DVRPTW-BENCH: A BENCHMARK FOR DYNAMIC VEHICLE ROUTING PROBLEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic vehicle routing with time windows (DVRPTW) requires planning routes as requests are revealed during operation. Solvers need to coordinate planning timing, dispatch, and routing over diverse, evolving request snapshots while maintaining computational efficiency. Existing benchmarks typically either use a small number of solvers to study problem difficulty or compare solver performance under fixed demand settings, therefore overlooking differences in the advantage of various methods across different scenarios. DVRPTW-BENCH addresses this gap by systematically evaluating 17 approaches across 135 demand configurations with 10 instances each. The evaluated approaches cover three planning components: decision timing, dispatch, and routing. And demand is modeled through four factors: problem scale, degree of dynamism (DoD), temporal arrival patterns, and spatial demand evolution. Experiments show higher DoD increases learned dispatch's advantage in distance per served customer (DPC) relative to rolling hybrid genetic search (HGS). Temporal concentration reduces DPC more for HGS than for neural constructors, while dispersed arrivals give learned dispatch a larger advantage. Strong bursts can also reverse the benefits of urgency triggers for some neural constructors on large instances. Spatial hotspot shifts yield larger DPC reductions for HGS than for neural constructors on large instances. The benchmark suite and code are available at https://anonymous.4open.science/r/iclr-dvrptw0925.

1 INTRODUCTION

Dynamic vehicle routing problem with time windows (DVRPTW) arises in applications such as food and parcel delivery, where requests arrive during operation. A fleet of capacity-limited vehicles must serve these requests within specified time windows while minimizing routing cost. Every planning decision relies on currently revealed requests.

Compared with static VRPTW, DVRPTW expands the decision space and exposes solvers to evolving routing subproblems during execution. A method must decide *when to plan*, *which requests to dispatch or defer*, and *how to route* the selected requests. These decisions are coupled: waiting can improve route consolidation while reducing the time left to meet deadlines thus affecting subsequent planning states. Effective methods therefore need to coordinate timing, dispatch, and routing throughout the dynamic process, creating challenges for solver design and evaluation.

The challenges faced by a dynamic solver also depend on how demand unfolds. Problem scale determines the total request count, while the degree of dynamism (DoD) specifies the proportion of requests revealed after operation begins. Concentrated arrivals can create less but larger planning pools, while dispersed arrivals introduce requests more gradually. Stationary hotspots maintain a relatively stable geographic pattern, whereas shifting hotspots change the spatial composition of subproblems. These variations influence the value of waiting and request selection, the opportunities for route consolidation, and the computation required at each planning step. A strategy's effectiveness therefore depends on the demand conditions.

Existing benchmarks typically study selected aspects of problem difficulty with a small set of solvers (Kilby et al., 1998; Van Lon & Holvoet, 2015; Van Lon et al., 2016), or compare solver performance under prescribed demand settings (Kool et al., 2023). Systematic comparisons of how different planning methods respond to changes in demand remain limited (Table 1). Overall performance scores can conceal which demand conditions favor a method and when its benefits diminish.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Demand-factor control and method evaluation in existing benchmarks. △ limited or coupled coverage; Numbers count methods or trigger families.

Feature	**DVRPTW-Bench**	van Veen et al. (2013)	van Lon et al. (2016)	EURO–NeurIPS (2023)	Zhang et al. (2023)
Demand design: factor control					
Problem size	✓	✗	✗	△	△
Degree of dynamism (DoD)	✓	✓	✗	✗	△
Temporal arrival profiles	✓	✗	✓	✗	✗
Time-varying hotspots	✓	✗	✗	✗	✓
Method evaluation					
Classical routing methods	✓ (7)	✓	✓	✓	✓
Neural route constructors	✓ (4)	✗	✗	✗	✗
Dispatch strategies	✓ (2)	✗	✗	✓	✗
Planning-trigger comparisons	✓ (4)	✗	✗	✗	✗

DoD denotes the dynamic-request share. In Zhang et al., size and DoD vary jointly.

