasks	Mean over seeds 42–51
Baseline slack	mean 1.479 h, median 1.446 h	Slack values attached to static chains
Planned gate-in tasks	90.0	Mean over seeds 42–51
Baseline cut-off risk tiers	46.0 emergency, 37.1 urgent, 66.9 normal trunkline tasks	Based on generated trunkline deadline percentiles
E-ALNS runtime	1.75 ± 0.76 s	Local single-run runtime for baseline construction

Optimization-Based Calibration and Online Comparison

A separate small-instance mixed-integer linear programming calibration used the same task-atom, precedence, time-window, vehicle-continuity, and cost definitions. A 10-order, 5-vehicle, 37-task instance reached an optimal solution; in a scale stress test, the 5-order and 10-order instances reached optimal status, whereas the 15-order and 20-order instances did not reach solved status within their stated limits. These results verify closure of the modeled business constraints and illustrate the rapid growth of exact-solution time.

At the 100-order, 60-vehicle static-planning scale, E-ALNS constructs the baseline schedule. On one representative instance, it reduced the static objective from 42,213.29 to 34,421.22 while retaining 89.86 min mean slack. A comparison adaptive-neighborhood configuration attained a lower nominal ledger value but retained only 18.23 min mean slack and placed 93.38% of task connections below 15 min. These instance-level values are distinct from the ten-seed summary in Table 5. The static results motivate the baseline layer but do not establish online superiority. RH-2 provides a matched event-triggered search benchmark on the same audited candidate set and executed DES state, but its two-step horizon does not represent the broader optimization frontier. Rolling MILP, CP-SAT, model-predictive control and deeper online ALNS could use longer lookahead or different neighborhoods. A fair comparison would require matched stopping rules, warm starts, state-transition logic, and end-to-end latency accounting. Because those methods are outside the present computational study, the later matched rolling-horizon benchmark is interpreted as evidence of complementarity with bounded search, not a general comparison with advanced online optimization.

3.10. Experimental Traceability and Audit Logs

The simulation records audit logs at two levels. The first level records learning and simulation configuration: random seed, episode count, encoder configuration, PPO parameters, curriculum level, reward weights, and simulation parameters. The second level records operational decisions: event time, state snapshot, candidate actions, mask status, selected action, reward decomposition, task completion state, and plan deviation.

The audit structure serves both experimental traceability and operational review. Learning-based dispatching results can be sensitive to random seeds, reward scaling, and environment assumptions, as emphasized in deep reinforcement learning reproducibility discussions [41]. For depot managers, the same log explains why a recommendation was generated, which constraints removed alternatives, and how much slack or baseline deviation the selected repair consumed.

To support auditability, the authors have retained internal experiment records comprising seed lists, configuration files, training logs, per-seed result files, simulation-event definitions, and figure-generation scripts. These records support internal traceability of the reported tables and figures. The resulting run logs and intermediate files have been retained as internal research records and are not publicly archived.

4. Results

4.1. Event-Driven Rescheduling Framework

The model produces a closed rescheduling loop: event detection, state freezing, token encoding, candidate-chain generation, feasibility auditing, policy selection, execution update, and log recording. The sequence separates operational feasibility from learned or rule-based ranking. The policy/rule selector ranks feasible options; the audit layer decides which options are allowed.

Figure 3 illustrates this loop, including the static baseline schedule, runtime event calendar, task–vehicle–environment state encoder, candidate-chain generator, feasibility mask, policy/rule repair selector, and the executed action and audit log.

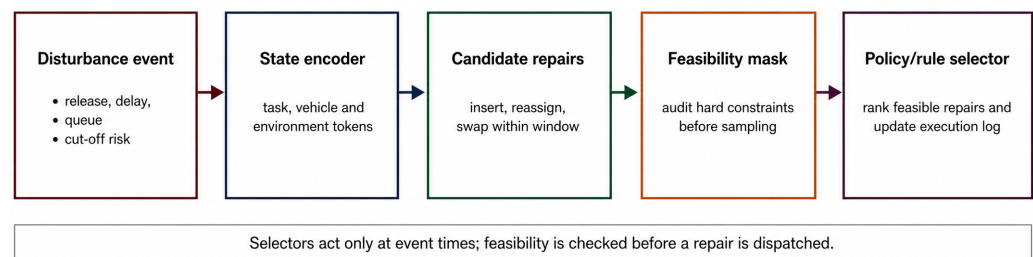


Figure 3. Event-driven and feasibility-audited ICD rescheduling framework.

Unlike a pure routing optimizer, the model treats dynamic rescheduling as controlled repair rather than complete rerouting. The policy acts only when the event calendar indicates that a decision is operationally meaningful.

4.2. Feasibility Masking Is a Core Safety Mechanism

The independent audit covered 552 visited states and 4,968,000 task–vehicle action cells across seeds 42–46 and low, nominal and high disturbance regimes. It found 0 false-positive and 0 false-negative cells relative to the separately coded oracle for the six implemented predicates. This is empirical agreement over tested states, not a proof that the mask contains every constraint of a real ICD.

In a controlled ablation using the same simulator and training configuration, removing action masking caused severe deterioration. The complete configuration achieved a mean reward of 25.09 and a training completion proxy of 0.7958, whereas the no-mask configuration reduced mean reward to −186.85 and the completion proxy to 0.0938. The result indicates that action masking is a boundary condition for applying RL to constrained dispatching, not a minor implementation detail.

Mask construction averaged 25.6 ms and had a 30.4 ms 95th percentile on the audited CPU runs. This preprocessing cost is materially larger than the neural scoring time and

is reported separately. The concrete steps needed to reduce the current $O(J^2V)$ list scan to $O(JV)$ construction are to store pending-task membership in a set and cache order-level predecessor summaries.

4.3. Five-Seed Long-Training Confirmation

Matched MLP and Transformer-PPO models were trained for 120 episodes on seeds 42–46; the final 60 episodes used a fixed 100-active-atom/60-vehicle local-repair window. This fits the 150-task-slot tensor with 50 padding slots; it is not simultaneous coverage of all task atoms expanded from the 100 source orders. Figure 4 reports the episode-120 held-out comparison with matched deterministic baselines, and Table 6 reports the corresponding training-tail and held-out metrics. The action mask, curriculum, instance, baseline plan, and evaluation seed are paired within each entry.

