SCIENTIFIC OASIS

Journal of Intelligent Decision Making and Granular Computing

Journal homepage: www. jidmag.org
ISSN: 3042-3759

Rolling-Horizon Vehicle Routing for Dynamic Urban Distribution with Soft Time Windows: A Dynamic Adaptive Large Neighborhood Search Approach

Anbang Xie [1], Min Kong [1,*]

[1] School of Economics and Management, Anhui Normal University, Wuhu, China

ARTICLE INFO

Article history:
R Received 17 April 2026
Received in revised form 4 July 2026
Accepted 20 August 2026
Available online 24 August 2026

Keywords:
Dynamic vehicle routing; Urban distribution; Rolling horizon; Adaptive large neighborhood search; Soft time windows; Dynamic orders.

ABSTRACT

Urban distribution plans often have to be adjusted after vehicles leave the depot because new orders continue to arrive and some vehicles are still executing earlier routes. This creates a routing problem in which available information and vehicle states change over time. The problem considered here involves a single depot, vehicle-capacity limits, soft latest-service times, fixed committed routes, and limited order postponement. The study aims to improve route adjustment when only part of the order information is known at the beginning of the planning horizon. A rolling-horizon dynamic adaptive large neighborhood search method is developed for this setting. At each decision epoch, newly released orders are combined with unfinished orders from the previous epoch, and only vehicles that have returned to the depot are available for replanning. Orders are served whenever a feasible insertion exists; postponement is used only when no feasible insertion can be found. Three operators are added to the search process to account for order time pressure, insertion priority, and postponement history. Local search, adaptive operator weighting, and simulated-annealing acceptance are also used. The common search parameters are selected through an orthogonal experiment. The method is tested on dynamic Solomon RC101 instances with 40, 60, and 80 customers and is compared with five benchmark approaches. It obtains the best overall mean rank of 2.05. At 40 customers, it is not the best-performing method. At 60 and 80 customers, however, it gives the lowest mean objective values. For the 60-customer instances, the mean objective value is 1905.79, compared with 2047.52 for the standard rolling-horizon adaptive large neighborhood search method. The corresponding values for the 80-customer instances are 2326.43 and 2351.53. The proposed method also reaches 90% of its total improvement after 9.5% of normalized search progress, while the standard rolling-horizon method requires 26.0%. Its performance advantage becomes larger when a greater share of orders is released dynamically. The method is therefore more useful when route adjustment becomes more difficult, although its advantage is not consistent under every tested condition.

* *Corresponding author.*
E-mail address: minkong@ahnu.edu.cn

https://doi.org/10.31181/jidmgc21202650

1.Introduction

Urban delivery demand has grown alongside e-commerce and online consumption. In 2024, China's online retail sales reached RMB 15.52 trillion, 7.2% above the previous year. Physical-goods online retail accounted for RMB 13.08 trillion, an increase of 6.5% and 26.8% of total consumer-goods retail sales. Express carriers handled 175.08 billion parcels, up 21.5%, and generated RMB 1.4 trillion in revenue, up 13.8%. At this volume, once-a-day static planning is increasingly inadequate. Orders may arrive after dispatch, so plans must be revised using the latest order information and vehicle execution states.

Dynamic arrivals become harder to manage when service-time requirements are also present. Customers expect delivery within agreed periods, yet a strict latest-service requirement can leave orders without a reasonable assignment when arrivals are concentrated or vehicles are scarce. Limited tardiness or postponement creates routing flexibility, but repeated deferral reduces service quality and shifts pressure to later epochs. Dynamic requests, time-dependent operating conditions, and vehicle workloads thus interact in ways that make predetermined routes unsuitable for a changing execution environment [1]. The timing of time-window assignment and adjustment can likewise affect transport cost, punctuality, and route quality [2].

As illustrated in Figure 1, the distribution system considered in this study consists of a single depot, a homogeneous vehicle fleet, and a set of customer orders. Each order has a release time, demand, service duration, priority level, and soft time window. Some orders are available at the beginning of the planning horizon, whereas the remaining orders are gradually released during vehicle execution. At each rolling decision time, newly released orders are combined with orders postponed from the preceding epoch to form the current active-order set. Vehicles that have completed their committed routes and returned to the depot can participate in replanning. Vehicles that are still executing routes committed in earlier epochs remain in transit, and their routes cannot be modified until they return. This operating process creates a direct relationship between current routing decisions and the adjustment opportunities available in subsequent epochs. Assigning all currently visible orders immediately may increase travel distance, vehicle use, and order tardiness. Postponing too many orders, by contrast, may concentrate delivery tasks in later epochs and increase the risk of repeated postponement. The decision maker must therefore coordinate order assignment, customer sequence, immediate service, permitted tardiness, and order postponement under changing order and vehicle states. The practical quality of a delivery route cannot be evaluated solely by travel distance. Different delivery periods may lead to different travel efficiencies and operating outcomes, and temporal conditions can change the relative value of alternative routes [3]. Research based on real last-mile delivery data has also shown that driver preferences and route characteristics influence actual route selection [4]. In addition, practical delivery operations may involve hidden constraints that are not explicitly represented in a basic routing model but still affect route feasibility [5]. Although this study does not directly model driver preferences or identify hidden constraints, these findings indicate that route evaluation should account not only for travel cost but also for service timing, vehicle execution states, and consistency with actual operations.

The problem remains computationally difficult even before its dynamic features are considered, because the capacitated vehicle routing problem is NP-hard. Repeated order releases and rolling decisions continually change the system state, while capacities, time windows, and committed routes narrow the feasible adjustment space. Re-solving the full mixed-integer model at every epoch would be costly and could imply revisions to routes already under execution. The setting therefore calls for a heuristic that can repeatedly construct and improve feasible solutions within the available decision time.

Adaptive large neighborhood search (ALNS) repeatedly removes and reinserts parts of an incumbent solution, allowing several problem-specific operators to coexist within one search framework. Voigt's [6] systematic analysis shows that removal strategies, insertion procedures, and operator composition directly influence performance; adaptive selection does not compensate for a poorly designed operator pool. This flexibility is useful in urban distribution because route reconstruction can incorporate changing order states, time pressure, and postponement history.

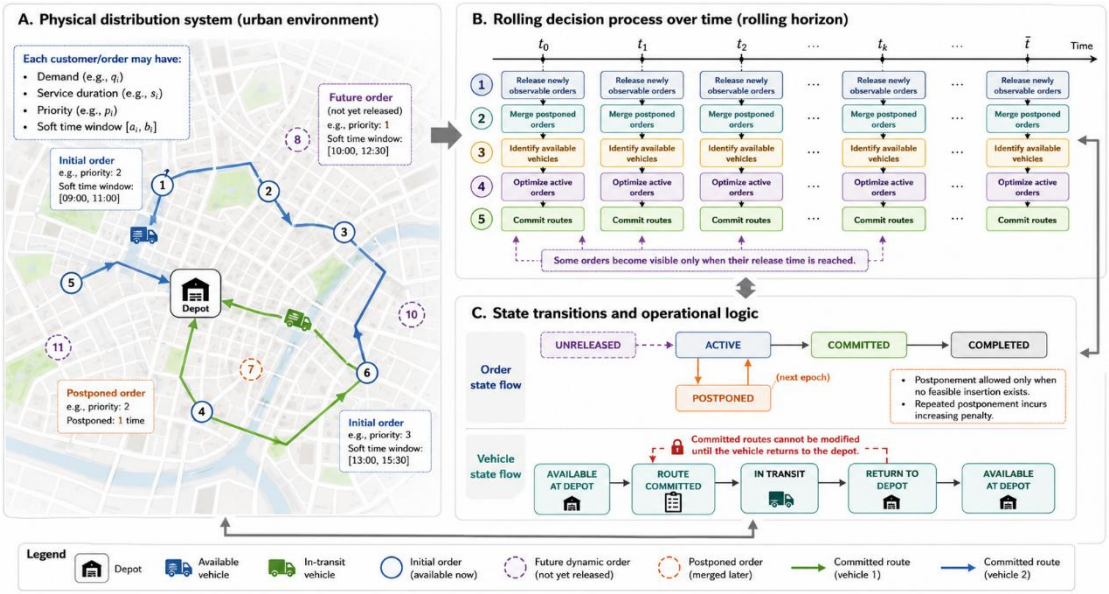

Fig. 1. Dynamic orders and time-constrained urban distribution process

These characteristics motivate a rolling-horizon Dynamic-Adaptive Large Neighborhood Search algorithm (D-ALNS). The rolling horizon provides the outer decision structure: at each epoch, the system updates order lifecycles and vehicle states, releases newly visible orders, and identifies vehicles that have returned to the depot. A service-first greedy procedure constructs the initial feasible solution. D-ALNS then applies destroy-and-repair operations, local search, adaptive operator selection, and simulated-annealing acceptance. Alongside random, worst-cost, and related removal and greedy and regret-2 insertion, the operator pool includes time-pressure removal, time-priority insertion, and postponed-priority insertion. The service-first rule applies during construction and repair: an order with any feasible insertion must be routed, while postponement is allowed only when every insertion is infeasible and the current epoch is not final. Committed routes remain outside subsequent optimization until their vehicles return, keeping planning decisions consistent with physical execution and preventing duplicate modification or charging.

The paper makes three contributions: The first is a rolling-horizon urban distribution model that represents dynamic release, soft time windows, vehicle capacity and availability, route commitment, and order postponement in one framework. Its objective combines travel, fixed dispatch, weighted tardiness, and postponement costs. Cumulative postponement carries an increasing penalty, while routes committed in earlier epochs remain fixed. The second is D-ALNS, which uses dynamic order-state information within service-first feasible construction, standard destroy and repair, local search, adaptive weighting, and simulated annealing. Its three dynamic-order-oriented operators are time-pressure removal, time-priority insertion, and postponed-priority insertion. RH-ALNS and D-ALNS otherwise share the rolling framework, initial solution, route evaluator, and search controls, permitting a direct assessment of the additional operators. The third is an evaluation across problem scales and dynamic conditions. Solomon benchmark instances provide clustered, random, and mixed

customer distributions. Parameter calibration, formal comparison, convergence analysis, ablation, and sensitivity experiments examine solution quality, computation, and robustness, including sensitivity to the dynamic-order ratio.

The paper proceeds as follows. Section 2 reviews dynamic vehicle routing, rolling-horizon optimization, and large neighborhood search. Section 3 defines the problem and mathematical model. Section 4 presents D-ALNS, and Section 5 reports the numerical experiments. The final section discusses the findings, limitations, and directions for further work.

2.Literature Review

Dynamic vehicle routing differs from static routing mainly because part of the information becomes available only after operations have started. Reviews of DVRP research organize this field according to information evolution, degree of dynamism, operational characteristics, and solution methodology [7-9]. Orders, vehicle states, and sometimes travel conditions may change during execution, so routing decisions need to be updated rather than determined only once before dispatch. Wang *et al.*, [10] address a multi-depot dynamic vehicle routing problem with time windows by solving a sequence of deterministic subproblems with ALNS. A similar rolling idea appears in same-day delivery and dispatch-wave settings. Pina-Pardo *et al.*, [11] consider newly arriving orders in a truck–drone delivery system, while Lan *et al.*, [12] study whether known requests should be dispatched immediately or retained for later waves. Baty *et al.*, [13] combine machine learning with combinatorial optimization for dynamic vehicle routing with time windows, using predefined decision epochs to update dispatch and routing decisions. These studies use different operational settings, but they all treat routing as a repeated decision process rather than a one-time planning problem.

Dynamicity can also come from changes other than new customer arrivals. Neria and Tzur [14] study pickup and allocation decisions when supply quantities and availability times are gradually revealed. Khorasanian *et al.*, [15] consider home-care routing in which the number of future visits is uncertain. In passenger–parcel transportation, Wang and Xu [16] jointly consider confirmation, compensation, and routing, so current decisions also depend on their possible future effects. Horváth and Tamási [17] provide a more general modeling and simulation framework for dynamic vehicle routing, where state updates and repeated decision points form the basis of the routing process. Together, these studies show that dynamic routing may involve changes in demand, service requirements, resource availability, and decision timing, not only the arrival of new orders.

Vehicle execution states are another important part of dynamic routing. Sze *et al.*, [18] revise routes after traffic disruptions by allowing route changes at critical nodes. Lin *et al.*, [19] combine signal control and routing in a two-stage rolling-horizon framework and update decisions according to predicted traffic states. In both cases, information observed during execution affects later routing decisions. The present study uses a more restrictive execution rule: once a route has been committed, it remains unchanged until the vehicle returns to the depot. This restriction keeps the rolling decision process consistent with the physical state of vehicles already in service.

Time-related constraints have also received considerable attention in vehicle routing. Zhang *et al.*, [20] study uncertain travel times and introduce a riskiness index for evaluating the probability and effect of time-window violations. Metz *et al.*, [21] focus on delay-resistant routing with heterogeneous time windows and seek schedules that remain reliable when travel and service times vary. Flexible time-window structures have also been incorporated into recent routing models. Li and Wang [22] consider hard and soft time windows simultaneously and develop a modified ALNS for the resulting routing problem. More directly related to dynamic routing, Belhaiza and Laporte [23] develop a data-driven heuristic for a dynamic vehicle routing problem with multiple soft time windows, combining predictive information with a dynamic hybrid ALNS. These studies show that

service quality depends on more than travel distance and that temporal flexibility can be incorporated into heuristic route reconstruction. In a rolling setting, newly released orders and shorter remaining service times can make temporal feasibility more difficult to maintain. The present study therefore includes soft latest-service times and tardiness costs, while further imposing immutable committed routes, depot-return-based vehicle availability, and a service-first postponement rule.

The size of the routing problem also affects how often plans can be revised. Accorsi and Vigo [24] show that efficient candidate sets, pruning rules, and data structures can greatly extend the scale handled by local search. Kerscher and Minner [25] use spatial, temporal, and demand information to decompose large vehicle-routing instances before applying local improvement. Their problems are static, but the computational issue is similar: repeated replanning becomes difficult if too much search effort is spent on unpromising candidates. For dynamic routing, this problem is more pronounced because useful solutions must be obtained within each decision period rather than after a long offline computation.