To address this gap, DVRPTW-BENCH combines controlled demand generation with systematic method evaluation, including decision timing, dispatch and routing. The main contributions are:

- **A benchmark connecting demand conditions and method performance.** DVRPTW-BENCH provides a common benchmark for comparing dynamic routing methods across controlled demand scenarios. The evaluation reveals how demand conditions change the relative benefits of routing, dispatch, and timing strategies.
- **Fine-grained modeling of dynamic demand.** The generator separately controls problem scale, degree of dynamism, temporal arrival patterns, and spatial demand evolution. It provides 1,350 instances across 135 demand configurations.
- **Systematic comparison of planning strategies.** The evaluation covers search-based and neural routing methods, learned and anticipatory dispatch, and alternative planning triggers under shared execution rules. Targeted comparisons assess routing performance and the additional value of dispatch and timing across demand settings. Runtime measurements and budget studies further characterize quality-computation trade-offs.

2 RELATED WORK

2.1 BENCHMARKS FOR DYNAMIC VEHICLE ROUTING

Existing benchmarks and evaluation studies capture different aspects of dynamic demand. Kilby et al. (1998) construct dynamic instances by progressively revealing requests from static routing datasets. van Veen et al. (2013) extend Solomon instances to dynamic settings with varying degrees of dynamism. Van Lon et al. (2016) examine arrival regularity and request urgency, while Zhang et al. (2023) evaluate routing and scheduling policies on large-scale street-network instances with stationary and time-varying spatial demand.

Other resources emphasize policy comparison and reusable evaluation environments. The EURO–NeurIPS competition compares policies using evaluation data derived from real-world routing problems (Kool et al., 2023). SVRPBench models stochastic traffic conditions (Heakl et al., 2025), dvrpsim provides configurable dynamic-routing simulation (Horváth & Tamási, 2025), and MAEnvs4VRP supports multi-agent routing environments (Gama et al., 2026).

Systematic comparisons of how demand conditions change the relative benefits of routing, dispatch, and planning-trigger strategies remain limited. DVRPTW-BENCH combines controlled variation in problem scale, DoD, temporal arrival profiles, and spatial evolution with comparisons of these

strategies under shared execution rules. Table 1 summarizes demand-factor control and method evaluation in representative benchmarks and evaluation studies.

2.2 Solution Approaches for Dynamic Routing

Planning triggers. Ninikas & Minis (2014) examine reoptimization after each new request or after a preset number of requests have arrived. Ulmer et al. (2017) compare fixed-interval replanning with event-driven triggers based on new request arrivals and vehicle-status changes.

Dispatch strategies. Dispatch strategies determine which pending requests to serve now and which to postpone. Waiting, buffering, and dispatch-wave methods preserve opportunities to consolidate requests (Mitrović-Minić & Laporte, 2004; Klapp et al., 2018; Voccia et al., 2019). Anticipatory policies make dispatch decisions using sampled future demand (Bent & Van Hentenryck, 2004; Hvattum et al., 2006; Lan et al., 2024). Baty et al. (2024) learn request-specific prizes and use a prize-collecting routing solver to jointly select requests and construct routes.

Route planning. Routing solvers construct feasible routes for a list of requests. Search-based methods, such as HGS and ALNS, iteratively improve candidate solutions (Wouda et al., 2024; Ropke & Pisinger, 2006). Neural constructive methods, including Attention Model, POMO, SymNCO, and PolyNet, generate routes using trained policies (Kool et al., 2019; Kwon et al., 2020; Kim et al., 2022; Hottung et al., 2025). Rolling reoptimization uses these solvers to solve the static snapshots at every planning timing.

3 Benchmark Design

This section defines the problem and the framework of DVRPTW-Bench.