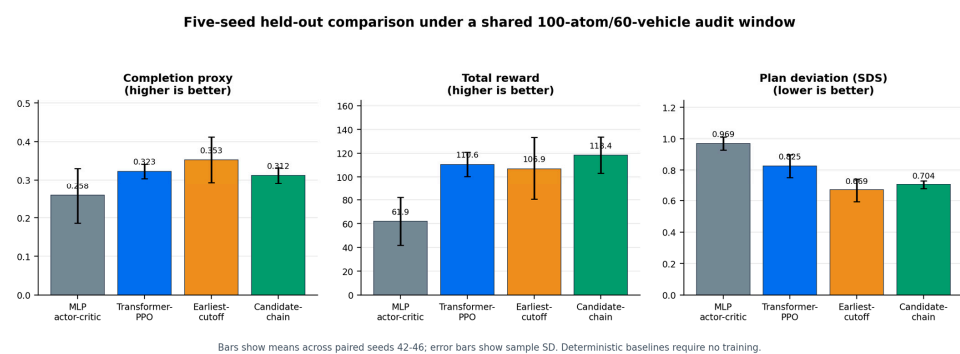


Figure 4. Five-seed held-out comparison at episode 120 under the shared 100-active-atom/60-vehicle audit window. Bars show means and error bars show sample SD across paired seeds 42–46; lower SDS is better.

Table 6. Five-seed 120-episode long-training confirmation and matched deterministic baselines (mean $\pm$ sample SD).

Model	Tail α_{train}	Tail Reward	Held-Out α_{online}	Held-Out Reward	Held-Out SDS
MLP actor–critic	0.2597 $\pm$ 0.0664	51.26 $\pm$ 24.06	0.2581 $\pm$ 0.0717	61.87 $\pm$ 20.21	0.969 $\pm$ 0.042
Transformer-PPO	0.2720 $\pm$ 0.0222	85.23 $\pm$ 8.75	0.3226 $\pm$ 0.0186	110.58 $\pm$ 10.24	0.825 $\pm$ 0.073
Earliest-cutoff rule	—	—	0.3527 $\pm$ 0.0591	106.94 $\pm$ 26.35	0.669 $\pm$ 0.075
Candidate-chain heuristic	—	—	0.3118 $\pm$ 0.0201	118.42 $\pm$ 15.28	0.704 $\pm$ 0.028

At episode 120, Transformer-PPO minus MLP was +0.0645 for α_{online} (bootstrap 95% confidence interval (CI) +0.0172 to +0.1139; Wilcoxon $p = 0.0625$), and +48.71 for reward (95% CI +35.13 to +63.27; $p = 0.0625$). On the same five held-out instances, the earliest-cutoff rule achieved $\alpha_{\text{online}} = 0.3527$ and reward = 106.94, while the candidate-chain heuristic achieved $\alpha_{\text{online}} = 0.3118$ and reward = 118.42. Thus, Transformer-PPO had higher sample means than the matched MLP and positive bootstrap mean-difference intervals in this finite-budget experiment, but the exact two-sided Wilcoxon tests did not cross the 0.05 threshold and the learned model did not dominate both deterministic baselines. With only five paired seeds, effect estimates and uncertainty are emphasized over binary significance.

The five-seed, 120-episode run strengthens the finite-budget evidence but does not establish universal RL convergence. Curriculum learning is non-stationary, and the stationary held-out checkpoints remain variable. The result is therefore treated as finite-budget confirmation rather than an asymptotic convergence proof.

4.4. Fixed-Scenario Convergence Across 20–120 Episodes

Figure 5 plots held-out α_{online} and reward at 20-episode checkpoints, with 95% t-interval bands over the five paired seeds. Table 7 reports the same means. Because every checkpoint is evaluated on a fixed nominal 100-atom local-repair window per seed, changes are not caused by a moving curriculum evaluation set.

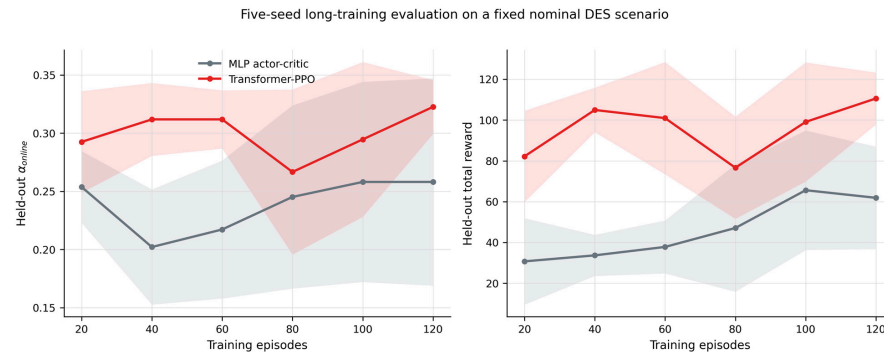


Figure 5. Five-seed fixed-scenario evaluation across 20–120 training episodes. Lines show means and shaded bands show 95% t intervals.

Table 7. Held-out checkpoint means across five paired seeds.

Episodes	MLP α_{online}	Transformer α_{online}	MLP Reward	Transformer Reward
20	0.2538	0.2925	30.72	82.18
40	0.2021	0.3118	33.66	104.94
60	0.2172	0.3118	37.80	100.95
80	0.2452	0.2666	47.09	76.62
100	0.2581	0.2946	65.60	99.08
120	0.2581	0.3226	61.87	110.58

From episode 100 to 120, the Transformer change was +0.0279 in α_{online} (95% CI +0.0043 to +0.0688; $p = 0.1250$) and +11.50 in reward (95% CI −0.62 to +27.05; $p = 0.1875$). A non-significant late change would be only limited plateau evidence; it is not treated as proof of convergence.