ALNS is well suited to this type of repeated route adjustment because different destroy and repair operators can be used within the same search framework. The adaptive framework was introduced by Ropke and Pisinger [26], who used several competing subheuristics with selection frequencies linked to their historical performance, and was subsequently generalized by Pisinger and Ropke [27] to several vehicle routing variants. Recent studies have also moved beyond fixed operator selection. Johnn *et al.*, [28] use graph reinforcement learning to choose ALNS operators from the current search state. Sun *et al.*, [29] design ALNS operators for electric vehicle routing with soft time windows, and Su *et al.*, [30] adapt removal operations in a multi-depot green vehicle routing problem. Xu *et al.*, [31] further use solution-state and temporal-flexibility information for neighborhood selection in routing with multiple time windows. These studies suggest that operator design can benefit from information about the current solution and operating state. The state information used in the present paper is different, however. It focuses on newly released orders, current time pressure, vehicle availability, and postponement history.

Existing research has therefore covered dynamic order arrival, dispatch timing, traffic disruption, time-window reliability, large-scale routing, and state-dependent neighborhood search from different angles. Fewer studies consider newly released orders, postponed orders, vehicle return states, and immutable committed routes within the same rolling-horizon process. There is also limited evidence on whether operators designed specifically for these dynamic order states can improve a standard rolling-horizon ALNS.

The model in this paper is built around this setting. Newly released orders and orders postponed from the previous epoch form the active order set. Only vehicles that have returned to the depot can be used in the current decision, and routes that are already being executed are not changed. Postponement is allowed only when no feasible insertion exists. On this basis, D-ALNS introduces time-pressure removal, time-priority insertion, and postponed-priority insertion to use the current order state directly during route reconstruction. Table 1 compares the studies that are most closely related to these features. Other cited work is used to support the discussion of dynamic routing, temporal constraints, computational scale, and neighborhood search.

Table 1
Key features of related studies

Reference	Dynamic	Rolling	Time	Exec. state	Deferral	NS	State ops.	Focus and method
Wang *et al.*, [10]	✓	✓	✓			✓		Multi-depot dynamic VRPTW; ALNS
Pina-Pardo *et al.*, [11]	✓	✓	✓	✓				Same-day delivery with truck–drone resupply
Baty *et al.*, [13]	✓	✓	✓		✓	✓	✓	ML-enhanced combinatorial optimization
Lan *et al.*, [12]	✓	✓	✓		✓			Dynamic dispatch waves; scenario sampling
Sze *et al.*, [18]		✓		✓		✓	✓	Traffic-driven route adjustment; AVNS
Zhang *et al.*, [20]			✓					Risk-aware routing under uncertain travel times
Johnn *et al.*, [28]						✓	✓	Graph-RL-based ALNS operator selection
Horváth and Tamási [17]	✓	✓	✓	✓				Dynamic-routing modeling and simulation
Wang and Xu [16]	✓	✓	✓	✓			✓	Anticipatory passenger–parcel routing
Sun *et al.*, [29]			✓			✓		Soft-TW electric routing; improved ALNS
This study	✓	✓	✓	✓	✓	✓	✓	Rolling-horizon D-ALNS with service-first postponement

3. Problem Description and Mathematical Model

This section defines the dynamic urban distribution problem with order release times and time-window constraints and then formulates it as a mixed-integer linear program (MILP).

1.1 Problem Description

Urban distribution sends customer orders from a depot to dispersed locations. A static vehicle routing model assumes that every order is known before routes are generated. In practice, some orders appear only after delivery has begun, and their demand, service duration, and time-window data are unavailable before release. A complete route plan therefore cannot be fixed at the start of the horizon; it must be revised as order and vehicle states become known.

The distribution system considered in this paper consists of one depot, a set of homogeneous distribution vehicles, and a set of customer orders. Each vehicle has the same loading capacity, travel speed, and fixed dispatch cost. A dispatched vehicle departs from the depot, visits a sequence of assigned customers, and returns to the depot after completing its delivery route. Customer demand is indivisible, and each order must be completely served by one vehicle.

Each customer order is characterized by a demand quantity, a release time, a service duration, an earliest service time, a latest preferred service time, and a priority weight. The earliest service time is treated as a hard time-window bound. When a vehicle arrives at a customer before the earliest service time, it must wait until service is allowed to begin. The latest preferred service time is treated as a soft time-window bound. A vehicle may begin service after this time, but the resulting tardiness incurs a penalty cost.

The delivery horizon is divided into rolling decision epochs to represent gradual order arrival. At an epoch, orders released since the preceding decision are combined with unfinished orders carried forward from that epoch. Together they form the active set for the current routing decision. Orders with later release times remain invisible and cannot enter the plan. Vehicle states are updated at the same epochs. A vehicle becomes available for replanning only after completing its committed route and returning to the depot by the current decision time. A vehicle still in transit cannot accept new

orders, and neither its route nor its visiting sequence may change. Replanning is consequently restricted to vehicles currently at the depot. An active order is either assigned to an available vehicle or postponed to the next epoch. Postponement does not cancel the order; the unfinished order returns to the active set at the next decision. Its postponement cost rises with the number of previous deferrals. At the final epoch postponement is prohibited, so every remaining active order must be served. Travel distance and time are deterministic over the horizon. The experiments use Euclidean distance and a constant vehicle speed, and every dispatched vehicle must return to the depot before the planning horizon ends. Four cost components arise at each epoch: travel by newly dispatched routes, fixed dispatch costs for activated vehicles, weighted tardiness for served orders, and postponement costs for orders carried forward. Routes committed at earlier epochs are historical costs and are not charged again.

The objective is to choose vehicle activation, customer assignment and sequence, service timing, and postponement so as to minimize total rolling-horizon cost subject to capacity, route continuity, time-window, vehicle-availability, and planning-horizon constraints. Figure 2 explains the specific situation.

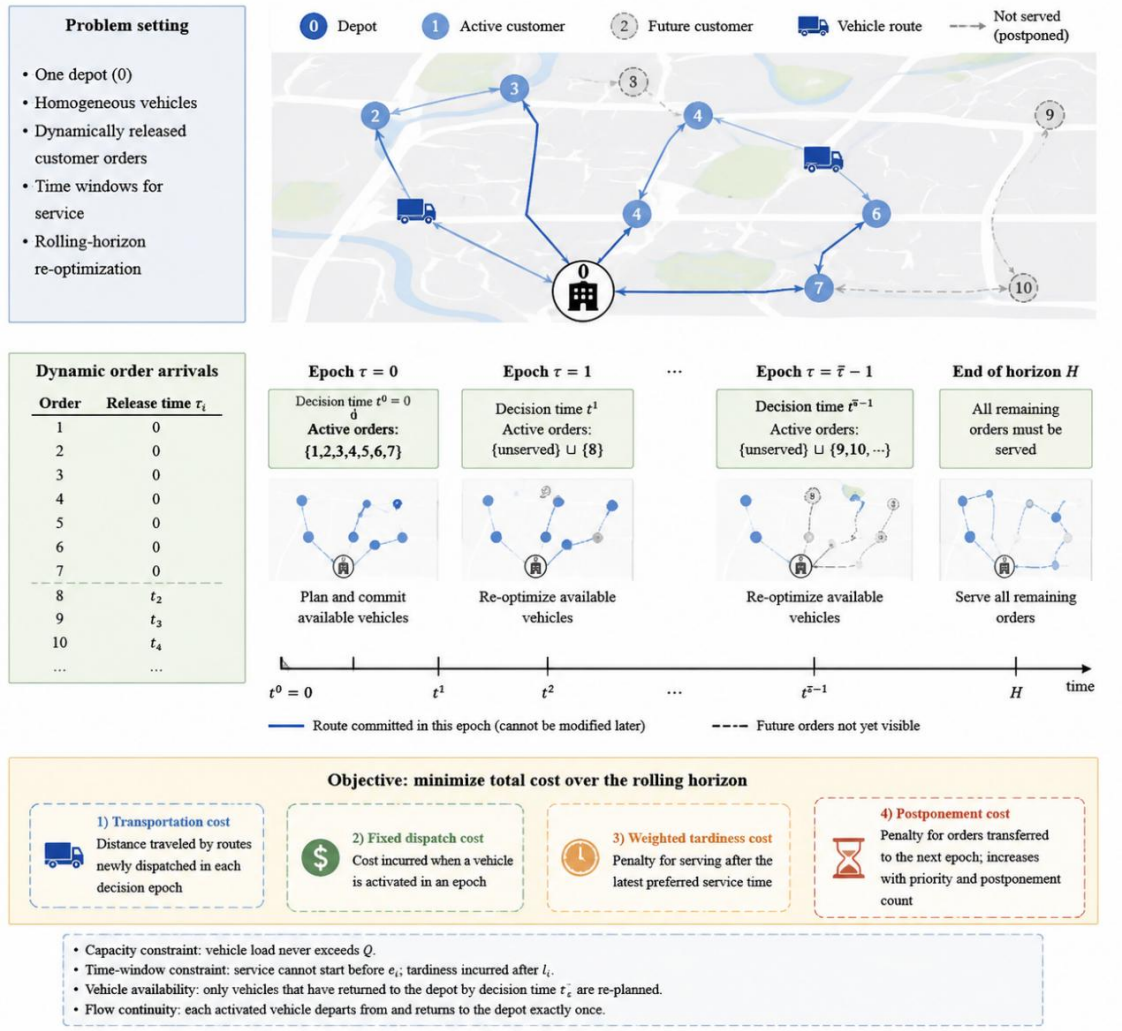

Fig. 2. Urban distribution with dynamic orders and time constraints

1.2 Mathematical Model

Table 2 lists the sets, indices, parameters, and decision variables used in the model.

Table 2
Notations used in the mathematical model

Notation	Definition
Sets:	
N	Set of all customer orders, $N = \{1,2, \dots, n\}$..
$\mathcal{K}$	Set of homogeneous distribution vehicles, $\mathcal{K} = \{1,2, \dots, K\}$.
T	Set of rolling decision epochs, $T = \{0,1, \dots, \bar{\tau}\}$.
N_{new}^{τ}	Set of orders newly observed at decision epoch τ. It contains orders satisfying $r_i \le t^0$ when $\tau = 0$, and orders satisfying $t^{\tau-1} < r_i \le t^{\tau}$ when $\tau \ge 1$..
N_{pen}^{τ}	Set of orders postponed from the preceding decision epoch to epoch τ, where $N_{\text{pen}}^{0} = \emptyset$.
V^{τ}	Set of nodes considered at decision epoch τ, where $V^{\tau} = \{0\} \cup N^{\tau}$, and node 0 denotes the depot.
A^{τ}	Set of candidate travel arcs at decision epoch τ, where $A^{\tau} = \{(i,j) \mid i,j \in V^{\tau}, i \ne j\}$.
$\mathcal{K}^{\tau}$	Set of vehicles available for replanning at decision epoch τ, where $\mathcal{K}^{\tau} = \{k \in \mathcal{K} \mid A_k^{\tau} \le t^{\tau}\}$.
Indices:	
i, j	Indices of depot or customer nodes, $i, j \in V^{\tau}$.
k	Index of distribution vehicles, $k \in \mathcal{K}$.
τ	Index of rolling decision epochs, $\tau \in T$.
Parameters:	
t^{τ}	Decision time of rolling epoch τ, where $t^0 = 0$ and $t^{\bar{\tau}} < H$. After the final decision epoch, committed routes continue to be executed until the end of the planning horizon H.
H	End time of the entire delivery planning horizon.
r_i	Release time of order i, $i \in N$..
d_{ij}	Travel distance from node $i$to node j, $(i,j) \in A^{\tau}$.
θ_{ij}	Travel time from node $i$to node j, $(i,j) \in A^{\tau}$.
q_i	Demand quantity of order i, $i \in N$.
Q	Maximum loading capacity of each vehicle.
e_i	Earliest service time of order i, $i \in N$.
l_i	Latest preferred service time of order i, $i \in N$.
s_i	Service duration of order i, $i \in N$.
p_i	Priority weight of order i, $i \in N$.
A_k^{τ}	Earliest available time of vehicle $k$at decision epoch τ, where $A_k^0 = t^0$.
v_i^{τ}	Number of times order $i$has been postponed before entering epoch τ; its initial value is zero when order $i$is newly released.
c^D	Unit-distance transportation cost.
c^F	Fixed dispatch cost incurred when one vehicle is activated.
c^L	Unit weighted tardiness cost.
c^P	Basic postponement cost coefficient.
γ	Growth coefficient of repeated-postponement cost.
M	A sufficiently large positive constant, no smaller than $H + \max_{i \in N} s_i + \max_{(i,j) \in A^{\tau}} \theta_{ij}$.
v	Constant travel speed of the homogeneous vehicles.
Decision variables	
Binary variables:	
x_{ijk}^{τ}	Equals 1 if vehicle $k$travels directly from node $i$to node $j$at decision epoch τ; otherwise, 0.
y_k^{τ}	Equals 1 if vehicle $k$is dispatched at decision epoch τ; otherwise, 0.
z_i^{τ}	Equals 1 if order $i$is postponed from decision epoch $\tau$to the next epoch; otherwise, 0.
Continuous variables:	
B_i^{τ}	Service start time of order $i$at decision epoch τ.
L_i^{τ}	Tardiness of order i at decision epoch τ.
U_{ik}^{τ}	Cumulative demand carried by vehicle $k$immediately after serving order i.
C_k^{τ}	Time at which vehicle $k$returns to the depot after completing the route dispatched at decision epoch τ.

Using the notation in Table 2, the problem is formulated as a mixed-integer linear program. The objective and constraints follow.