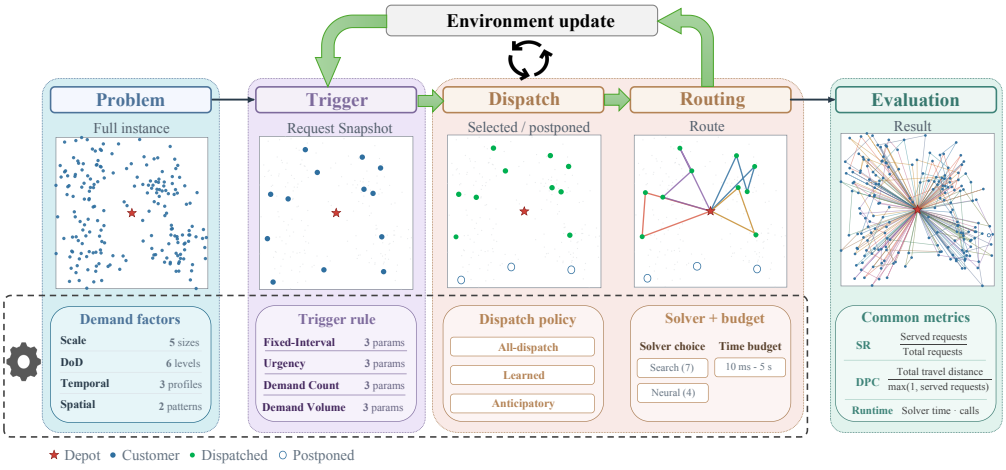


Figure 1: The framework of DVRPTW-Bench

3.1 Problem Definition

DVRPTW-Bench focuses on depot-based delivery. Once a vehicle departs, its route remains fixed until it returns to the depot. Newly revealed requests can only be assigned to trips that have not yet departed. Let $G = (\mathcal{V}, \mathcal{E})$ be a complete directed graph with depot $0 \in \mathcal{V}$ and customer set $N = \mathcal{V} \setminus \{0\}$, with $n = |N|$. Customer $i \in N$ has location x_i, demand q_i, service time s_i, release time r_i, and service window $[e_i, l_i]$. A homogeneous fleet of vehicles with capacity Q operates over horizon $[0, H]$. Travel from i to j has distance d_{ij} and travel time t_{ij}. Here, Euclidean distance is adopted where $d_{ij} = \|x_i - x_j\|_2$ and $t_{ij} = \lceil d_{ij}/v \rceil$.

Let π be an online routing policy, $\mathcal{R}^{\text{exec}}(\pi)$ the set of executed depot-to-depot trips, and $y_i(\pi) \in \{0, 1\}$ indicate whether customer i is served. For a trip $R = (0, i_1, \ldots, i_{m_R}, 0)$ departured at δ_R, the

DVRPTW objective and its core feasibility constraints are formulated as

$$\begin{aligned}
\min_{\pi\in\Pi} \quad & \left(\sum_{i\in N}\big(1-y_i(\pi)\big),\ \sum_{R\in\mathcal{R}^{\mathrm{exec}}(\pi)}\sum_{k=0}^{m_R} d_{i_k i_{k+1}}\right)_{\mathrm{lex}} \\
\text{s.t.} \quad & u_{i_k} = \max\left\{e_{i_k}, u_{i_{k-1}} + s_{i_{k-1}} + t_{i_{k-1}i_k}\right\}, \quad u_{i_0} = \delta_R, \\
& u_{i_k} \le l_{i_k}, \qquad k = 1,\dots,m_R, \\
& \sum_{k=1}^{m_R} q_{i_k} \le Q, \\
& r_{i_k} \le \delta_R, \qquad k = 1,\dots,m_R, \qquad \forall R \in \mathcal{R}^{\mathrm{exec}}(\pi).
\end{aligned} \tag{1}$$

Here $i_0 = i_{m_R+1} = 0$ and $s_0 = 0$. $(\cdot,\cdot)_{\mathrm{lex}}$ denotes lexicographic minimization: first minimizes unserved customers and then executed travel distance. Service starts after both the time window opens and the vehicle arrives. They also must begin by the window end. Trip demand cannot exceed vehicle capacity. Only requests already released can be included.