4.5. Ten-Seed Benchmark and Multi-Objective Trade-Off

The ten-seed benchmark extends the comparison from encoder choice to online behavior. Table 8 reports rule, heuristic, MLP, Transformer, no-mask, and no-curriculum controls under a common full-instance evaluation. Because the learned policies use a 40-episode screening budget, these results are interpreted as robustness and trade-off evidence rather than final converged performance.

Table 8. Ten-seed online evaluation across rules, heuristics, learned policies, and ablations.

Method	Seeds	Episodes	α_{online}	Reward	Selector/Update Latency (ms)	SDS	Stability	Invalid Rate
Earliest-cutoff rule	10	0	0.0890 ± 0.0058	85.06 ± 9.34	30.89 ± 13.97	0.173 ± 0.065	0.890	0.000
Candidate-chain heuristic	10	0	0.0866 ± 0.0048	106.75 ± 11.24	65.25 ± 25.33	0.714 ± 0.035	0.895	0.000
MLP actor-critic	10	40	0.0618 ± 0.0110	32.65 ± 13.86	1.47 ± 0.38	0.918 ± 0.086	0.576	0.000
Transformer-PPO	10	40	0.0818 ± 0.0103	73.90 ± 17.10	32.71 ± 11.99	0.542 ± 0.116	0.769	0.000
Transformer-PPO (no mask)	10	40	0.0003 ± 0.0009	−266.04 ± 11.95	33.16 ± 11.90	0.100 ± 0.300	0.955	0.499
Transformer-PPO (no curriculum)	10	40	0.0546 ± 0.0130	52.18 ± 13.85	33.30 ± 11.34	0.790 ± 0.092	0.735	0.000

Latency in Table 8 is selector/policy scoring plus the assignment-state update after a feasibility mask has been produced. Mask construction is profiled separately in Section 4.2. Neither measure includes external data acquisition or network I/O; deployment end-to-end latency is their sum plus integration overhead.

Table 8 gives four observations. First, the no-mask ablation fails operationally; approximately half of its sampled actions are invalid, and its α_{online} collapses to near zero. Its high cross-seed reward stability is not desirable stability; it means that the ablation fails consistently. Second, the full Transformer-PPO policy improves on the MLP actor–critic in α_{online} , reward, plan deviation, and cross-seed stability, but it is slower than the MLP because attention is evaluated over task and vehicle tokens. Third, the earliest-cutoff rule is a strong baseline. Against this rule, Transformer-PPO does not show an online advantage under the current 40-episode budget; the rule has higher α_{online} , total reward, and cross-seed stability, with lower SDS and similar latency. The learned policy is mainly supported by its improvement over the MLP and ablations and by potential extensibility, not by current dominance over the rule. Fourth, the candidate-chain heuristic obtains the highest reward and slightly higher α_{online} than Transformer-PPO, but it does so with higher response latency and larger plan deviation. The evidence therefore supports a feasibility-audited learning framework and a Transformer-over-MLP representation effect; it does not show that Transformer-PPO dominates deterministic rules or candidate-chain heuristics under the present budget.

Table 9 reports the component-level effect of replacing the full Transformer-PPO policy with each learned ablation. Positive $\Delta\alpha_{\text{online}}$ means that the full Transformer-PPO policy has a higher online proxy. Negative ΔSDS means that the full policy has lower plan deviation than the comparator. The last column is deliberately labeled as a training-tail stability metric; it is computed from learned-policy training tails and is not the cross-seed online stability reported in Table 8.

Table 9. Ablation effects relative to the full Transformer-PPO policy.

Comparison	$\Delta\alpha_{\text{online}}$	ΔSDS	Δ Latency (ms)	Δ Training-Tail Stability Metric
Transformer-PPO vs. MLP actor–critic	0.0200	−0.3766	31.24	0.1435
Transformer-PPO vs. Transformer-PPO (no mask)	0.0815	0.4417	−0.45	0.1185
Transformer-PPO vs. Transformer-PPO (no curriculum)	0.0272	−0.2486	−0.59	0.1967

Table 10 reports paired ten-seed statistical comparisons. Differences are Transformer-PPO minus comparator. Confidence intervals are paired bootstrap 95% intervals, and p -values are two-sided Wilcoxon signed-rank tests. To improve readability, Table 11 separates the α_{online} effect sizes and interpretation from the numerical confidence-interval table.

Table 10. Paired ten-seed statistical comparison for Transformer-PPO minus comparator.

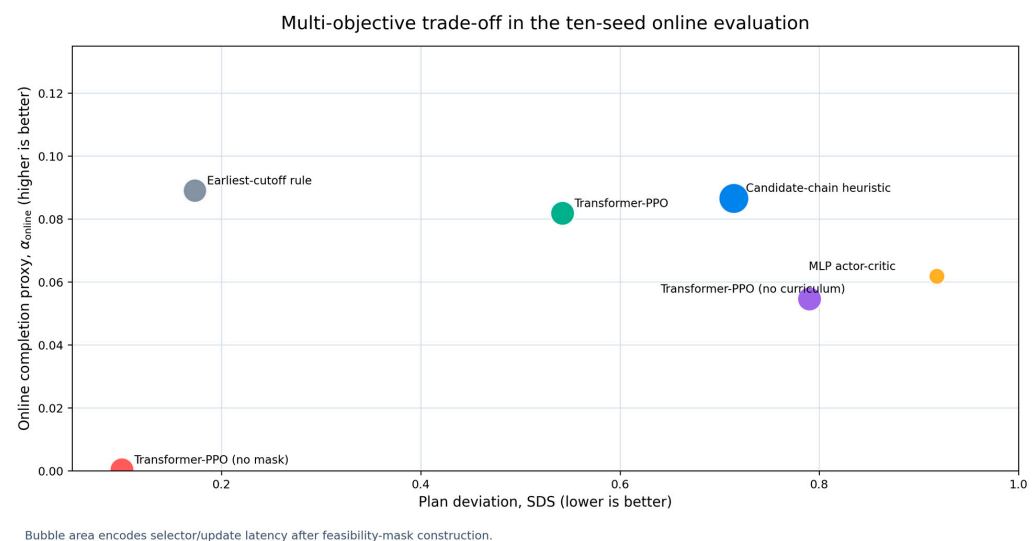
Comparator	$\Delta\alpha_{\text{online}}$	Δ Reward	ΔSDS	Δ Latency (ms)
MLP actor-critic	+0.0200 [0.0090, 0.0305], $p = 0.0137$	+41.25 [31.09, 51.65], $p = 0.0020$	−0.3766 [−0.4386, −0.3144], $p = 0.0020$	+31.24 [25.11, 39.56], $p = 0.0020$
Earliest-cutoff rule	−0.0072 [−0.0143, −0.0006], $p = 0.1602$	−11.16 [−25.19, 1.17], $p = 0.2324$	+0.3685 [0.2838, 0.4611], $p = 0.0020$	+1.83 [−4.80, 5.61], $p = 0.0840$
Candidate-chain heuristic	−0.0048 [−0.0123, 0.0012], $p = 0.3281$	−32.85 [−44.64, −22.18], $p = 0.0020$	−0.1728 [−0.2493, −0.0961], $p = 0.0059$	−32.53 [−43.67, −23.56], $p = 0.0020$
Transformer-PPO (no mask)	+0.0815 [0.0743, 0.0869], $p = 0.0020$	+339.94 [323.74, 354.99], $p = 0.0020$	+0.4417 [0.2509, 0.5744], $p = 0.0039$	−0.45 [−1.43, 0.56], $p = 0.4316$
Transformer-PPO (no curriculum)	+0.0272 [0.0152, 0.0370], $p = 0.0059$	+21.72 [7.31, 34.46], $p = 0.0273$	−0.2486 [−0.3179, −0.1729], $p = 0.0020$	−0.59 [−1.80, 0.47], $p = 0.6250$