Objective:

$$minZ = \sum_{\tau\in T}\Big[\, c^D \sum_{k\in\mathcal{K}^\tau}\sum_{i,j\in A(\tau)} d_{ij}x_{ijk}^{\tau} + c^F \sum_{k\in\mathcal{K}^\tau} y_k^{\tau} + c^L \sum_{i\in N^\tau} p_i L_i^{\tau} + c^P \sum_{i\in N^\tau} p_i(1+\gamma v_i^{\tau}) Z_i^{\tau} \tag{1}$$

Subject to:

$$\sum_{k\in\mathcal{K}^\tau}\sum_{j\in\mathcal{V}^\tau, j\neq i} d_{ij}x_{ijk}^{\tau} + z_i^{\tau} = 1, \quad \forall i\in N^\tau,\ \tau\in T \tag{2}$$

$$z_i^{\bar{\tau}} = 0, \quad \forall i\in\mathcal{N}^{\bar{\tau}} \tag{3}$$

$$\sum_{j\in\mathcal{V}^\tau, j\neq i} x_{ijk}^{\tau} = \sum_{j\in\mathcal{V}^\tau, j\neq i} x_{jik}^{\tau}, \quad \forall i\in\mathcal{N}^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{4}$$

$$\sum_{j\in N^\tau} x_{0jk}^{\tau} = y_k^{\tau}, \quad \forall k\in\mathcal{K}^\tau,\ \tau\in T \tag{5}$$

$$\sum_{i\in N^\tau} x_{i0k}^{\tau} = y_k^{\tau}, \quad \forall k\in\mathcal{K}^\tau,\ \tau\in T \tag{6}$$

$$q_i \sum_{j\in\mathcal{V}^\tau, j\neq i} x_{ijk}^{\tau} \le U_{ik}^{\tau} \le Q \sum_{j\in\mathcal{V}^\tau, j\neq i} x_{ijk}^{\tau}, \quad \forall i\in N^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{7}$$

$$U_{jk}^{\tau} \ge U_{ik}^{\tau} + q_j - Q\big(1-x_{ijk}^{\tau}\big), \quad \forall i,j\in N^\tau,\ i\neq j,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{8}$$

$$B_j^{\tau} \ge t^{\tau} + \theta_{0j} - M\big(1-x_{0jk}^{\tau}\big), \quad \forall j\in N^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{9}$$

$$B_j^{\tau} \ge B_i^{\tau} + s_i + \theta_{ij} - M\big(1-x_{ijk}^{\tau}\big), \quad \forall i,j\in N^\tau,\ i\neq j,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{10}$$

$$B_i^{\tau} \ge e_i(1-z_i^{\tau}), \quad \forall i\in N^\tau,\ \tau\in T \tag{11}$$

$$B_i^{\tau} \le H(1-z_i^{\tau}), \quad \forall i\in N^\tau,\ \tau\in T \tag{12}$$

$$L_i^{\tau} \ge B_i^{\tau} - l_i, \quad \forall i\in N^\tau,\ \tau\in T \tag{13}$$

$$0 \le L_i^{\tau} \le H(1-z_i^{\tau}), \quad \forall i\in N^\tau,\ \tau\in T \tag{14}$$

$$C_k^{\tau} \ge B_i^{\tau} + s_i + \theta_{i0} - M(1-x_{i0k}^{\tau}), \quad \forall i\in N^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{15}$$

$$C_k^{\tau} \le B_i^{\tau} + s_i + \theta_{i0} + M(1-x_{i0k}^{\tau}), \quad \forall i\in N^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{16}$$

$$0 \le C_k^{\tau} \le Hy_k^{\tau}, \quad \forall k\in\mathcal{K}^\tau,\ \tau\in T \tag{17}$$

$$x_{ijk}^{\tau}\in\{0,1\}, \quad \forall(i,j)\in A^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{18}$$

$$y_k^{\tau}\in\{0,1\}, \quad \forall k\in\mathcal{K}^\tau,\ \tau\in T \tag{19}$$

$$z_i^{\tau}\in\{0,1\}, \quad \forall i\in N^\tau,\ \tau\in T \tag{20}$$

$$B_i^{\tau}, L_i^{\tau} \ge 0, \quad \forall i\in N^\tau,\ \tau\in T \tag{21}$$

$$U_{ik}^{\tau} \ge 0, \quad \forall i\in N^\tau,\ k\in\mathcal{K}^\tau,\ \tau\in T \tag{22}$$

$$C_k^{\tau} \ge 0, \quad \forall k\in\mathcal{K}^\tau,\ \tau\in \mathrm{T} \tag{23}$$

$$N_{pen}^{\tau+1} = \{i\in N^\tau \mid z_i^{\tau}=1\}, \quad \tau=0,\dots,\bar{\tau}-1 \tag{24}$$

$$v_i^{\tau+1} = v_i^{\tau} + 1, \quad \forall i\in N_{pen}^{\tau+1},\ \tau=0,\dots,\bar{\tau}-1 \tag{25}$$

$$A_k^{\tau+1} = \begin{cases} C_k^{\tau}, & k\in\mathcal{K}^\tau,\ y_k^{\tau}=1, \\ A_k^{\tau}, & otherwise, \end{cases} \quad \forall k\in\mathcal{K},\ \tau=0,\dots,\bar{\tau}-1 \tag{26}$$

Equation (1) defines the total objective function over the complete rolling horizon. The first term represents the transportation cost generated by routes newly dispatched at each decision epoch. The second term calculates the fixed dispatch cost of activated vehicles. The third term measures the weighted tardiness cost of served orders. The fourth term calculates the postponement cost of unfinished orders. The postponement cost increases with the priority weight and cumulative postponement count of each order. Costs generated by routes committed at previous epochs are not

recalculated in subsequent decision epochs. Constraint (2) ensures that each active order is either served exactly once in the current decision epoch or postponed to the next epoch. Constraint (3) prohibits order postponement at the final decision epoch. Constraint (4) maintains flow conservation at each customer node visited by a vehicle. Constraints (5) and (6) require each activated vehicle to depart from and return to the depot exactly once. Constraint (7) links the cumulative-load variable to the customer served by each vehicle and restricts the cumulative load by the maximum vehicle capacity. Constraint (8) propagates the cumulative load between consecutively visited customers. Since all customer demands are positive, Constraints (7) and (8) also eliminate customer-only subtours disconnected from the depot. Constraint (9) defines the earliest possible service time of the first customer visited by an activated vehicle. Constraint (10) describes the time-propagation relationship between two consecutively visited customers. Constraint (11) prevents the service of an assigned order from starting before its earliest service time. Constraint (12) sets the service-start-time variable of a postponed order to zero and bounds the service time of a served order within the planning horizon. Constraint (13) calculates the tardiness of each served order relative to its latest preferred service time. Constraint (14) imposes nonnegativity on tardiness and ensures that a postponed order does not generate tardiness cost in the current decision epoch. Constraints (15) and (16) determine the vehicle return time according to the final customer on the selected route. Constraint (17) sets the return time of an unused vehicle to zero and requires every dispatched vehicle to return to the depot before the end of the planning horizon. Constraints (18)-(20) specify the value domains of the binary routing, vehicle-activation, and order-postponement variables. Constraints (21)–(23) specify the nonnegative domains of the service-start-time, tardiness, cumulative-load, and vehicle-return-time variables. Equation (24) transfers postponed orders into the postponed-order set of the next rolling decision epoch. Equation (25) increases the cumulative postponement count of every postponed order. Equation (26) updates vehicle availability. A vehicle dispatched at the current decision epoch becomes available again at its route-return time, whereas the availability time of an unused vehicle or a vehicle still executing a previously committed route remains unchanged.

4.Proposed Algorithm

The model in Section 3 describes a routing problem in which orders appear over time and vehicle availability changes during execution. A dispatched route remains fixed until its vehicle returns to the depot. Solving the entire horizon as a single static problem is therefore inappropriate: the available information and feasible decisions change from one decision epoch to the next.

D-ALNS addresses this structure through two coupled levels. The outer rolling-horizon process updates order and vehicle states. Within each nonempty epoch, the inner search improves routes for currently available vehicles using only the orders visible at that time.

1.3 Overall Algorithmic Framework

At each decision time t^{τ}, the system first advances vehicle execution states to the current epoch. Vehicles that have completed their routes and returned to the depot become available again. Vehicles still travelling on previously committed routes remain unavailable, and their routes are not included in the current optimization.

Orders released by t^{τ} are then identified. Newly released orders and orders postponed from the preceding epoch form the active-order set $\mathcal{N}^{\tau}$. The available-vehicle set $\mathcal{K}^{\tau}$ contains only vehicles located at the depot at the current decision time.

If $\mathcal{N}^{\tau}$ is empty, the search procedure is skipped. Otherwise, a feasible initial solution is constructed and improved by D-ALNS. When the search terminates, the best routes found for the

current epoch are committed. The dispatched vehicles enter the in-transit state, while orders that cannot be feasibly assigned are carried forward to the next epoch.

The operator weights, search statistics, simulated-annealing temperature, current solution, and stage-best solution are initialized separately at each nonempty decision epoch. Search information is not transferred between epochs because the composition of the active orders and available vehicles may change considerably. Figure 3 illustrates the general structure of this framework.

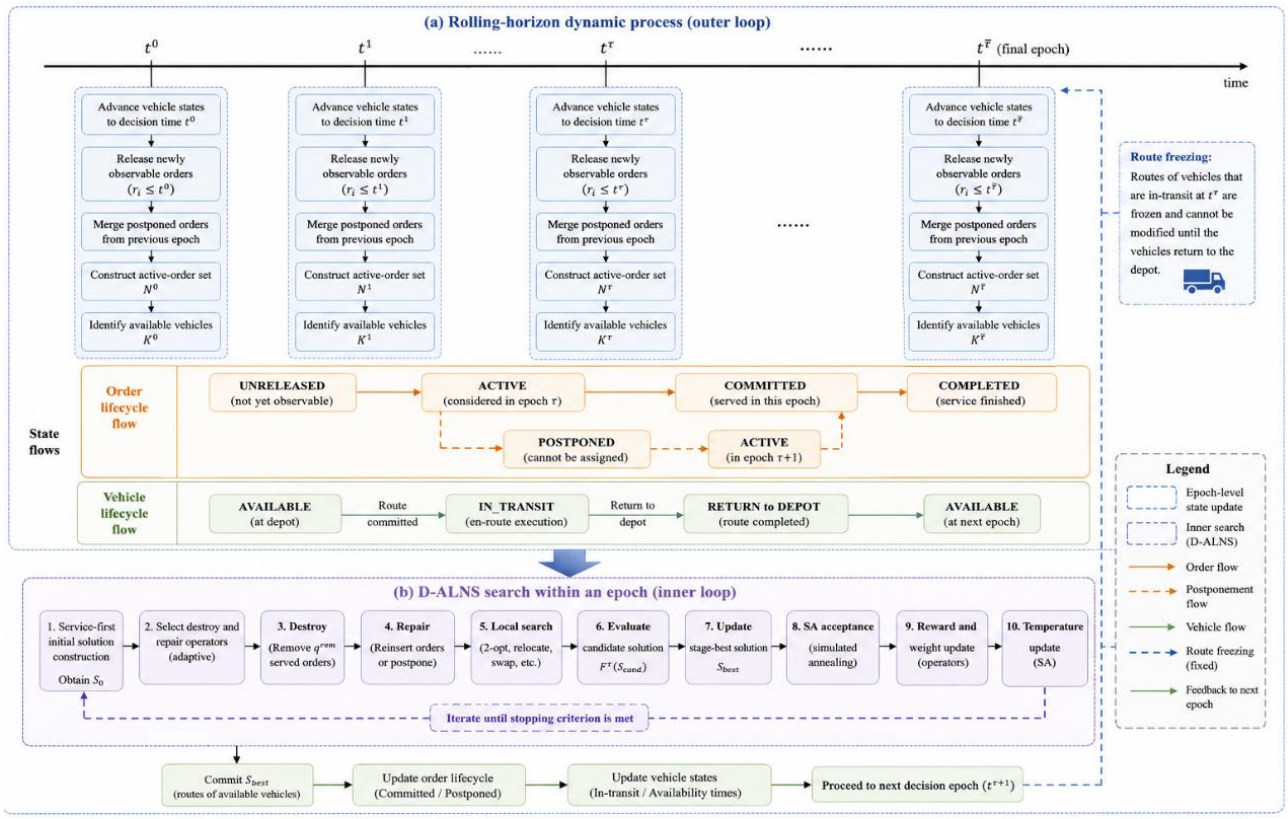

Fig. 3. Rolling-horizon D-ALNS framework for dynamic urban distribution

1.4 Solution Representation and Evaluation

1.4.1 Solution representation

A solution at decision epoch τconsists of a set of vehicle routes and a set of postponed orders:

$$S^{\tau} = (\mathcal{R}^{\tau}, \mathcal{P}^{\tau}), \tag{27}$$

where $\mathcal{R}^{\tau}$ is the route set generated for the available vehicles and $\mathcal{P}^{\tau}$ contains the orders postponed to epoch $\tau + 1$.

The route assigned to vehicle $k$is represented by an ordered customer sequence:

$$R_k^{\tau} = \left(0, i_1, i_2, \dots, i_{n_k}, 0\right), k \in \mathcal{K}^{\tau},$$

where 0 denotes the depot, n_k is the number of customers assigned to vehicle k, and i_h is the customer in route position h. An unused vehicle has an empty route and incurs no fixed dispatch cost.

Every active order appears either in one vehicle route or in $\mathcal{P}^{\tau}$. The customer sequences of different vehicles do not overlap.

The mathematical model in Section 3 uses $x_{ijk}^{\tau}$to describe whether vehicle $k$travels from node $i$to node j. D-ALNS uses the ordered sequence R_k^{τ}instead. The two forms describe the same routing

decision, but the sequence representation is easier to use in insertion, removal, relocation, and 2-opt operations

1.4.2 Route evaluation

The initial constructor, destroy and repair operators, local search, RH-ALNS, D-ALNS, and all comparison algorithms share one route evaluator. Feasibility and objective values are therefore computed on the same basis.