3.2 Instance Generation

The generator controls scale, degree of dynamism (DoD), temporal demand profiles and spatial evolution. The process is as follows and distribution parameters are provided in Appendix B.

Scale and DoD. Problem scale n is the total number of customers, and DoD is the fraction of dynamic requests. The dynamic request count is

$$M = \lfloor n \cdot \mathrm{DoD} \rfloor. \tag{2}$$

Temporal profiles. Temporal profiles describe the distributions of release times and time windows. Latent demand times $\tilde{r}_i \sim f_T$ are sampled over horizon H, separately for dynamic and static requests. DVRPTW-BENCH provides three predefined profiles: *Uniform*, *Strong-burst* with four narrow peaks, and *Realistic double-peak* with lunch and dinner peaks:

$$\begin{aligned}
f_U(t) &= \frac{1}{H}, \\
f_B(t) &= (1-\rho) f_U(t) + \frac{\rho}{K_B}\sum_{k=1}^{K_B} \phi(t;\mu_k,\sigma_B^2), \\
f_R(t) &= \omega_0 U_O(t) + \omega_L\,\mathrm{TN}_{W_L}(t;\mu_L,\sigma_L^2) + \omega_D\,\mathrm{TN}_{W_D}(t;\mu_D,\sigma_D^2).
\end{aligned} \tag{3}$$

Gaussian mixtures model the temporal peaks. Here $K_B = 4$, ϕ is a Gaussian density, and TN_W is a Gaussian density truncated to window W. U_O is uniform outside the lunch and dinner windows, with $\omega_0 = 1 - \omega_L - \omega_D$. Time windows are anchored to $\tilde{r}_i$ using sampled reaction times Δ_i and window widths w_i:

$$e_i = \tilde{r}_i + \Delta_i, \qquad l_i = e_i + w_i. \tag{4}$$

Dynamic requests use $r_i = \tilde{r}_i$, while static requests use $r_i = 0$.

Spatial evolution. Spatial evolution describes how the spatial distribution of requests changes over time. The horizon is divided into four equal periods. Dynamic temporal records in period b are assigned to clusters using period-specific probabilities:

$$q_{b,c} = \begin{cases} \dfrac{n_c}{n}, & \text{stationary}, \\ (1-\alpha)\dfrac{n_c}{n} + \alpha\dfrac{\mathbf{1}\{c \in A_b\}}{|A_b|}, & \text{time-varying}, \end{cases} \tag{5}$$

where n_c is the size of cluster c, A_b is the set of active hotspot clusters in period b, and α controls hotspot strength. Static records are randomly paired with the remaining nodes. Spatial comparisons use $n \ge 200$, ensuring at least four clusters for the four-period hotspot rotation.

Scenarios are labeled US, UV, BS, BV, RS, or RV. The first letter denotes the temporal profile (U: Uniform; B: Strong-burst; R: Realistic double-peak), and the second denotes the spatial pattern (S: stationary; V: time-varying).

3.3 Environment and Interaction Protocol

The environment advances the simulation minute by minute. A *planning trigger* uses the current time and revealed requests to determine whether to invoke the solver for route planning.

When the solver is invoked at time t, the environment provides $\mathcal{U}_t$, the set of all released, unassigned requests whose time windows have not expired. $|\mathcal{U}_t|$ is termed the *planning-pool size*. Previous route plans and vehicle positions are not included in the method's observation. Later, the solver returns J_t planned routes and their departure times,

$$\mathcal{P}_t = \{(R_j, \delta_j)\}_{j=1}^{J_t}, \qquad J_t \geq 0, \qquad R_j = (0, i_1, \ldots, i_{m_j}, 0), \tag{6}$$

where R_j is a depot-to-depot customer sequence and $\delta_j \geq t$ is its planned departure time. The environment validates the returned routes, assigns requests to accepted routes. Requests left unassigned after validation remain pending until assigned or their time windows expire.