Table 11. Effect-size interpretation for α_{online} and benchmark conclusions.

Comparator	Cohen's d_z for α_{online}	Main Interpretation
MLP actor–critic	1.087	Transformer improves over the MLP but is slower
Earliest-cutoff rule	−0.608	The rule remains the strongest low-deviation baseline
Candidate-chain heuristic	−0.411	The heuristic gives higher reward; Transformer reduces SDS and latency
Transformer-PPO (no mask)	7.464	Action masking is essential for feasible behavior
Transformer-PPO (no curriculum)	1.460	Curriculum improves α_{online} and reward in this setting

The paired bootstrap estimates the mean difference, whereas Wilcoxon tests the signed rank distribution. They need not return the same binary conclusion, especially with small, heterogeneous seed samples. The Table 8 discrepancy is therefore reported as inferential sensitivity rather than resolved in favor of one test; neither result supports a blanket superiority claim.

Figure 6 plots the same comparison as a multi-objective trade-off. The horizontal axis represents plan deviation; the vertical axis represents α_{online} ; and bubble area represents response latency. The desirable region is the upper left, but the plot also shows why practical deployment requires policy governance. The earliest-cutoff rule is the strongest low-deviation point in the reported metrics. The candidate-chain heuristic delivers the highest reward but at higher latency and deviation. Transformer-PPO is best interpreted as a learned feasible-action ranking module that improves over the MLP and ablations, not as the current winner against the rule baseline.

**Figure 6.** Multi-objective trade-off across ten seeds. The x -axis reports plan deviation, the y -axis reports α_{online} , and bubble area encodes selector/update latency after feasibility-mask construction.

4.6. Operational Meaning of Candidate Chains

The candidate-chain layer translates low-level task–vehicle action indices into dispatcher-readable repair options. A candidate may represent a trunkline gate-in, a factory heavy-container pickup, a branch-line empty-container delivery, an empty-container backhaul, or a shadow-task activation repair. Each candidate records the task type, priority, vehicle, slack, appointment state, predicted travel time, and deviation from the baseline plan.

This representation supports explainability. A dispatcher can inspect why an action is recommended; it may protect a port-bound cut-off task, reduce future empty travel, preserve a trunkline vehicle, or consume slack in a controlled manner. In contrast, an unstructured action ID would be difficult to interpret and audit.

4.7. Discrete-Event Simulation Closes the Evaluation Loop

The discrete-event simulation loop evaluates the policy on executed trajectories. Figure 7 shows how event-triggered states, process updates, model actions, and repair assessment are connected in the simulator. This matters because a locally attractive action may create downstream infeasibility. The simulator updates vehicle availability, task completion, slack consumption, and plan deviation after each selected action. It then generates the next decision event from operational conditions.

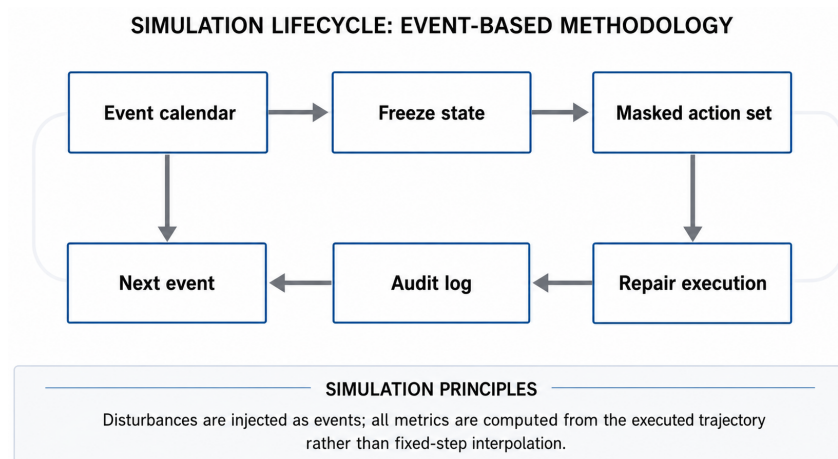


Figure 7. Discrete-event simulation loop for evaluating policy decisions on executed trajectories.

This evaluation design aligns with the systems objective of the paper. The goal is not only to select the best one-step task assignment. The goal is to maintain a stable and feasible logistics process as disturbances unfold over time.