For route $R_k^\tau = (0, i_1, \dots, i_{n_k}, 0)$, vehicle $k$departs from the depot at t^τ. Let $a_{i_h}^\tau$denote its arrival time at customer i_h. Arrival time, service start time, tardiness, and depot return time are obtained from:

$$\begin{aligned} a_{i_h}^\tau &= \begin{cases} t^\tau + \theta_{0i_1}, & h = 1, \\ B_{i_{h-1}}^\tau + s_{i_{h-1}} + \theta_{i_{h-1}i_h}, & h = 2, \dots, n_k, \end{cases} \\ B_{i_h}^\tau &= \max\{e_{i_h}, a_{i_h}^\tau\}, \\ L_{i_h}^\tau &= \max\{0, B_{i_h}^\tau - l_{i_h}\}, \\ C_k^\tau &= B_{i_{n_k}}^\tau + s_{i_{n_k}} + \theta_{i_{n_k}0}. \end{aligned} \tag{28}$$

A vehicle waits when it arrives before e_{i_h}. Since $l_{i_h}$is a soft upper bound, service after l_{i_h}remains feasible but incurs tardiness cost.

A route is feasible if its total demand does not exceed the vehicle capacity and the vehicle returns to the depot within the planning horizon:

$$\sum_{h=1}^{n_k} q_{i_h} \le Q, C_k^\tau \le H. \tag{29}$$

The stage objective is evaluated with Equation (1) after route evaluation. It contains travel, fixed vehicle-dispatch, weighted tardiness, and postponement costs. Costs from routes committed in earlier epochs are excluded from the current-stage objective because they have already entered the historical cost ledger.

The notation $F^\tau(S^\tau)$is used below for the objective value of solution S^τat epoch τ. A smaller value represents a better stage solution.

4.3 Adaptive Large Neighborhood Search

4.3.1 Initial-solution construction

A service-first feasible greedy procedure is used to construct the initial solution.

An empty route is first created for each available vehicle. The active orders are then processed sequentially. For each order, all insertion positions in all available routes are examined. A candidate position is feasible only if the resulting route satisfies Equation (29).

Let $S^\tau \oplus (i, k, h)$denote the solution obtained by inserting order $i$into position $h$of vehicle k's route. The incremental cost and the selected insertion position are:

$$\begin{aligned} \Delta F_{ikh}^\tau &= F^\tau(S^\tau \oplus (i,k,h)) - F^\tau(S^\tau), \\ (k^*, h^*) &= \arg\min_{k,h}\{\Delta F_{ikh}^\tau \mid S^\tau \oplus (i,k,h) \text{ is feasible}\}. \end{aligned} \tag{30}$$

Order $i$is inserted into position $h^*$of vehicle k^*. If no feasible insertion exists and the current epoch is not the final epoch, the order is placed in $\mathcal{P}^\tau$.

Postponement is treated as a remedy for infeasibility rather than as an ordinary cost-saving choice. An order cannot be postponed when at least one feasible insertion exists. The same rule is applied during the repair stage.

The resulting greedy solution is used as both the initial current solution and the initial stage-best solution. RH-ALNS and D-ALNS start from the same solution when they are tested on the same scenario.

4.3.2 Standard destroy and repair operators

Each search iteration selects one destroy operator and one repair operator. The destroy operator removes part of the currently served orders, and the repair operator reinserts them into feasible positions.

Let n_{ser}^{τ}be the number of served orders in the current solution and ηbe the removal ratio. The number of removed orders is:

$$q^{\mathrm{rem}} = \max\{1, \lfloor \eta n_{\mathrm{ser}}^{\tau} \rfloor\}. \tag{31}$$

The standard operator pool is shown in Table 3.

Table 3
The general meaning of ordinary operators

Type	Operator	Description
Destroy	Random removal D1	Removes randomly selected served orders.
	Worst-cost removal D2	Removes orders with large marginal objective contributions
	Related removal D3	Removes geographically or temporally related customers around a randomly selected seed customer.
Repair	Greedy insertion R1	Repeatedly inserts the unassigned customer–position pair with the minimum feasible incremental cost.
	Regret-2 insertion R2	Prioritizes the customer with the largest difference between its best and second-best feasible insertion costs.

Random removal (D1) selects q^{rem}served orders without considering their route positions or objective contributions. The selected orders are placed in a temporary unassigned set.

Its purpose is diversification. Random changes allow the search to explore route structures that may not be reached when removal is based only on cost.

Worst-cost removal (D2) gives priority to orders whose removal produces a large reduction in the current objective. The removal gain of order $i$is:

$$\Delta_i^{\mathrm{rem}} = F^{\tau}(S^{\tau}) - F^{\tau}(S^{\tau} \setminus i), \tag{32}$$

where $S^{\tau} \setminus i$denotes the solution after removing order $i$from its route.

Orders with larger values of $\Delta_i^{\mathrm{rem}}$are removed earlier. Such orders generally cause long detours, high tardiness, or inefficient vehicle use.

Related removal (D3) begins with a seed order and removes orders that are close to it in geographical location or have similar time-window characteristics.

Instead of changing isolated customers, the operator removes a group of related orders and allows several nearby route segments to be rebuilt together.

Greedy insertion (R1) uses the incremental cost in Eq. (30). At each step, the feasible order-position pair with the smallest incremental cost is selected. The insertion costs of the remaining orders are recalculated after the route set changes.

An order is postponed only if no feasible insertion is available.

Regret-2 insertion (R2): Greedy insertion may use a favorable position for a flexible order while leaving a less flexible order without a feasible alternative. Regret-2 insertion accounts for this difference.

Let $\Delta_i^{(1)}$and $\Delta_i^{(2)}$be the smallest and second-smallest feasible insertion costs of order i. Its regret value is:

$$\text{Regret}_i = \Delta_i^{(2)} - \Delta_i^{(1)}. \quad (33)$$

Orders with larger regret values are inserted first. If only one feasible insertion is available, the order is treated as highly urgent in the repair sequence.

4.3.3 Local search

After repair, the candidate solution is further examined using three local-search neighborhoods.

Intra-route relocation removes an order from its current position and inserts it elsewhere in the same route. This changes the visiting sequence but not the assigned vehicle. Inter-route relocation transfers an order from one route to a feasible position in another route. It can improve load allocation and reduce both detours and tardiness.

The 2-opt operation reverses a segment of one route. It is mainly used to remove route crossings and shorten travel distance. A best-improvement rule is adopted. All feasible moves in the current neighborhood are evaluated, and the move producing the largest strict reduction in $F^{\tau}(S^{\tau})$ is accepted. The procedure stops when no improving move is found or when the preset local-search limit is reached.

Local search does not change the postponed-order set. It also cannot modify routes that were committed in earlier epochs.

4.4 Dynamic-Order-Oriented Operators

Standard destruction and repair operators use route structure and objective contribution but do not directly exploit current time pressure or postponement history. D-ALNS adds three operators for that information: time-pressure removal, time-priority insertion, and postponed-priority insertion.

4.4.1 Time-pressure removal

Time-pressure removal focuses on served orders with limited temporal flexibility. The residual slack of order $i$is:

$$\text{Slack}_i^{\tau} = l_i - B_i^{\tau}. \quad (34)$$

A small slack indicates that the order is close to, or has already passed, its latest preferred service time. Orders with smaller slack are removed earlier and reconsidered during repair.

This operation gives the search an opportunity to move time-critical orders to earlier route positions or assign them to vehicles with more suitable schedules.

4.4.2 Time-priority insertion

Time-priority insertion first determines which unassigned order should be handled and then chooses its best feasible insertion position.

Orders with less remaining time before $l_i$receive higher priority. The order priority $p_i$is also considered, so that an urgent high-priority order is not delayed by a more flexible order. After an order has been selected, its feasible positions are evaluated using the incremental cost in Equation (30).

This differs from greedy insertion, which compares all order-position pairs directly. Time-priority insertion protects urgent orders before selecting their route positions.

4.4.3 Postponed-priority insertion

Postponed-priority insertion gives precedence to orders with larger values of v_i^{τ}. These orders have already remained unfinished across one or more decision epochs and face increasing postponement costs.

When several orders have the same postponement count, time urgency and feasible insertion cost are used to determine their order of processing. The postponement-cost function itself is unchanged; historical information is used only to guide the repair sequence.

4.5 Difference between RH-ALNS and D-ALNS

RH-ALNS and D-ALNS use the same rolling-horizon process, initial solution, route evaluator, local search, adaptive operator weights, and simulated-annealing rule.

RH-ALNS uses the standard operators D1–D3 and R1–R2. D-ALNS adds time-pressure removal, time-priority insertion, and postponed-priority insertion. The comparison between the two methods therefore examines the effect of the dynamic-order-oriented operators while keeping the other algorithm components unchanged.

Figure 4 illustrates the removal of orders with small slack, the priority protection of urgent orders, and the earlier reinsertion of orders with larger postponement counts.

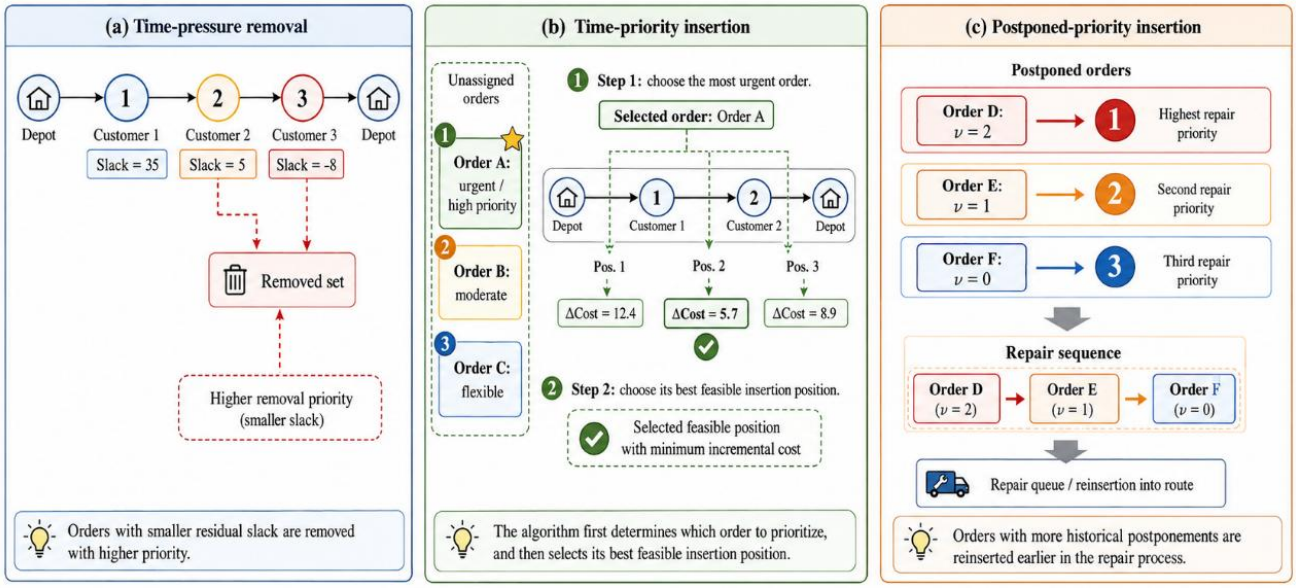

Fig. 4. Mechanisms of the dynamic-order-oriented destroy and repair operators

4.6.1 Adaptive operator weights

No destruction or repair operator performs best in every search state. D-ALNS adjusts their selection probabilities according to their recent results.

Let w_o^g be the weight of operator o during update segment g. Its selection probability is:

$$P_o^g = \frac{w_o^g}{\sum_{h\in\mathcal{O}} w_h^g}, \tag{35}$$

where $\mathcal{O}$ denotes the relevant destroy or repair operator pool. Destroy and repair operators are selected separately.

After each iteration, the selected operators receive a reward according to the result of the candidate solution. A new stage-best solution receives the highest reward. Smaller rewards are assigned when the current solution is improved, an equal solution is accepted, or a worse solution is accepted by simulated annealing. A rejected candidate receives no positive reward.

Let π_o^g be the cumulative reward of operator o in segment g, n_o^g its number of uses, ρ the reaction factor, and $w_{\min}$ the minimum weight. The update rule is:

$$w_o^{g+1} = \begin{cases} \max\left\{w_{\min}, (1-\rho)w_o^g + \rho\dfrac{\pi_o^g}{n_o^g}\right\}, & n_o^g > 0, \\ w_o^g, & n_o^g = 0. \end{cases} \tag{36}$$

The lower bound $w_{\min}$prevents an operator from permanently losing the chance to be selected after a period of weak performance.

4.6.2 Simulated-annealing acceptance

Restricting the search to improving solutions can lead to early convergence. Simulated annealing is therefore used to accept some worse candidates.

Let $\Delta F = F^\tau(S_{\text{cand}}^\tau) - F^\tau(S_{\text{cur}}^\tau)$. The candidate is accepted with probability:

$$P_{\text{acc}} = \begin{cases} 1, & \Delta F \le 0, \\ \exp(-\dfrac{\Delta F}{T}) & \Delta F > 0, \end{cases} \tag{37}$$

where $T$is the current temperature.

The temperature is reduced after every search iteration:

$$T \leftarrow \max\{T_{\min}, \alpha T\}, \tag{38}$$

where $\alpha$is the cooling rate and $T_{\min}$is the minimum temperature.

The current solution and the stage-best solution are maintained separately. A worse candidate accepted by simulated annealing may replace the current solution, but the stage-best solution is updated only when a strictly smaller objective value is obtained

4.7 Overall D-ALNS Procedure

The full rolling-horizon procedure is given in Algorithm 1.