3.4 Method Families and Tunable Factors

The benchmark compares three method families: planning trigger, dispatch and routing methods. Implementation details and experimental configurations are provided in Appendix C.

Planning trigger. DVRPTW-BENCH sets four trigger families. *Fixed-interval* triggers planning at regular time intervals. *Demand-count* and *demand-volume* trigger when the request count or total demand in $\mathcal{U}_t$ reaches a preset percentage of the corresponding total for all dynamic requests. These proportional thresholds adapt to different problem scales. *Urgency* triggers when any request in $\mathcal{U}_t$ has slack at or below a preset threshold, where

$$\text{slack}_i(t) = l_i - t_{0i} - t. \tag{7}$$

Here, t_{0i} is the travel time from the depot to customer i. Slack measures the remaining time before the latest depot departure that allows arrival by l_i.

Route constructor. Two solver classes are considered under this scheme.

Search-based solvers include seven methods: HGS through PyVRP (Wouda et al., 2024), OR-Tools (Perron & Furnon, 2019), ALNS (Ropke & Pisinger, 2006), Ant Colony Optimization (ACO) (Dorigo et al., 1996), Lin-Kernighan heuristic (LK) (Lin & Kernighan, 1973), Nearest Neighbor with 2-opt improvement (NN+2-opt) (Rosenkrantz et al., 1977; Croes, 1958), and Tabu Search (Glover, 1989).

Neural constructive solvers include Attention model, POMO, SymNCO, and PolyNet (Kool et al., 2019; Kwon et al., 2020; Kim et al., 2022; Hottung et al., 2025). They construct routes directly for the current snapshot using learned policies rather than search. Dynamic routing requires these policies to generalize across snapshots with varying sizes and demand patterns. So two training settings are considered. Scale-matched models are trained separately for each nominal problem size, while mixed-scale models are trained jointly on multiple sizes $n \in \{20, 50, 100, 200\}$. Unless otherwise specified, the main comparison uses mixed-scale checkpoints. Generalization is discussed further in Appendix D.3.

Dispatch methods. ML-CO (Baty et al., 2024) assigns learned dispatch values to requests and uses prize-collecting HGS to select requests that dispatch now and construct route. ML-CO is evaluated with the open-released checkpoint and four retrained variants: SL (small scales and low DoD), LH (large scales and high DoD), GEN (all scales and multiple DoD levels), and GEN-BAL2 (GEN with balanced observation counts across scales). The main comparison uses SL, which has the best overall performance. ICD (Lan et al., 2024) uses an iterative sample-scenario policy to evaluate dispatch choices together with sampled future demand, allowing requests to be postponed when future arrivals are expected to create better routing opportunities. ICD uses the instance's demand-generation parameters to sample future requests.

3.5 Evaluation Metrics

The primary evaluation metrics are service rate (SR) and distance per served customer (DPC):

$$\text{SR} = \frac{n_{\text{served}}}{n}, \qquad \text{DPC} = \frac{\text{Total Distance}}{\max(1, n_{\text{served}})}. \tag{8}$$

Table 2: Benchmark factors and evaluated configurations.

Axis	Evaluated settings / methods
Scale	20, 50, 100, 200, 500
DoD	0, 0.2, 0.5, 0.8, 0.9, 0.95
Temporal profile	Uniform(U); Strong burst(B); Realistic double peak(R)
Spatial	Stationary(S); Time-varying hotspots(V) in applicable dynamic cells
Decision timing	fixed interval: 15/30/60 min; urgency: 5/10/20 min slack; demand count: 0.5/1/2%; demand volume: 0.5/1/2%
Search solvers	HGS, OR-Tools, ALNS; ACO, LK, NN+2-opt, Tabu Search
Neural constructors	Attention Model, POMO, SymNCO, PolyNet
Learned dispatch	ML-CO: official model; SL (small/low-DoD); LH (large/high-DoD); GEN (general); GEN-BAL2 (scale-balanced)
Anticipatory dispatch	ICD: sampling from instance-specific demand-generation distributions