4.8. Metric Interpretation and Evidence Strength

The reported metrics capture different parts of dispatching performance. The online proxy α_{online} measures whether the simulated policy completes or protects key atomic tasks under the environment’s internal task-status accounting. It is useful for model comparison because all online policies face the same task distribution, disturbance process, and feasibility mask. It is not equivalent to an enterprise-level port cut-off compliance rate. A field compliance rate would require real appointment records, measured service times, actual gate-in timestamps, and agreed business definitions of success.

Mean reward is also not sufficient by itself. In a multi-objective dispatching environment, a policy may obtain high reward by repeatedly selecting easy low-risk actions if the reward weights are poorly calibrated. Conversely, a policy may sacrifice short-term reward to protect a high-risk gate-in task. For this reason, the paper reports reward together with α_{train} or α_{online} and stability. The no-curriculum ablation illustrates this point; a policy can display a high mean reward while failing to form reliable high-priority task-completion behavior. Therefore, the evaluation logic is multi-metric by design.

Stability is especially important in operational decision-support systems. A dispatching system that improves completion by frequently disrupting the baseline plan may be mathematically attractive but managerially unacceptable. In the five-seed episode-120 held-out comparison, Transformer-PPO had lower mean plan-deviation SDS than the matched MLP (0.825 versus 0.969), while the earliest-cutoff rule and candidate-chain heuristic were lower still (0.669 and 0.704). In the separate ten-seed screen, Transformer-PPO also reduced SDS relative to the MLP (0.542 versus 0.918), although the rule baseline remained lower (0.173). These results support treating stability as an independent governance metric; they do not establish a blanket learned-policy advantage.

Taken together, the results provide moderate evidence. The evaluation includes five-seed 120-episode checkpoint curves, paired inference, an independent mask audit, reward-weight sensitivity, a rolling-search comparator, and a three-by-three stress test. Nevertheless, five long-run seeds provide limited statistical power, the sensitivity screen uses three seeds and all observations remain synthetic. The results demonstrate trade-offs under the specified simulation protocol and agreement for the implemented feasibility rules, not field or asymptotic superiority.

4.9. Systems-Level Interpretation

The main systems result is not the neural encoder alone. Baseline planning, event-driven triggering, candidate-chain generation, feasibility masking, and learning-based ranking together create the dynamic dispatching loop.

Within this decomposition, the baseline plan provides a stable reference and slack resources, while the event calendar determines when a new decision is needed. Candidate chains translate task-vehicle pairs into operational repair options; the feasibility mask enforces hard rules; the policy ranks feasible repairs by expected long-term value; and the simulation loop evaluates downstream effects along executed trajectories.

This decomposition keeps the neural model from becoming a standalone black box. If a recommendation fails, the audit log can identify whether the failure came from an omitted business rule, an inaccurate travel-time distribution, a reward-weight problem, or a policy-ranking error.

4.10. Static Planning and Dynamic Repair as Two Time Scales

A key systems insight is that static planning and dynamic rescheduling should not be treated as separate systems. Static planning creates the structural conditions for later repair. If the baseline plan contains no slack and no reserved capacity, the dynamic layer can only perform costly reassignment after a disturbance occurs. Conversely, if the dynamic layer ignores the baseline plan, it may produce frequent changes that destabilize execution.

The proposed architecture links offline planning with event-driven online repair. Static planning provides task chains, slack, and shadow tasks. The online policy consumes or reconstructs these resources through audited local actions. Robustness is therefore represented by concrete operational objects: reserved capacity, remaining slack, masked repair options, and recorded deviation from the baseline.

4.11. Matched Rolling-Horizon Online Benchmark

Table 12 reports the matched large-scale nominal comparison. RH-2 achieved α_{online} 0.3011, reward 110.73, SDS 0.706, and mean response latency 230.9 ms across three seeds. Transformer-PPO achieved 0.2867, 68.61, 0.908, and 63.4 ms, respectively. RH-2 therefore provides a genuine online search comparator but not a universal winner. Its short lookahead changes the reward/completion trade-off while imposing substantially greater latency. The appropriate conclusion is complementarity of method rather than RL's superiority.

Table 12. Matched large-scale nominal online benchmark with corrected event-dependent travel times (mean $\pm$ SD over seeds 42–44).

Method	Seeds	Train ep.	α_{online}	Reward	Selector/Update Latency (ms)	SDS	Violations
Earliest-cutoff rule	3	0	0.3190 $\pm$ 0.0365	94.32 $\pm$ 21.44	36.9 $\pm$ 1.9	0.648 $\pm$ 0.075	0.0 violations
Candidate-chain heuristic	3	0	0.3011 $\pm$ 0.0088	110.74 $\pm$ 9.50	55.3 $\pm$ 3.0	0.706 $\pm$ 0.029	0.0 violations
Transformer-PPO	3	30	0.2867 $\pm$ 0.0134	68.61 $\pm$ 8.70	63.4 $\pm$ 4.8	0.908 $\pm$ 0.046	0.0 violations
RH-2 rolling lookahead	3	0	0.3011 $\pm$ 0.0088	110.73 $\pm$ 9.48	230.9 $\pm$ 20.2	0.706 $\pm$ 0.029	0.0 violations

4.12. Robustness Across Scale and Disturbance Intensity

Table 13 summarizes the robustness screen. Across the nine Transformer-PPO cells, mean α_{online} ranged from 0.2852 to 0.7967, reward from 61.20 to 115.15, SDS from 0.805 to 0.921, and mean latency from 22.5 to 66.6 ms. No audited policy action produced a recorded feasibility violation in the 27 Transformer seed-scenario trajectories. The variation across cells shows that feasibility transfers more consistently than performance; task scale and disturbance intensity still change the efficiency-stability trade-off. The high regime also activates more shadow demand earlier, so α_{online} and reward are not expected to deteriorate monotonically: both the workload and the set of executable opportunities change. The table is therefore a scenario-sensitivity test rather than a single-parameter dose-response curve.

Table 13. Transformer-PPO robustness screen across scale and disturbance intensity (mean $\pm$ SD over seeds 42–44; 30 training episodes per seed).