Algorithm 1: Rolling-Horizon D-ALNS Procedure

Input: Customer orders, vehicle data, cost parameters, rolling decision times, and D-ALNS parameters
Output: Vehicle routes committed at each epoch and the accumulated objective value

Initialize order lifecycle states and vehicle execution states
for τ = 0, 1, ..., $\bar{\tau}$ do
    Advance vehicle execution states to decision time t^τ
    Release orders satisfying $r_i \le t^\tau$ that have not yet entered the system
    Merge newly released and previously postponed orders to construct N^τ
    Construct the available-vehicle set K^τ from vehicle return states
    if $N^\tau = \emptyset$ then
        Continue to the next decision epoch
    end if
    Construct a service-first feasible greedy initial solution S_0
    Set $S_{cur} \leftarrow S_0$ and $S_{best} \leftarrow S_0$
    Initialize destroy- and repair-operator weights
    Initialize the simulated-annealing temperature T
    for g = 1, 2, ..., G_{max} do
        Select one destroy operator according to adaptive weights
        Select one repair operator according to adaptive weights
        Remove q^{rem} served orders from S_{cur}
        Repair the partial solution
        Apply local search to the repaired candidate
        Evaluate $F^\tau(S_{cand})$ using the unified evaluator
        if $F^\tau(S_{cand}) < F^\tau(S_{best})$ then

22: $S_{best} \leftarrow S_{cand}$
23: end if
24: Accept or reject S_{cand} using the simulated-annealing criterion
25: Assign rewards to the selected operators
26: Update operator weights at the end of each update segment
27: Update the simulated-annealing temperature
28: end for
29: Commit the vehicle routes contained in S_{best}
30: Change dispatched vehicles to the in-transit state
31: Transfer P^{τ} to the postponed-order set of the next epoch
32: Update postponement counts and vehicle availability times
33: end for
34: Continue executing committed routes until all orders are completed
35: Return the accumulated objective value and all committed routes

Only information available at t^{τ} is used in the current search. Once the routes in S_{best} are committed, they remain unchanged in subsequent epochs.

The service-first rule is applied in both initial construction and repair. An order is postponed only when no feasible insertion exists and the current epoch is not the final one.

5.Numerical Experiments

The numerical study covers common-parameter calibration, formal algorithm comparison, convergence, operator ablation, and sensitivity to the dynamic order ratio. Shared RH-ALNS and D-ALNS parameters are calibrated first through an orthogonal design and independent confirmation. The formal comparison then evaluates six algorithms at several customer scales, followed by analyses of search trajectories, dynamic-specific operators, and changes in order dynamism.

All reported results are obtained from complete rolling-horizon simulations. At each decision epoch, an algorithm can access only currently released orders and currently available vehicles. Future orders remain invisible to the solver. Once a route is committed, it is frozen and cannot be modified in subsequent epochs. The accumulated objective over the planning horizon comprises transportation cost, fixed dispatch cost, weighted tardiness cost, and postponement cost.

5.1 Experimental Setup

5.1.1 Test instances and dynamic scenarios

The experimental instances are constructed from the Solomon benchmark [32]. The main comparison uses the first 40, 60, and 80 customers of RC101. Customer coordinates, demands, service durations, and time-window information are inherited from the original instance. Euclidean distance is used, and vehicle speed is set to 1, such that travel time is numerically equal to travel distance.

The fleet contains 25 homogeneous vehicles, each with a capacity of 200. The planning horizon is 240 times units and contains six decision epochs at 0, 40, 80, 120, 160, and 200. After the last decision epoch, committed routes continue to be executed, but no additional dispatch decision is generated.

The dynamic order ratio is set to 40% in the main comparison. Sixty percent of the orders are available at the initial epoch, whereas the remaining orders are released dynamically. Ten scenarios are generated for each customer scale using seeds 30–39. Each algorithm is therefore evaluated in 30 formal rolling-horizon runs.

The latest preferred service time is treated as a soft time constraint. Service after this time is allowed but incurs weighted tardiness cost. An active order may be postponed only when no feasible insertion exists, and postponement is prohibited in the final decision epoch. Table 4 is experimental instance and model-parameter settings.

Table 4
Experimental instance and model-parameter settings

Parameter	Parameter level	Reference
Maximum vehicle capacity (Q)	200	[5]
End of the planning horizon (H)	240	N/A
Rolling decision times (t^{τ})	{0,40,80,120,160,200}	[33]
Vehicle travel speed (v)	1	N/A
Initial-order proportion $(1-\rho)$	60%	N/A
Dynamic-order ratio	{0.20,0.40,0.60}	[34]
Unit weighted tardiness cost (c_L)	1.0	[6]
Basic postponement cost coefficient (c_p)	1.0	[6]
Fixed dispatch cost per activated vehicle (c_F)	1.0	[6]

Notes: The customer coordinates, demands, ready times, due dates, service durations, fleet size, vehicle capacity, and planning horizon are inherited from the Solomon RC101 benchmark

All computations for the algorithms are implemented in Python 3.12. We conduct all experiments on a personal computer running Windows 11. The hardware includes an Intel(R) Core(TM) i5-10500H CPU @ 2.50 GHz and 16 GB RAM.

5.1.2 Comparison algorithms

Six algorithms are included in the main comparison. GREEDY denotes the service-first feasible greedy construction procedure. It inserts active orders into feasible route positions according to incremental cost and postpones an order only when no feasible insertion exists. GA is a genetic algorithm using order crossover and three mutation operators: swap, relocation, and inversion. TS is a tabu-search algorithm using the same three neighborhood structures and an aspiration criterion. ILS is an iterated local search that combines deterministic best-improvement local search with perturbation. RH-ALNS is the rolling-horizon ALNS baseline containing only the standard destroy and repair operators. D-ALNS extends RH-ALNS by adding time-pressure removal, time-priority insertion, and postponed-priority insertion.

GA, TS, and ILS stop according to newly evaluated complete candidates, whereas RH-ALNS and D-ALNS stop by ALNS iteration count. These units are not computationally identical; solver CPU time is therefore reported separately, and the study does not claim exact equality of low-level search effort.

5.1.3 Evaluation of metrics and statistical tests

The total objective over the complete planning horizon is the primary performance measure. Smaller values indicate better solutions. Transportation, fixed dispatch, weighted tardiness, and postponement costs are also reported.

The paired gap of algorithm $a$relative to D-ALNS on scenario s is as follows:

$$\mathrm{Gap}_{a,s} = \frac{F_{a,s} - F_{\text{D-ALNS},s}}{F_{\text{D-ALNS},s}} \times 100\%$$

A positive value indicates that algorithm $a$is worse than D-ALNS, whereas a negative value indicates that it performs better on the same scenario.

5.2 Common Parameter Fine-Tuning for RH-ALNS and D-ALNS

5.2.1 Factors and orthogonal design

RH-ALNS and D-ALNS share the rolling-horizon structure, adaptive weight mechanism, simulated-annealing acceptance criterion, and local search. Their principal difference is the composition of the operator pool. To preserve a fair comparison, the shared search parameters are tuned jointly and are kept identical in all subsequent experiments.

Four factors are selected: A: destroy fraction range; B: operator-weight reaction factor; C: initial temperature; D: cooling rate, Table 5.

Table 5
Factors and levels

Factor	Parameter level	Level 1	Level 2	Level 3
A	Destroy fraction range	0.05–0.15	0.10–0.30	0.20–0.40
B	Reaction factor	0.10	0.20	0.30
C	Initial temperature	10.0	20.0	40.0
D	Cooling rate	0.90	0.95	0.98

An $L_9(3^4)$orthogonal design is used. The tuning set contains C101, R101, and RC101 scenarios generated using seeds 24 and 25, resulting in six instances. Each of the nine parameter configurations is evaluated using both RH-ALNS and D-ALNS, giving $9 \times 6 \times 2 = 108$ tuning runs.

For each algorithm-instance block, the best objective observed among the nine configurations is denoted by $F_{a,i}^{\min}$. The normalized objective gap of configuration $q$is

$$G_{q,a,i} = \frac{F_{q,a,i} - F_{a,i}^{\min}}{F_{a,i}^{\min}}.$$

Smaller values indicate that the configuration is closer to the best observed result for that algorithm-instance block, Table 6.

Table 6
Results of the $L_9(3^4)$orthogonal experiment

Configuration	A	B	C	D	Mean normalized gap (%)	Worst gap (%)	Mean runtime (s)	Rank
L9-1	0.05–0.15	0.10	10	0.90	6.892	14.876	57.09	8
L9-2	0.05–0.15	0.20	20	0.95	7.000	14.876	58.02	9
L9-3	0.05–0.15	0.30	40	0.98	6.470	12.710	57.21	7
L9-4	0.10–0.30	0.10	20	0.98	2.189	6.035	156.74	4
L9-5	0.10–0.30	0.20	40	0.90	3.102	6.813	164.86	6
L9-6	0.10–0.30	0.30	10	0.95	2.343	8.336	171.84	5
L9-7	0.20–0.40	0.10	40	0.95	0.988	2.068	320.76	2
L9-8	0.20–0.40	0.20	10	0.98	0.955	4.013	307.87	1
L9-9	0.20–0.40	0.30	20	0.90	1.489	4.491	327.59	3

All 108 tuning runs are feasible, with no failed run or audit violation. L9-8 obtains the smallest mean normalized gap among the configurations included in the orthogonal design.

5.2.2 Main-effect analysis

The combined main effects of the shared parameters are shown in Figure 5.

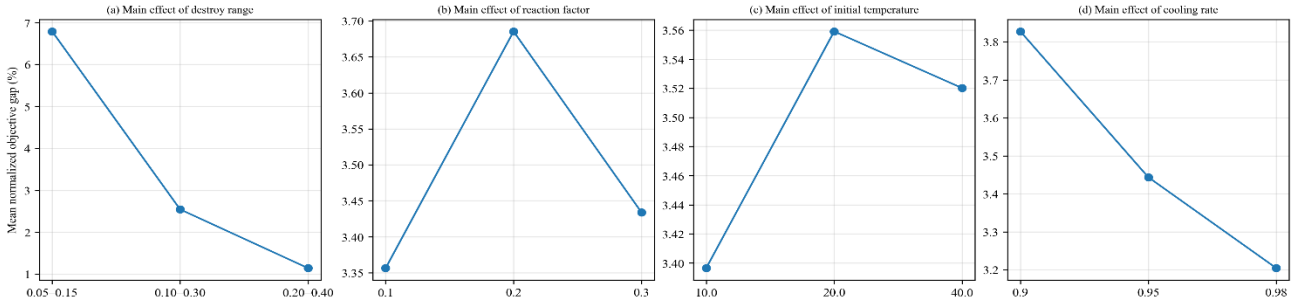

Fig. 5. Combined main-effect plots of the common RH-ALNS and D-ALNS parameters

The destroy fraction range has the strongest effect. Increasing it from 0.05–0.15 to 0.10–0.30 reduces the mean normalized gap from 6.787% to 2.545%, and the gap further decreases to 1.144% at 0.20–0.40.

The cooling rate is the second most influential factor. The mean normalized gap decreases from 3.828% at 0.90 to 3.204% at 0.98. The reaction factor and initial temperature have smaller main effects. Their best combined levels are 0.10 and 10.0, respectively.

The influence order is

$$A > D > B > C.$$

Algorithm-specific main effects are shown in Figure 6.

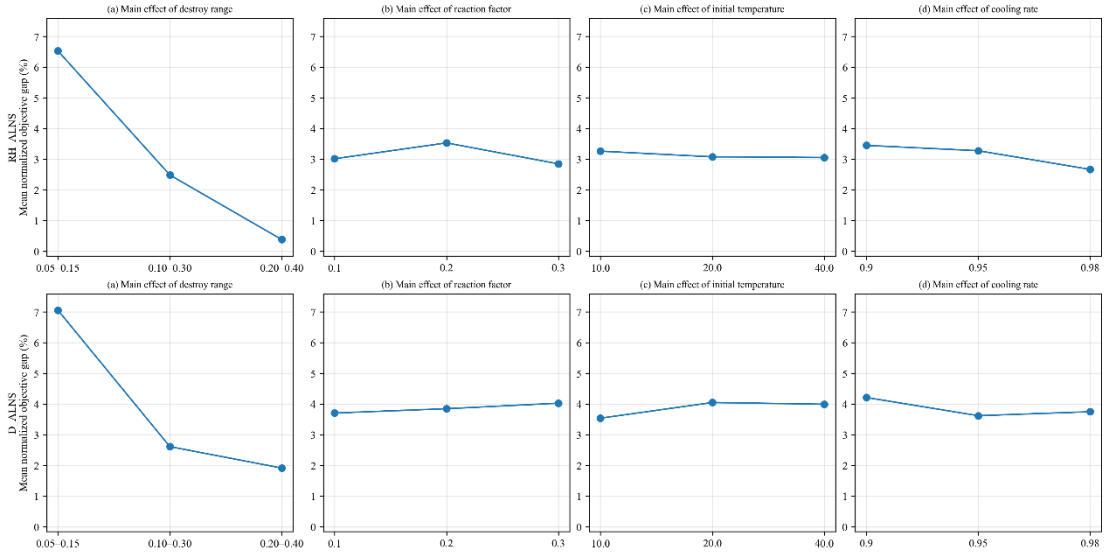

Fig. 6. Algorithm-specific main-effect plots of the RH-ALNS and D-ALNS parameters

Both algorithms favor the 0.20–0.40 destroy range. However, their preferred levels for the other factors are not identical. Selecting independent algorithm-specific settings would confound the effect of the dynamic operator pool with parameter differences. The combined main-effect prediction is therefore used, yielding A3B1C1D3.

5.2.3 Confirmation and selected parameters

Three candidate configurations are tested on independent seed-26 scenarios: the best observed orthogonal configuration; the engineering baseline; the main-effect-predicted configuration.

The three candidates were evaluated on independent seed-26 scenarios for C101, R101, and RC101 with both RH-ALNS and D-ALNS. Each candidate therefore comprised six runs, for 18 confirmation runs in total. Table 7 reports the results.

Table 7
Confirmation results

Candidate	A	B	C	D	Feasible rate	Mean normalized gap (%)	Worst gap (%)	Standard deviation (%)	Mean runtime (s)
Best	0.20–0.40	0.2	10	0.98	100%	1.038	4.294	1.498	329.4
Baseline	0.10–0.30	0.2	20	0.95	100%	0.972	2.534	1.05	175.22
Predicted	0.20–0.40	0.1	10	0.98	100%	0.571	2.039	0.777	309.14

The main-effect-predicted configuration obtains the smallest mean gap, worst gap, and standard deviation. It is therefore selected for formal experiments. Six deterministic replay runs are performed, and all six reproduce the original result signatures.

The final parameter settings are shown in Table 8.

Table 8
Selected formal parameters of RH-ALNS and D-ALNS

Parameter	Symbol	Value	Type
Destroy fraction range	A	0.20–0.40	Tuned
Reaction factor	B	0.1	Tuned
Initial temperature	C	10	Tuned
Cooling rate	D	0.98	Tuned
Iterations per nonempty epoch	N	100	Fixed
Segment length	L	10	Fixed
Minimum operator weight	w_{win}	0.05	Fixed
Local-search rounds	R_{ls}	2	Fixed
Local-search evaluation limit	E_{ls}	200	Fixed

The subsequent comprehensive comparison, convergence, ablation, and sensitivity experiments all utilize the parameter settings listed in Table 8. Apart from the composition of the operator pool, all other parameters for RH-ALNS and D-ALNS are identical.

5.3 Comprehensive Algorithm Comparison

5.3.1 Baseline parameter settings

The formal parameters for GA, TS, and ILS are shown in Table 9. These parameters are fixed prior to formal experiments.

Table 9
Parameters of the comparison metaheuristics

Algorithm	Parameter	Value
ILS	Maximum candidate evaluations per searchable epoch	100
	Perturbation strength	2
	Neighborhood sample size	10
TS	Tabu tenure ($tabu\_tenure$)	7
	Neighborhood sample size	10
	Maximum candidate evaluations per searchable epoch	100
GA	Maximum candidate evaluations per searchable epoch	100
	Population size (pop_size)	20
	Mutation rate (m_r)	0.2
	Crossover rate (c_r)	0.8

5.3.2 Solution quality at different scales

The comprehensive comparison combines three customer scales, ten scenario seeds, and six algorithms, giving 3 x 10 x 6 = 180 complete rolling runs. Every run passed the feasibility and integrity audits. Table 10 summarizes the main results.

Table 10
Algorithm comparison results at different problem sizes

n	Algorithm	Mean objective ± SD	Mean gap (%)	Mean rank	CPU time (s)
40	GREEDY	1749.00 ± 124.72	26.07	6.00	0.20
40	GA	1373.99 ± 64.99	-1.30	3.35	14.98
40	TS	1363.67 ± 70.53	-2.05	2.35	14.87
40	ILS	1412.54 ± 91.83	1.51	3.60	14.73
40	RH-ALNS	1391.73 ± 104.65	-0.23	2.75	29.01
40	D-ALNS	1395.28 ± 109.28	0.00	2.95	21.09
60	GREEDY	2329.07 ± 253.74	22.19	6.00	0.38
60	GA	1953.94 ± 111.38	2.57	3.30	33.39
60	TS	1926.94 ± 117.62	1.20	2.00	33.11
60	ILS	1970.19 ± 133.53	3.44	3.30	33.32
60	RH-ALNS	2047.52 ± 103.10	7.44	4.55	70.36
60	D-ALNS	1905.79 ± 99.56	0.00	1.85	45.07
80	GREEDY	2976.12 ± 145.18	28.15	6.00	0.63
80	GA	2573.46 ± 121.12	10.79	4.30	57.46
80	TS	2505.95 ± 117.15	7.79	3.40	57.65
80	ILS	2575.03 ± 151.27	10.82	4.30	57.20
80	RH-ALNS	2351.53 ± 112.83	1.16	1.65	171.40
80	D-ALNS	2326.43 ± 125.06	0.00	1.35	106.43

Table 10 compares the six algorithms at three customer scales. For $n = 40$, TS gives the lowest mean objective value of 1363.67, followed by GA at 1373.99. RH-ALNS and D-ALNS obtain 1391.73 and 1395.28, respectively, so their difference is small. At $n = 60$, D-ALNS performs best, with a mean objective value of 1905.79. TS ranks second at 1926.94, while RH-ALNS increases to 2047.52, corresponding to a gap of 7.44% relative to D-ALNS. For $n = 80$, D-ALNS again gives the lowest mean objective value, at 2326.43. RH-ALNS is the closest method, with a gap of 1.16%, while the gaps of TS, GA, and ILS are 7.79%, 10.79%, and 10.82%. GREEDY performs worst at all three scales. The results show that D-ALNS has no clear advantage for the 40-customer instances, but its performance becomes stronger at $n = 60$ and $n = 80$.

Figure 7 compares the distributions of the objective-value gaps between the comparison algorithms and D-ALNS for three problem sizes $n = 40$, $n = 60$, and $n = 80$. The boxes show the variation of the gap values across the ten scenarios. The line inside each box is the median, the whiskers indicate the non-outlier range, and the circles denote outliers. The dashed horizontal line marks a zero gap. Positive values mean that the comparison algorithm has a higher objective value than D-ALNS, while negative values mean that it performs better. The percentages above the boxes report the corresponding mean gap magnitudes.

For $n = 40$, GREEDY has the largest gap, with a mean value of 26.07% and a wide distribution. The other four algorithms remain close to D-ALNS. The mean gap magnitudes of GA, TS, ILS, and RH-ALNS are 1.30%, 2.05%, 1.51%, and 0.23%, respectively. The signed results in Table 10 show that GA, TS, and RH-ALNS obtain slightly lower objective values than D-ALNS at this scale. For $n = 60$, the pattern changes. GREEDY still has a large mean gap of 22.19%, while the gaps of GA, TS, and ILS are 2.57%, 1.20%, and 3.44%. RH-ALNS rises to 7.44%, making its difference from D-ALNS much clearer than at $n = 40$. For $n = 80$, GREEDY again shows the largest gap at 28.15%. GA and ILS exceed 10%, and TS records 7.79%. RH-ALNS returns to a much smaller gap of 1.16%.

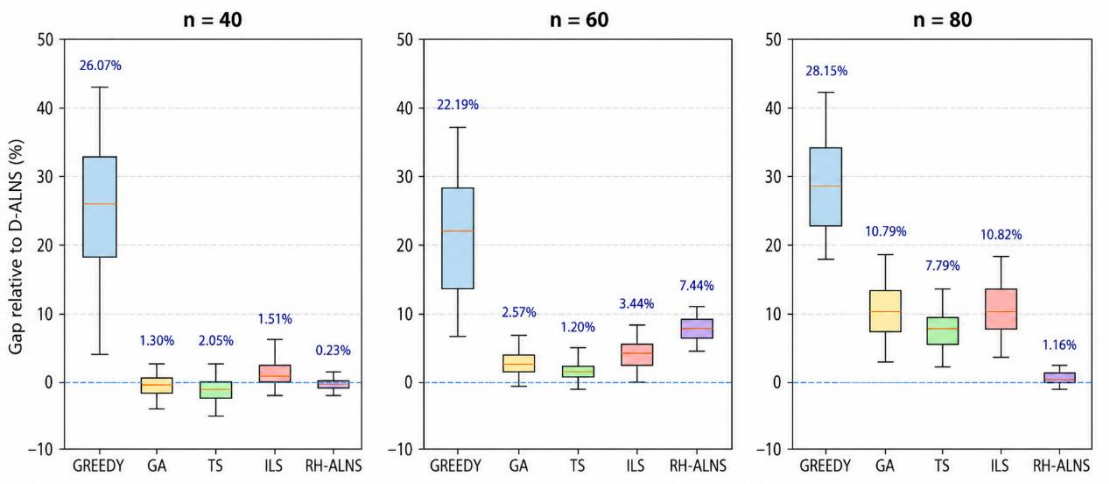

Fig. 7. Gap distributions relative to D-ALNS

The three panels show that the relative performance changes with problem size. GREEDY remains far from D-ALNS in all three cases. GA and ILS also show larger gaps as the scale increases. TS is competitive at $n = 40$ and $n = 60$, but its gap becomes larger at $n = 80$. RH-ALNS is close to D-ALNS at $n = 40$ and $n = 80$, while a larger difference appears at $n = 60$. D-ALNS therefore does not dominate at every scale, but its performance is more favorable in the medium- and large-scale instances.

5.3.3 Overall ranking

Table 11 reports average ranks over the 30 matched scale-scenario blocks.

Table 11 shows that D-ALNS obtains the lowest overall mean rank of 2.05. TS follows with 2.58, while RH-ALNS ranks third at 2.98. GA and ILS perform less favorably overall, with mean ranks of 3.65 and 3.73, and GREEDY remains last. The mean objective gaps show a similar pattern. Compared with D-ALNS, the overall gaps are 2.31% for TS, 2.79% for RH-ALNS, 4.02% for GA, and 5.26% for ILS.

Table 11
Overall ranks

Algorithm	Mean rank	Median rank	Mean gap (%)
D-ALNS	2.05	2	0
TS	2.58	3	2.31
RH-ALNS	2.98	3	2.79
GA	3.65	4	4.02
ILS	3.73	4	5.26
GREEDY	6	6	25.47

The advantage of D-ALNS is not equally strong at every problem size. For the 40-customer instances, several algorithms remain close to D-ALNS and some even obtain slightly lower objective values. The difference becomes clearer at larger scales. At $n = 60$, RH-ALNS is 7.44% higher than D-ALNS in mean objective value, and D-ALNS also retains a 1.16% advantage over RH-ALNS at $n = 80$. This suggests that the dynamic-specific operators are more useful when the routing problem becomes more difficult, although their contribution varies with problem size.

5.4 Convergence Analysis

For the convergence experiments, four instances generated using seeds 30–33 with a scale of 60 customers were selected, and the "epoch 0" phase was analyzed. Since there were no pre-existing routes or variations in vehicle status at this stage, the different algorithms operated on the same set of active orders and initial states.

GA, TS, and ILS used the number of candidate evaluations as the base unit of search, whereas RH-ALNS and D-ALNS used the number of ALNS iterations. To facilitate cross-algorithm comparison, the horizontal axis was standardized to a normalized search progress of 0%–100%, though this does not imply an identical underlying computational budget, Table 12.

Table 12
Convergence performance

Algorithm	Objective	Improvement	Normalized	Progress	CPU time	Final CPU time (s)
GA	896.89	22.97	27.43	66	16.95	25.54
TS	819.26	28.91	13.8	17	4.5	25.18
ILS	892.6	23.18	24.75	47	12.1	25.3
RH-ALNS	778.96	32.66	4.16	26	17.7	67.34
D-ALNS	778.89	32.46	4.01	9.5	5.27	41.59

Notes: objective; Mean final objective；Improvement; Mean improvement (%)；Normalized; Normalized AU；Progress; Progress to 90% improvement (%); CPU time; CPU time to 90% improvement (s)

Figure 8 presents the normalized convergence processes of five heuristic algorithms for the n=60 instances. Four instances generated using seeds 30–33 are considered, and the analysis focuses on epoch 0 of the rolling-horizon process. The horizontal axis represents normalized search progress. Candidate-solution evaluations are used as the original search unit for GA, TS, and ILS, whereas ALNS iterations are used for RH-ALNS and D-ALNS. These measures are converted to a common range of 0%–100% only to facilitate the comparison of convergence patterns; they should not be interpreted as identical computational budgets. The vertical axis denotes the gap between the incumbent best objective value and the best final objective observed in the experiment. Lower values indicate that the current solution is closer to the best observed result. Distinct marker shapes identify the algorithms and are displayed at fixed search-progress intervals.

The convergence curves show that D-ALNS and RH-ALNS reduce the objective gap rapidly during the early search stage and outperform GA, TS, and ILS over most of the search process. D-ALNS obtains the lowest normalized AUC, at 4.01, and reaches 90% of its total improvement after only 9.5% of the normalized search process, corresponding to 5.27 s of CPU time. In comparison, RH-ALNS has a normalized AUC of 4.16 and requires 26.0% of the search process and 17.70 s to reach the same improvement level. Their mean final objective values are 778.89 and 778.96, respectively, indicating almost identical final solution quality. Nevertheless, D-ALNS produces high-quality solutions considerably earlier and therefore demonstrates better early-stage search efficiency.

TS performs best among the remaining methods, with a normalized AUC of 13.80. It reaches 90% of its total improvement at 17.0% of normalized search progress and finishes with a mean objective

of 819.26. GA and ILS converge more slowly: their AUC values are 27.43 and 24.75, and they require 66.0% and 47.0% of the search process, respectively, to reach the same improvement threshold. Both also finish above the ALNS variants. D-ALNS thus combines competitive final quality with faster early improvement, consistent with the intended role of its dynamic-order-oriented operators.

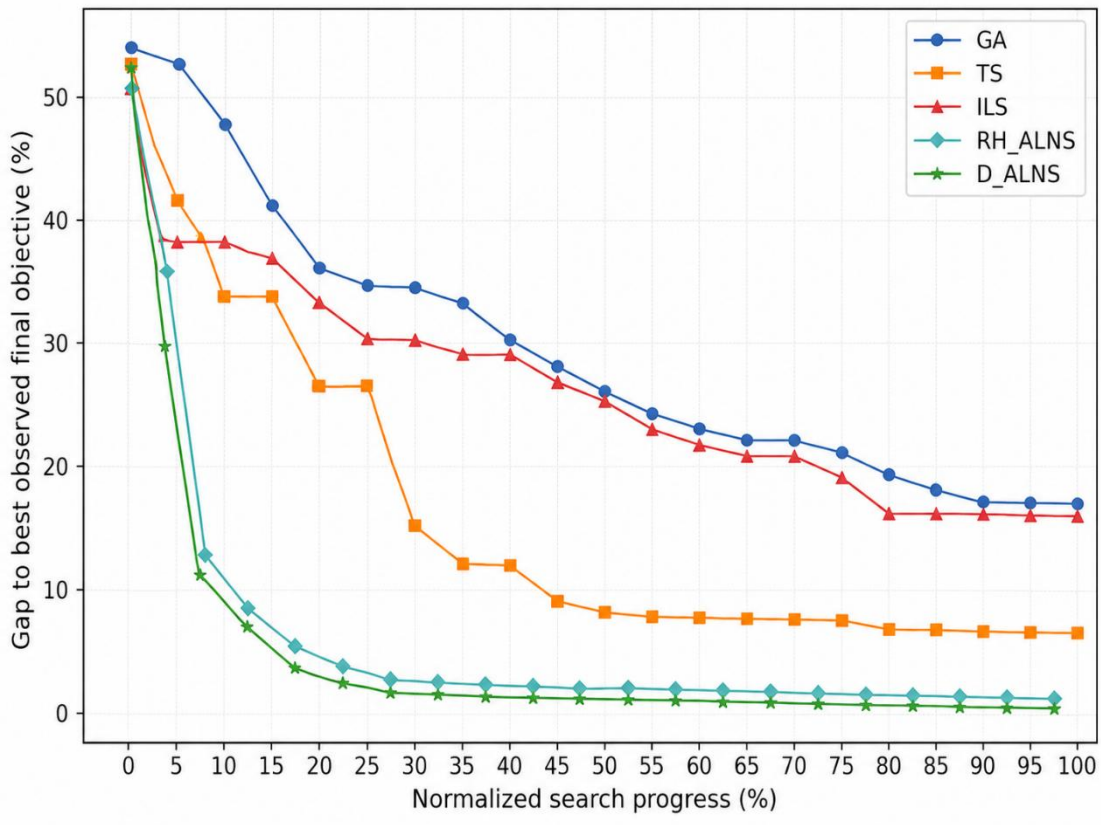

Fig. 8. Mean convergence gap versus normalized search progress

5.5 Ablation Study on Dynamic-Specific Operators

The ablation experiment isolates the contribution of the three dynamic-specific operators through single-factor removal. It fixes the customer count at 60 and the dynamic order ratio at 40%, using the ten scenarios generated by seeds 30-39. Every variant executes the full rolling horizon, from order release and vehicle-state updates through stage planning and route execution. Within each scenario, customer and vehicle data, release processes, cost parameters, and decision epochs are identical; only the active operator pool changes.