Scale	Atoms (Real + Shadow)/Vehicles	Disturbance	α_{online}	Reward	Selector/Update Latency (ms)	SDS
Small	41 (33 + 8)/20	Low	0.6285 $\pm$ 0.1682	62.31 $\pm$ 13.55	22.5 $\pm$ 1.5	0.872 $\pm$ 0.039
Small	41 (33 + 8)/20	Nominal	0.6053 $\pm$ 0.1549	80.39 $\pm$ 22.14	24.1 $\pm$ 1.0	0.856 $\pm$ 0.026
Small	41 (33 + 8)/20	High	0.7967 $\pm$ 0.0754	102.12 $\pm$ 14.35	28.7 $\pm$ 2.8	0.865 $\pm$ 0.045
Medium	70 (62 + 8)/40	Low	0.4062 $\pm$ 0.0128	61.20 $\pm$ 20.58	45.7 $\pm$ 5.3	0.821 $\pm$ 0.095
Medium	70 (62 + 8)/40	Nominal	0.4030 $\pm$ 0.0122	68.12 $\pm$ 22.39	48.8 $\pm$ 10.9	0.812 $\pm$ 0.076
Medium	70 (62 + 8)/40	High	0.5476 $\pm$ 0.0134	115.15 $\pm$ 12.94	42.5 $\pm$ 3.4	0.805 $\pm$ 0.069
Large	100 (88 + 12)/60	Low	0.2852 $\pm$ 0.0138	64.97 $\pm$ 5.40	66.6 $\pm$ 10.1	0.921 $\pm$ 0.060
Large	100 (88 + 12)/60	Nominal	0.2867 $\pm$ 0.0134	68.61 $\pm$ 8.70	63.4 $\pm$ 4.8	0.908 $\pm$ 0.046
Large	100 (88 + 12)/60	High	0.3819 $\pm$ 0.0196	101.93 $\pm$ 24.49	65.4 $\pm$ 1.0	0.873 $\pm$ 0.101

Relative ranking is also metric-dependent. Across the nine factorial cells, the earliest-cutoff rule, candidate-chain heuristic and Transformer-PPO led α_{online} in 9, 0, and 0 cells, led reward in 0, 9, and 0 cells, and achieved the lowest SDS in 9, 0, and 0 cells. This explicitly rejects a blanket superiority claim and positions the learned policy as one audited ranking option in a governed decision-support system.

4.13. Reward-Weight Sensitivity

The five local weight profiles retained zero audited feasibility violations, confirming that hard safety does not depend on reward calibration. Operational outcomes were sensitive; mean α_{online} ranged from 0.2688 to 0.3154, mileage from 425.1 to 642.9 km, and SDS from 0.689 to 0.861. The nominal profile gave the lowest mean mileage (425.1 km) and SDS (0.689); the safety-heavy α_{online} difference relative to nominal was -0.0036 (Wilcoxon $p = 1.0000$). Table 14 therefore supports the nominal vector as a declared practical compromise, not as a uniquely optimal estimate.

Table 14. Reward-weight sensitivity after 60 episodes (mean $\pm$ sample SD over seeds 42–44).

Profile	(ω_{safety} , ω_{econ} , ω_{stab})	α_{online}	Mileage (km)	SDS	Violations
Safety-heavy	(0.80, 0.10, 0.10)	0.2688 $\pm$ 0.0775	540.2 $\pm$ 208.7	0.812 $\pm$ 0.081	0.0
Nominal	(0.70, 0.15, 0.15)	0.2724 $\pm$ 0.0508	425.1 $\pm$ 74.3	0.689 $\pm$ 0.034	0.0
Balanced	(0.60, 0.20, 0.20)	0.2939 $\pm$ 0.0164	598.0 $\pm$ 94.0	0.861 $\pm$ 0.129	0.0
Economy-heavy	(0.60, 0.30, 0.10)	0.3154 $\pm$ 0.0345	563.0 $\pm$ 39.8	0.798 $\pm$ 0.026	0.0
Stability-heavy	(0.60, 0.10, 0.30)	0.3011 $\pm$ 0.0108	642.9 $\pm$ 317.8	0.736 $\pm$ 0.020	0.0

Cross-profile total reward is intentionally omitted because changing the weights changes the reward measurement scale. The screen supports a bounded calibration region and makes the safety-efficiency-stability trade-off explicit.

5. Discussion

5.1. System Integration Across Planning and Execution

The first contribution is the two-time-scale interface. E-ALNS supplies a stable baseline, slack, and shadow capacity; event detection freezes an execution state; candidate chains expose local repairs; the audit validates hard rules; and DES logs the consequence. None of these modules alone is novel, but their explicit interface addresses the gap between daily ICD planning and governed runtime repair.

5.2. Feasibility Governance: Scope and Boundary

Masking is an established policy-gradient technique [26], not an algorithmic innovation. The contribution is an enumerated ICD rule boundary and an auditable separation between permission and preference. Zero oracle disagreements support the six implemented predicates; they do not cover contracts, bays, swaps, or invalid raw data. Reporting construction cost prevents feasibility governance from being treated as computationally free.

5.3. Stability, Interpretability and Adaptability

Candidate-chain records explain task type, vehicle, slack, appointment state, cut-off risk, and baseline deviation. The Transformer provides a relational ranking mechanism that can be retrained when distributions change, while SDS quantifies the instability of a plan. These are useful decision-support properties even when a deterministic rule wins the current completion or reward metric.

5.4. Boundary of Learned-Policy Performance

Taken together, Table 8, the five-seed long run, and the factorial screen do not demonstrate universal Transformer-PPO dominance. Different methods lead on different objectives, and the long-run intervals remain wide. Moreover, RH-2 is a bounded two-step comparator, so the present evidence cannot establish how learned ranking compares with rolling MILP, CP-SAT, model-predictive control, or deeper online ALNS under matched compute budgets. The learned policy should therefore serve as one governed ranking option alongside optimization or deterministic fallbacks and dispatcher review, not as a replacement for optimization or human control.