The complete D-ALNS adds three operators for dynamic orders to the standard RH-ALNS operator pool. The Time-pressure removal (TPR) operator prioritizes removing orders with higher time pressure from the current plan based on their remaining service time and the urgency of their time windows, providing space for subsequent route reconfiguration. The Time-priority insertion (TPI) operator comprehensively considers order priority, remaining service time, and insertion increment cost during the repair process, allowing urgent orders to be prioritized for entry into feasible routes. The Postponed-priority insertion (PPI) operator improves the insertion order of already delayed orders in subsequent stages to reduce the possibility of consecutive delays for the same order.

Using these three operators, five algorithm variants are designed. D-ALNS FULL is a complete algorithm including TPR, TPI, and PPI; w/o TPR removes only the time-pressure removal operator; w/o TPI removes only the time-priority insertion operator; w/o PPI removes only the postponed-priority insertion operator; RH-ALNS does not include any dynamic-specific operators, retaining only

standard operators such as random removal, worst-cost removal, related removal, greedy insertion, and regret-2 insertion. Importantly, the ablation variants do not set the initial weight of the removed operator to 0, but rather directly remove the operator from the candidate operator pool and renormalize the selection probability among the remaining operators. This prevents the removed operator from re-entering the search process through weight updates or abnormal rollbacks.

All five variants use the parameters in Table 8, including the destroy-fraction range, weight reaction factor, initial temperature, cooling rate, iterations per stage, and local-search budget. Fixing these settings makes operator removal, rather than parameter variation, the main source of performance differences.

The experiments employed a paired block design. For dynamic scenarios generated from the same seed, the five algorithm variants used the same order data, vehicle data, decision stages, and cost coefficients, thus forming a complete paired block. Ten scenarios resulted in ten paired blocks. The results of the complete D-ALNS and RH-ALNS experiments were directly reused from the formal results under the same instance and parameter conditions in the comprehensive comparative experiment. The three single-operator ablation variants were re-run on each of the ten scenarios, resulting in 30 additional complete rolling experiments and ultimately 50 algorithm-scenario results. All runs completed 60 customer orders without any constraint violations, future information usage errors, vehicle status errors, route modifications, or objective value reconciliation errors.

To measure the change in the target value after removing a certain operator, the pairing gap of variant a relative to the full D-ALNS in scene s is defined as follows:

$$Gap_{a,s} = \frac{F_{a,s} - F_{D-ALNS,s}}{F_{D-ALNS,s}} \times 100\%$$

Here, $F_{a,s}$ represents the objective value over the full planning horizon for variant a in scene s, while $F_{D-ALNS,s}$ represents the objective value for the full D-ALNS algorithm in the same scenario. A gap greater than 0 indicates that the objective value increased after removing the corresponding operator, implying that the operator plays a positive role in that scenario; a gap less than 0 indicates that the ablation variant achieved a lower objective value; and a gap of 0 indicates that both algorithms yielded the same result. All instances with negative gaps or identical results were retained in the experiment; no data were truncated or filtered out.

Table 13
Dynamic-operator ablation results

Variant	Mean objective ± SD	Gap to full D-ALNS (%)	Mean rank	CPU time (s)
D-ALNS FULL	1905.79 ± 99.56	0	1.5	45.07
w/o TPR	1962.14 ± 104.54	2.96	2.9	45.62
w/o TPI	1997.70 ± 103.22	4.82	3	54.14
w/o PPI	2013.52 ± 98.21	5.67	3.6	55.7
RH-ALNS	2047.52 ± 103.10	7.44	4	70.36

Table 13 shows that the complete D-ALNS achieves the best overall performance among the five variants, with a mean objective value of 1905.79 ± 99.56, a mean rank of 1.50, and an average CPU time of 45.07s. Removing each dynamic-specific operator leads to a higher mean objective value under the tested scenarios. Among the single-operator ablations, removing the postponed-priority insertion operator produces the largest deterioration. The mean objective value of w/o PPI increases to 2013.52 ± 98.21, corresponding to a 5.67% gap relative to the complete D-ALNS, while its mean rank decreases to 3.60. Removing the time-priority insertion operator results in a mean objective value of 1997.70 ± 103.22, a gap of 4.82%, and a mean rank of 3.00. The effect of removing the

time-pressure removal operator is comparatively smaller, although its mean objective value still increases to 1962.14 ± 104.54, with a gap of 2.96% and a mean rank of 2.90.

RH-ALNS, which excludes all three dynamic-specific operators, performs worst, with a mean objective value of 2047.52 ± 103.10, a 7.44% gap relative to the complete D-ALNS, a mean rank of 4.00, and the longest average CPU time of 70.36 s. Under the current experimental setting, since no actual postponement occurs in these scenarios, the result for PPI should not be interpreted as direct evidence of the value of postponement-history information., followed by time-priority insertion and time-pressure removal. The combined use of the three dynamic-specific operators improves not only the final objective value but also the overall ranking and computational efficiency. These results indicate that incorporating order urgency and historical postponement information enhances the ability of D-ALNS to respond effectively to dynamic urban distribution scenarios. The individual gap values should not be added directly, because the operators may have complementary effects when used jointly.

5.6 Sensitivity to the Dynamic-Order Ratio

The sensitivity experiment treats the dynamic order ratio as its only factor while holding customer count and algorithm parameters fixed. It examines how revealing a larger share of orders after the initial epoch changes total and time-related costs, and whether the relative performance of RH-ALNS and D-ALNS changes with that share.

The dynamic order ratio is defined as

$$\rho = \frac{| N^{\mathrm{dyn}} |}{| N |}$$

where N denotes the complete set of customer orders and N^{dyn} represents the subset of orders released after the initial decision epoch.

The number of customers is fixed at 60, while the dynamic order ratio is set to 20%, 40%, and 60%, representing low, medium, and high levels of order dynamism, respectively. The corresponding numbers of initially available and dynamically released orders are shown in Table 14.

Table 14
Experimental design for the dynamic order ratio sensitivity analysis

Dynamic order ratio	Orders available initially	Dynamically released orders	Scenario seeds	Algorithms
20%	48	12	30–39	RH-ALNS, D-ALNS
40%	36	24	30–39	RH-ALNS, D-ALNS
60%	24	36	30–39	RH-ALNS, D-ALNS

Ten scenarios are generated for each dynamic order ratio using seeds 30–39. For scenarios with the same seed, the customer scale, coordinates, demands, service durations, time windows, fleet size, vehicle capacity, planning horizon, and decision epochs are kept unchanged. The only experimental change is the dynamic order ratio and the resulting division between initially available and dynamically released orders. Holding these attributes fixed limits variation unrelated to the experimental factor.

Only RH-ALNS and D-ALNS are considered here. Rather than repeating the six-algorithm benchmark, the analysis asks whether dynamic-specific operators change the behavior of standard rolling-horizon ALNS as order dynamism varies. GREEDY, GA, TS, and ILS are therefore not rerun.

Both algorithms use the formal parameter settings reported in Table 8, including the same destroy fraction range, reaction factor, initial temperature, cooling rate, number of iterations per epoch, and local-search budget. The only methodological difference is that D-ALNS contains the three

dynamic-specific operators. No parameter retuning is carried out for different dynamic order ratios, so that any observed performance change can be attributed to the scenario dynamics and operator structure rather than to parameter adjustment.

Each run covers the complete rolling-horizon process. At every decision epoch, the system first releases all orders that have arrived by the current time and updates the vehicle states. The algorithm then constructs and commits a dispatch plan for the currently active orders. Orders that have not yet been released remain invisible to the solver, and previously committed routes cannot be modified in subsequent epochs. This process continues until the final decision epoch and the end of the planning horizon.

The 20 formal runs for the 40% dynamic order ratio are directly reused from the comprehensive experiment because they use the same instances, scenario seeds, algorithm parameters, and implementation. The 20% and 60% settings each require 20 additional runs. Therefore, the number of newly executed runs is $2\text{ratios} \times 10\text{scenarios} \times 2\text{algorithms} = 40$. The complete sensitivity experiment contains $3\text{ratios} \times 10$ scenarios $\times$ 2 algorithms $= 60$ full rolling-horizon runs.

Each ratio-seed combination forms a paired block with one RH-ALNS and one D-ALNS result, yielding 30 complete blocks. Pairing controls for scenario-specific difficulty in the algorithm comparison.

The reported performance measures include the total objective value, transportation cost, fixed dispatch cost, weighted tardiness cost, postponement cost, and CPU time. The total objective measures overall solution quality, while the cost components indicate whether the increase in system cost is mainly caused by longer travel distance, greater tardiness, or order postponement.

To measure the paired performance of D-ALNS relative to RH-ALNS, the improvement rate for scenario $s$and dynamic order ratio $\rho$is defined as

$$(\text{Improvement})_{s,\rho} = \frac{F_{\text{RH-ALNS},s,\rho} - F_{\text{D-ALNS},s,\rho}}{F_{\text{D-ALNS},s,\rho}} \times 100\%,$$

where $F_{\text{RH-ALNS},s,\rho}$and $F_{\text{D-ALNS},s,\rho}$denote the total objective values obtained by RH-ALNS and D-ALNS under the same scenario and dynamic order ratio.

A positive improvement value indicates that D-ALNS obtains a lower objective value than RH-ALNS. Negative values indicate that RH-ALNS performs better, while a value of zero means that the two algorithms produce the same objective value. All positive, negative, and zero values are retained without truncation or selective removal.

All 60 formal runs complete all 60 customer orders. No violation is detected in future-information isolation, order completion, duplicate service, vehicle state, committed-route immutability, terminal feasibility, or objective reconciliation. Therefore, differences among the three dynamic ratios can be attributed to scenario dynamism and algorithm behavior rather than to infeasible solutions or data inconsistency.

Figure 9 presents the distribution of the paired improvement rates of D-ALNS over RH-ALNS under different dynamic-order ratios. For each ratio, ten paired scenarios generated using seeds 30–39 are considered. The horizontal axis represents dynamic-order ratios of 20%, 40%, and 60%, while the vertical axis reports the objective-value improvement of D-ALNS relative to RH-ALNS. Positive values indicate that D-ALNS obtains a lower total objective value and therefore outperforms RH-ALNS. The boxes show the interquartile distributions across the paired scenarios, the horizontal lines inside the boxes denote the medians, and the individual points represent scenario-level results. The values above the boxes indicate the corresponding mean improvement rates.

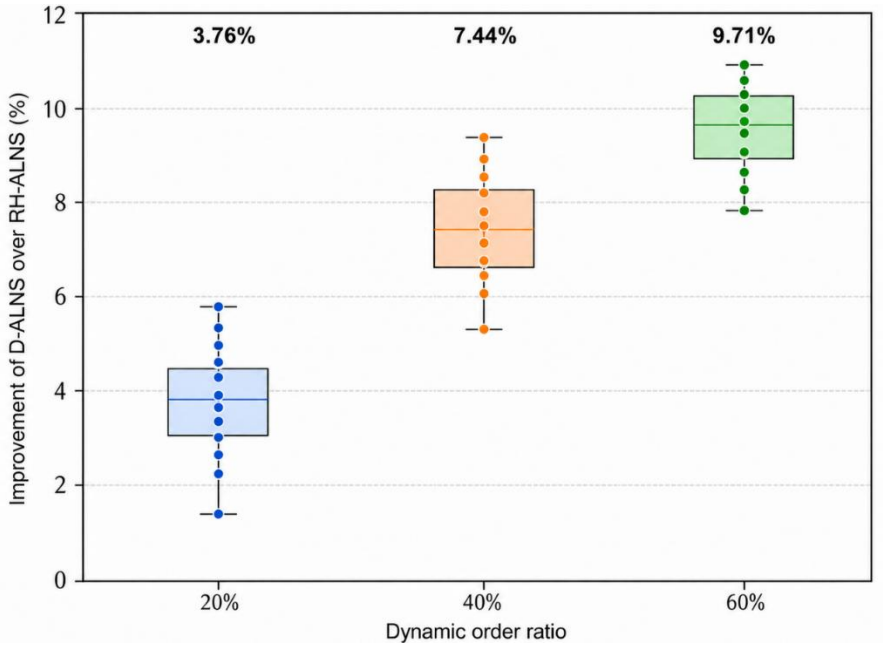

Fig. 9. Improvement distribution of D-ALNS over RH-ALNS under different dynamic order ratios

The results indicate that the advantage of D-ALNS increases as the dynamic-order ratio rises. At a dynamic-order ratio of 20%, the mean improvement is 3.76%, indicating a relatively limited difference between the two algorithms under low-dynamic conditions. When the ratio increases to 40%, the mean improvement rises to 7.44%. At the highest dynamic-order ratio of 60%, the mean improvement further increases to 9.71%, and the overall distribution shifts upward. This pattern suggests that, as more orders are released dynamically, time urgency, historical postponement, and the need for route reconstruction become increasingly important. Consequently, the time-pressure removal, time-priority insertion, and postponed-priority insertion operators in D-ALNS provide greater benefits. These results confirm that the contribution of the dynamic-order-oriented operators becomes more pronounced as the level of order dynamism increases, demonstrating the stronger adaptability of D-ALNS in highly dynamic urban distribution environments.

6.Conclusion, Limitations, and Future Research

The problem considered here involves orders that become known during route execution, while vehicles may still be serving routes fixed at earlier decision epochs. The model therefore combines rolling decisions with vehicle capacity, soft latest-service times, vehicle availability, committed routes, and order postponement. D-ALNS is built on the same rolling-horizon structure as RH-ALNS, but adds time-pressure removal, time-priority insertion, and postponed-priority insertion. The two algorithms otherwise use the same route evaluator, local search, adaptive weighting, simulated-annealing mechanism, and common parameter settings.

The comparison across the three problem sizes does not produce the same ranking at every scale. For n=40, TS records the lowest mean objective value and D-ALNS has no clear advantage. The pattern changes at n=60, where D-ALNS reaches 1905.79, compared with 2047.52 for RH-ALNS, giving a gap of 7.44%. At n=80, D-ALNS again has the lowest mean objective value, 2326.43, while RH-ALNS follows at 2351.53. Across all 30 scale–scenario blocks, the mean ranks of D-ALNS, TS, and RH-ALNS are 2.05, 2.58, and 2.98, respectively. The benefit of D-ALNS is therefore more visible on the medium- and large-scale cases, although it does not dominate all algorithms at the smallest scale. Similar dynamic-routing studies have also relied on repeated re-optimization and adaptive neighborhoods to deal with changes in customer and vehicle states [10,18].

The convergence experiment gives a slightly different picture. D-ALNS and RH-ALNS finish with almost the same mean objective values on the selected 60-customer epoch-0 instances, at 778.89 and 778.96. Their early search behavior is less similar. D-ALNS reaches 90% of its total improvement after 9.5% of normalized search progress and 5.27 s of CPU time, whereas RH-ALNS requires 26.0% and 17.70 s. The normalized AUC values are 4.01 and 4.16. In these cases, the main difference lies in how quickly useful route changes are found rather than in the final solution reached after the full search.

The ablation experiment also favors the complete operator set. Full D-ALNS obtains a mean objective value of 1905.79. Removing TPR, TPI, and PPI raises this value to 1962.14, 1997.70, and 2013.52, respectively, while RH-ALNS reaches 2047.52. The largest descriptive change appears after removing PPI. That result should not be overstated, since no order is actually postponed in the formal scenarios. The available vehicles and planning horizon are sufficient to serve all active orders. The experiment therefore supports the complete operator combination, but it does not provide a direct test of the value of postponement history itself.

A clearer change appears when the dynamic-order ratio is increased. The reported difference between D-ALNS and RH-ALNS rises from 3.76% at 20% dynamic orders to 7.44% at 40% and 9.71% at 60%. When fewer orders are visible at the beginning of the horizon, later routing decisions have to absorb more newly released demand. Under these conditions, the additional operators in D-ALNS have a larger effect. The gain is therefore related to the degree of order dynamism rather than being constant across all instances.

The current experiments still simplify several parts of actual urban distribution. The instances are derived from Solomon RC101 rather than operational delivery records, and the model assumes one depot, homogeneous vehicles, deterministic travel times, and constant travel speed. Multi-depot systems and heterogeneous fleets would introduce additional assignment and routing decisions; Wang *et al.*, [10] for example, study these decisions jointly in a dynamic vehicle-routing setting. Travel conditions may also change after a route has been dispatched. Contextual stochastic routing offers one possible way to incorporate observed conditions and historical travel-time information into route evaluation [35]. Another limitation is the absence of actual postponement in the formal runs. More restrictive fleet capacity, shorter service windows, or concentrated order releases would be needed to examine that mechanism directly.

Computational comparison is also not based on an identical low-level search budget. GA, TS, and ILS stop according to candidate evaluations, whereas RH-ALNS and D-ALNS use ALNS iterations. CPU time is reported separately, but the two stopping units are not equivalent. The convergence analysis is narrower still, covering four 60-customer epoch-0 cases. Later epochs may behave differently because some vehicles are already committed and the remaining planning time is shorter.

Further work can therefore focus on more restrictive and less deterministic operating conditions. Historical order and vehicle records would allow release times, service durations, priority levels, and travel times to be estimated from observed data. The model could also include multiple depots, heterogeneous or electric vehicles, charging constraints, and working-time limits. Routing may eventually need to be linked with request acceptance or other platform decisions; Wang and Xu [16] consider such interactions in dynamic passenger–parcel transportation. Search control could also vary with the current rolling state instead of keeping the same destroy fraction, reaction factor, cooling schedule, and search intensity at every epoch. Learning-based methods offer one option for this type of control. Kadyrov *et al.*, [36] for example, use deep reinforcement learning for dynamic routing under demand and traffic uncertainty. In the present framework, learning would be more naturally used to guide operator or parameter selection while the route evaluator continues to enforce feasibility.

D-ALNS does not produce the best result under every tested condition. Its main advantage appears in the 60- and 80-customer cases, in the early stage of the search, and when a larger share of orders is released dynamically. These results provide support for using dynamic order information in neighborhood design, but more demanding instances and operational data are still needed to determine how stable that advantage is in practice.

Acknowledgement
The author gratefully acknowledges the support from the Youth Project of Anhui Provincial Philosophy and Social Science Planning Program under Grant AHSKQ2022D070.

Conflicts of Interest
The authors declare no conflicts of interest.

References
[1] Zhang, W., Gao, B., Chen, P., Chen, Y., & Xiao, J. (2026). Dynamic request vehicle routing problem with workload balancing under time-dependent travel times. *Transportation Research Part E: Logistics and Transportation Review*, 210, 104832. https://doi.org/10.1016/j.tre.2026.104832
[2] Çelik, S., Schrotenboer, A. H., van der Heide-Martin, L., & van Woensel, T. (2026). Is waiting worth it? The value of delaying time window assignment in vehicle routing problems. *Transportation Research Part B: Methodological*, 204, 103381. https://doi.org/10.1016/j.trb.2025.103381
[3] Liu, Y., Yu, Y., Baldacci, R., Tang, J., & Sun, W. (2025). Optimizing carbon emissions in green logistics for time-dependent routing. *Transportation Research Part B: Methodological*, 192, 103155. https://doi.org/10.1016/j.trb.2025.103155
[4] Nourmohammadi, Z., Hu, B., Rey, D., & Saberi, M. (2025). A data-driven preference learning approach for multi-objective vehicle routing problems in last-mile delivery. *Transportation Research Part C: Emerging Technologies*, 174, 105101. https://doi.org/10.1016/j.trc.2025.105101
[5] Konovalenko, A., Hvattum, L. M., & Msakni, M. K. (2025). Using machine learning to identify hidden constraints in vehicle routing problems. *Computers & Operations Research*, 179, 107029. https://doi.org/10.1016/j.cor.2025.107029
[6] Voigt, S. (2025). A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. *European Journal of Operational Research*, 322(2), 357–375. https://doi.org/10.1016/j.ejor.2024.05.033
[7] Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1), 1–11. https://doi.org/10.1016/j.ejor.2012.08.015
[8] Psaraftis, H. N., Wen, M., & Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1), 3–31. https://doi.org/10.1002/net.21628
[9] Ojeda Rios, B. H., Xavier, E. C., Miyazawa, F. K., Amorim, P., Curcio, E., & Santos, M. J. (2021). Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering*, 160, 107604. https://doi.org/10.1016/j.cie.2021.107604
[10] Wang, S., Sun, W., & Huang, M. (2024). An adaptive large neighborhood search for the multi-depot dynamic vehicle routing problem with time windows. *Computers & Industrial Engineering*, 191, 110122. https://doi.org/10.1016/j.cie.2024.110122
[11] Pina-Pardo, J. C., Silva, D. F., Smith, A. E., & Gatica, R. A. (2024). Dynamic vehicle routing problem with drone resupply for same-day delivery. *Transportation Research Part C: Emerging Technologies*, 162, 104611. https://doi.org/10.1016/j.trc.2024.104611
[12] Lan, L., van Doorn, J. M. H., Wouda, N. A., Rijal, A., & Bhulai, S. (2024). An iterative sample scenario approach for the dynamic dispatch waves problem. *Transportation Science*, 58(4), 726–740. https://doi.org/10.1287/trsc.2023.0111
[13] Baty, L., Jungel, K., Klein, P. S., Parmentier, A., & Schiffer, M. (2024). Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. *Transportation Science*, 58(4), 708–725. https://doi.org/10.1287/trsc.2023.0107
[14] Neria, G., & Tzur, M. (2024). The dynamic pickup and allocation with fairness problem. *Transportation Science*, 58(4), 821–840. https://doi.org/10.1287/trsc.2023.0228
[15] Khorasanian, D., Patrick, J., & Sauré, A. (2024). Dynamic home care routing and scheduling with uncertain number of visits per referral. *Transportation Science*, 58(4), 841–859. https://doi.org/10.1287/trsc.2023.0120

[16] Wang, Y., & Xu, M. (2025). Dynamic confirmation, compensation and routing for combined urban transportation of passengers and parcels. *Transportation Science*, 59(5), 932–950. https://doi.org/10.1287/trsc.2024.0827
[17] Horváth, M., & Tamási, T. (2025). A general modeling and simulation framework for dynamic vehicle routing. *EURO Journal on Transportation and Logistics*, 14, 100159. https://doi.org/10.1016/j.ejtl.2025.100159
[18] Sze, J. F., Salhi, S., & Wassan, N. (2024). An adaptive variable neighbourhood search approach for the dynamic vehicle routing problem. *Computers & Operations Research*, 164, 106531. https://doi.org/10.1016/j.cor.2024.106531
[19] Lin, S., Hu, J., Ma, W., Zheng, C., & Li, R. (2024). Integrated real-time signal control and routing optimization: A two-stage rolling horizon framework with decentralized solution. *Transportation Research Part C: Emerging Technologies*, 165, 104734. https://doi.org/10.1016/j.trc.2024.104734
[20] Zhang, Z., Zhang, Y., & Baldacci, R. (2024). Generalized riskiness index in vehicle routing under uncertain travel times: Formulations, properties, and exact solution framework. *Transportation Science*, 58(4), 761–780. https://doi.org/10.1287/trsc.2023.0345
[21] Metz, L., Mutzel, P., Niemann, T., Schürmann, L., Stiller, S., & Tillmann, A. M. (2024). Delay-resistant robust vehicle routing with heterogeneous time windows. *Computers & Operations Research*, 164, 106553. https://doi.org/10.1016/j.cor.2024.106553
[22] Li, N., & Wang, Z. (2025). Vehicle routing problem for omnichannel retailing including multiple types of time windows and products. *Computers & Operations Research*, 173, 106828. https://doi.org/10.1016/j.cor.2024.106828
[23] Belhaiza, S., & Laporte, G. (2026). A data-driven heuristic for the dynamic vehicle routing problem with multiple soft time windows. *Computers & Operations Research*, 187, 107341. https://doi.org/10.1016/j.cor.2025.107341
[24] Accorsi, L., & Vigo, D. (2024). Routing one million customers in a handful of minutes. *Computers & Operations Research*, 164, 106562. https://doi.org/10.1016/j.cor.2024.106562
[25] Kerscher, C., & Minner, S. (2025). Decompose-route-improve framework for solving large-scale vehicle routing problems with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 204, 104409. https://doi.org/10.1016/j.tre.2025.104409
[26] Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4), 455–472. https://doi.org/10.1287/trsc.1050.0135
[27] Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8), 2403–2435. https://doi.org/10.1016/j.cor.2005.09.012
[28] Johnn, S.-N., Darvariu, V.-A., Handl, J., & Kalcsics, J. (2024). A graph reinforcement learning framework for neural adaptive large neighbourhood search. *Computers & Operations Research*, 172, 106791. https://doi.org/10.1016/j.cor.2024.106791
[29] Sun, H., Wu, T., Bai, Q., & Zhou, Z. (2025). Improved adaptive large-scale neighborhood search algorithm for electric vehicle routing problem with soft time windows and linear weight-related discharging. *Expert Systems with Applications*, 280, 127344. https://doi.org/10.1016/j.eswa.2025.127344
[30] Su, Y., Zhang, S., Wang, Y., Cui, X., & Wu, Y. (2025). An adaptive large neighborhood search with multi-deletion operators for multi-depot green vehicle routing problem with time windows. *Swarm and Evolutionary Computation*, 95, 101942. https://doi.org/10.1016/j.swevo.2025.101942
[31] Xu, K., Cao, Z., Zheng, C., & Liu, L. (2026). Learning to search for vehicle routing with multiple time windows. *Computers & Industrial Engineering*, 213, 111760. https://doi.org/10.1016/j.cie.2025.111760
[32] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265. https://doi.org/10.1287/opre.35.2.254
[33] Keskin, M., Branke, J., Deineko, V., & Strauss, A. K. (2023). Dynamic multi-period vehicle routing with touting. *European Journal of Operational Research*, 310(1), 168–184. https://doi.org/10.1016/j.ejor.2023.02.037
[34] Zhang, J., Luo, K., Florio, A. M., & van Woensel, T. (2023). Solving large-scale dynamic vehicle routing problems with stochastic requests. *European Journal of Operational Research*, 306(2), 596–614. https://doi.org/10.1016/j.ejor.2022.07.015
[35] Serrano, B., Florio, A. M., Minner, S., Schiffer, M., & Vidal, T. (2026). Contextual stochastic vehicle routing with time windows. *INFORMS Journal on Computing*. Advance online publication. https://doi.org/10.1287/ijoc.2025.1189
[36] Kadyrov, S., Azamov, A., Abdumajitov, Y., & Turan, C. (2025). Deep reinforcement learning for dynamic vehicle routing with demand and traffic uncertainty. *Operations Research Perspectives*, 15, 100351. https://doi.org/10.1016/j.orp.2025.100351