5.5. Deployment Roadmap

Deployment should proceed from field-calibrated offline simulation to shadow mode, assisted dispatch and only then narrowly defined automation. An initial field study should replay historical decisions without affecting operations and compare recommendations with realized dispatches, cut-off compliance, empty mileage, response latency, and SDS. Prospective shadow-mode operation should then record dispatcher acceptance, overrides and previously unmodeled constraints across multiple disturbance regimes. Assisted dispatch should begin only after preregistered safety and service thresholds are met without deterioration relative to the incumbent rule or optimizer. Mask construction should be cached and profiled, missing constraints added from disagreement logs, and automatic execution restricted to predefined low-risk audited actions.

5.6. Positioning for Systems Research

The systems contribution is the governed coupling of planning, execution, learning, and organizational acceptance. A snapshot-optimal route may be operationally poor if it changes instructions or consumes slack prematurely; so, feasibility, stability, latency, and explanation accompany reward and completion metrics.

5.7. Limitations

This study remains an internally validated simulation study. It uses no real ICD event logs, live dispatcher comparisons, out-of-time operational tests, or external-depot replications. Accordingly, the simulated online completion proxy and reward should not be interpreted as field completion, cut-off compliance, cost savings, or user acceptance. The five 120-episode seeds strengthen the long-run comparison but offer limited statistical power and do not prove convergence; the reward-weight screen uses three seeds. The mask covers only the six implemented predicates, and its current list scan is unoptimized. The 150-slot policy tensor accommodates 100 active atoms and 50 padding slots in the audit and long-run experiments, rather than all approximately 370 atoms derived from 100 source orders. Deployment would therefore require rolling-window remapping or a larger encoder. RH-2 is a two-step benchmark and does not represent advanced rolling MILP, CP-SAT, model-predictive control, or deeper online ALNS. Transfer to another depot would also require local task and event schemas, an independently audited constraint set, recalibrated travel/service distributions and reward preferences, a regenerated static baseline, and retraining or out-of-distribution testing. Operator heterogeneity, bay capacity, swaps, and other site-specific rules remain outside the current model.

5.8. Future Research Directions

Future research should proceed along three tracks. First, multi-ICD historical replay and prospective shadow-mode studies should use chronological train/calibration/test splits and site-specific KPIs, with dispatcher overrides analyzed before assisted use. Second, rolling MILP, CP-SAT, model-predictive control, and deeper online ALNS should be compared under identical event states, candidate definitions, feasibility rules, warm starts, and wall-clock budgets. Pareto reporting should retain completion, cost, SDS, violations, and end-to-end latency. Third, cross-site studies should distinguish invariant interfaces—event triggering, permission-before-preference auditing and logging—from site-specific modules for task ontology, constraints, distributions, baseline planning, and reward calibration. Rule predicates should be cached and, where possible, formally verified. The target is a monitored planning-and-repair system rather than an unconstrained autonomous policy.

6. Conclusions

This study presents an event-driven and feasibility-audited decision-support framework for ICD truck rescheduling. Rather than proposing a new PPO, Transformer, or masking algorithm, it connects a static baseline, event-triggered candidate repairs, six declared hard-rule checks, relational ranking, executed DES transitions, and traceable logs.

The results support a bounded conclusion. Five-seed 120-episode curves strengthen the encoder comparison, the independent oracle audit supports the implemented mask, and reward sensitivity reveals calibration trade-offs. However, rules, heuristics, and RH-2 remain stronger on selected metrics, and no method dominates completion, reward, stability and latency simultaneously.

The framework is therefore positioned as governed, simulation-based decision support. Its modular interfaces may be reusable, but performance cannot be assumed to transfer across depots without site-specific schema and constraint mapping, field calibration, independent rule auditing, and out-of-distribution testing. Historical replay, prospective shadow-mode evaluation, broader constraint coverage, faster mask construction, larger statistical studies, matched rolling MILP and CP-SAT, and deeper online ALNS baselines are needed before claims of real-world effectiveness or autonomous deployment.

Author Contributions: S.F. (Shucheng Fan): Conceptualization, methodology, software, formal analysis, investigation, data curation, visualization, and writing—original draft. S.F. (Shaochuan Fu): Supervision, validation, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No external, proprietary, or personal datasets were used in this study. The reported results were generated from synthetic instances under the simulation design, parameter settings, and random seeds specified in the manuscript. The resulting run logs and intermediate files have been retained as internal research records and are not publicly archived.

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

References

1. Crainic, T.G.; Gendreau, M.; Potvin, J.-Y. Intelligent freight-transportation systems: Assessment and the contribution of operations research. *Transp. Res. Part C Emerg. Technol.* **2009**, *17*, 541–557. [\[CrossRef\]](#)
2. SteadieSeifi, M.; Dellaert, N.P.; Nuijten, W.; Van Woensel, T.; Raoufi, R. Multimodal freight transportation planning: A literature review. *Eur. J. Oper. Res.* **2014**, *233*, 1–15. [\[CrossRef\]](#)
3. Namboothiri, R.; Erera, A.L. Planning local container drayage operations given a port access appointment system. *Transp. Res. Part E Logist. Transp. Rev.* **2008**, *44*, 185–202. [\[CrossRef\]](#)
4. Jula, H.; Chassiakos, A.; Ioannou, P. Port dynamic empty container reuse. *Transp. Res. Part E Logist. Transp. Rev.* **2006**, *42*, 43–60. [\[CrossRef\]](#)
5. Vieira, G.E.; Herrmann, J.W.; Lin, E. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *J. Sched.* **2003**, *6*, 39–62. [\[CrossRef\]](#)
6. Ouelhadj, D.; Petrovic, S. A survey of dynamic scheduling in manufacturing systems. *J. Sched.* **2009**, *12*, 417–431. [\[CrossRef\]](#)
7. Aytug, H.; Lawley, M.A.; McKay, K.; Mohan, S.; Uzsoy, R. Executing production schedules in the face of uncertainties: A review and some future directions. *Eur. J. Oper. Res.* **2005**, *161*, 86–110. [\[CrossRef\]](#)
8. Herroelen, W.; Leus, R. Robust and reactive project scheduling: A review and classification of procedures. *Int. J. Prod. Res.* **2004**, *42*, 1599–1620. [\[CrossRef\]](#)
9. Bengio, Y.; Lodi, A.; Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d’horizon. *Eur. J. Oper. Res.* **2021**, *290*, 405–421. [\[CrossRef\]](#)
10. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement learning for combinatorial optimization: A survey. *Comput. Oper. Res.* **2021**, *134*, 105400. [\[CrossRef\]](#)
11. Steenken, D.; Voß, S.; Stahlbock, R. Container terminal operation and operations research—a classification and literature review. *OR Spectr.* **2004**, *26*, 3–49. [\[CrossRef\]](#)
12. Fazi, S.; Choudhary, S.K.; Dong, J.-X. The multi-trip container drayage problem with synchronization for efficient empty containers re-usage. *Eur. J. Oper. Res.* **2023**, *310*, 343–359. [\[CrossRef\]](#)
13. Chen, R.; Meng, Q.; Jia, P. Container port drayage operations and management: Past and future. *Transp. Res. Part E Logist. Transp. Rev.* **2022**, *159*, 102633. [\[CrossRef\]](#)
14. Chen, R.; Jia, S.; Meng, Q. Dynamic container drayage booking and routing decision support approach for E-commerce platforms. *Transp. Res. Part E Logist. Transp. Rev.* **2023**, *177*, 103220. [\[CrossRef\]](#)
15. Stoop, K.; Pickavet, M.; Colle, D.; Audenaert, P. The dynamic stochastic container drayage problem with truck appointment scheduling. *OR Spectr.* **2024**, *46*, 953–985. [\[CrossRef\]](#)
16. Huang, Y.; Jin, Z.; Liu, P.; Wang, W.; Zhang, D. Container drayage problem integrated with truck appointment system and separation mode. *Comput. Ind. Eng.* **2024**, *193*, 110307. [\[CrossRef\]](#)
17. Ghiani, G.; Guerriero, F.; Laporte, G.; Musmanno, R. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *Eur. J. Oper. Res.* **2003**, *151*, 1–11. [\[CrossRef\]](#)
18. Pillac, V.; Gendreau, M.; Guéret, C.; Medaglia, A.L. A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.* **2013**, *225*, 1–11. [\[CrossRef\]](#)
19. Ojeda Rios, B.H.; Xavier, E.C.; Miyazawa, F.K.; Amorim, P.; Curcio, E.; Santos, M.J. Recent dynamic vehicle routing problems: A survey. *Comput. Ind. Eng.* **2021**, *160*, 107604. [\[CrossRef\]](#)

20. Hildebrandt, F.D.; Thomas, B.W.; Ulmer, M.W. Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Comput. Oper. Res.* **2023**, *150*, 106071. [\[CrossRef\]](#)
21. Baty, L.; Jungel, K.; Klein, P.S.; Parmentier, A.; Schiffer, M. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. *Transp. Sci.* **2024**, *58*, 708–725. [\[CrossRef\]](#)
22. Bi, J.; Ma, Y.; Zhou, J.; Song, W.; Cao, Z.; Wu, Y.; Zhang, J. Learning to handle complex constraints for vehicle routing problems. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 93479–93509. [\[CrossRef\]](#)
23. Nazari, M.; Oroojlooy, A.; Snyder, L.; Takáč, M. Reinforcement learning for solving the vehicle routing problem. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 9839–9849.
24. Kool, W.; van Hoof, H.; Welling, M. Attention, learn to solve routing problems! In Proceedings of the International Conference on Learning Representations 2019, New Orleans, LA, USA, 6–9 May 2019.
25. Konovalenko, A.; Hvattum, L.M. Optimizing a dynamic vehicle routing problem with deep reinforcement learning: Analyzing state-space components. *Logistics* **2024**, *8*, 96. [\[CrossRef\]](#)
26. Huang, S.; Ontañón, S. A closer look at invalid action masking in policy gradient algorithms. In *The International FLAIRS Conference Proceedings*, University of Florida George A. Smathers Libraries ed; 2022; Volume 35. [\[CrossRef\]](#)
27. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [\[CrossRef\]](#)
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
29. Zhang, J.; van Woensel, T. Dynamic vehicle routing with random requests: A literature review. *Int. J. Prod. Econ.* **2023**, *256*, 108751. [\[CrossRef\]](#)
30. Zhou, C.; Ma, J.; Douge, L.; Chew, E.P.; Lee, L.H. Reinforcement learning-based approach for dynamic vehicle routing problem with stochastic demand. *Comput. Ind. Eng.* **2023**, *182*, 109443. [\[CrossRef\]](#)
31. Boggyrbayeva, A.; Meraliyev, M.; Mustakhov, T.; Dauletbayev, B. Machine learning to solve vehicle routing problems: A survey. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 4754–4772. [\[CrossRef\]](#)
32. Kadyrov, S.; Azamov, A.; Abdumajitov, Y.; Turan, C. Deep reinforcement learning for dynamic vehicle routing with demand and traffic uncertainty. *Oper. Res. Perspect.* **2025**, *15*, 100351. [\[CrossRef\]](#)
33. Demirbilek, M. Optional and mandatory assignment strategies for dynamic vehicle routing with time windows. *Ain Shams Eng. J.* **2025**, *16*, 103462. [\[CrossRef\]](#)
34. Sun, S.; Dong, Y. Improving container port efficiency: A data-driven model for optimizing truck arrival appointments through distributionally robust optimization. *J. Adv. Transp.* **2025**, *2025*, 8137761. [\[CrossRef\]](#)
35. Pinedo, M.L. *Scheduling: Theory, Algorithms, and Systems*, 5th ed.; Springer: Cham, Switzerland, 2016. [\[CrossRef\]](#)
36. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [\[CrossRef\]](#)
37. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; Wiley: New York, NY, USA, 1994. [\[CrossRef\]](#)
38. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
39. Law, A.M. *Simulation Modeling and Analysis*, 5th ed.; McGraw-Hill Education: New York, NY, USA, 2015.
40. Banks, J.; Carson, J.S.; Nelson, B.L.; Nicol, D.M. *Discrete-Event System Simulation*, 5th ed.; Pearson: Upper Saddle River, NJ, USA, 2010.
41. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2018; Volume 32, pp. 3207–3214. [\[CrossRef\]](#)

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.