Release-aware HGS (RA-HGS) knows all requests information including release time at the begining and constructs feasible routes, each departing only after all its requests have been released. The relative gap to this reference helps distinguish increases in baseline routing cost from deterioration in an online method's performance. For an online policy π, the online–reference gap is

$$\mathrm{OG}(\pi) = \frac{\mathrm{DPC}(\pi)}{\mathrm{DPC}^{\mathrm{RA}}} - 1. \tag{9}$$

4 Experimental Evaluation

This section introduces the experimental setup and evaluates dynamic routing methods in terms of routing quality, robustness, decision timing, and computational efficiency.

4.1 Experimental Setup

The evaluation comprises 1,350 instances from 135 demand configurations with ten replicas each. Table 2 summarizes the benchmark factors and evaluated methods. Experiments were conducted on a server with two Intel Xeon Gold 6459C CPUs and eight NVIDIA GeForce RTX 5090 GPUs, using PyTorch 2.10.0 with CUDA 12.8.

4.2 Comparison of Routing and Dispatch Methods

Under Fix-30, the evaluation first compares seven search-based routing solvers to establish a routing reference. It then compares rolling HGS with neural route constructors (Attention, PolyNet, SymNCO, and POMO) and dispatch methods (ML-CO and ICD).

4.2.1 Traditional Search-Based Routing Solvers

Table 3: Routing solvers under Fix-30

Solver	SR (%)	DPC	Time (s)
HGS	99.26	**313.8**	23.0
OR-Tools	99.26	320.5	23.9
ALNS	99.26	325.2	24.9
LK	99.26	346.6	22.9
Tabu Search	99.26	348.9	22.9
ACO	99.26	349.1	24.2
NN+2-opt	99.26	384.5	< 0.1

Across the 1,350-instance main grid, HGS achieves the lowest mean DPC at the same mean service rate under the one-second per-call budget (Table 3). OR-Tools and ALNS are the next-best methods, with mean per-instance DPC gaps of +3.72% and +5.77% relative to HGS, respectively. The remaining search heuristics have gaps ranging from +15.57% to +27.70%. HGS is therefore used as the comparison baseline and the routing backbone for dispatch methods.

4.2.2 NEURAL ROUTE CONSTRUCTION AND DISPATCH

Table 4 compares neural route constructors and dispatch methods against rolling HGS on the full 1,350-instance grid. Full configurations appear in Table 7.

Neural constructors yield higher DPC than rolling HGS at similar service rates. PolyNet and SymNCO have comparable mean DPC gaps (15.10% and 15.32%), followed by POMO (16.84%) and Attention (51.23%). These gaps are usually accompanied by fewer customers per trip.

In online execution, neural constructors process sequential subproblem snapshots $\mathcal{U}_t$ with scale volatility and spatial non-stationarity. PolyNet overcomes this constraint by employing K specialized policy heads independent of snapshot scale. SymNCO's geometric symmetry regularization mitigates spatial orientation bias. This regularization promotes reuse of geometric routing patterns, offering a plausible explanation for its lower pooled DPC.

Dispatch decisions further affect the performance. ML-CO SL reduces DPC by 6.93% with the same HGS routing backbone. ICD with instance configuration yields a smaller DPC reduction of 1.15%. Its DPC remains substantially higher than the offline release-aware reference, highlighting the gap between anticipation using demand-generation parameters and planning with realized future requests.

Table 4: Representative method families on the 1,350-instance grid. ΔDPC is the mean per-instance DPC gap vs rolling HGS (95% bootstrap CI).

Method	SR (%)	ΔDPC (%) [95% CI]	Cust./trip	Single(%)	Depot(%)	Calls	Time(s)
Rolling HGS	99.26	—	3.39	35.1	69.7	23.8	23.0
Attention (mixed)	98.99	+51.23 [48.72, 53.82]	1.90	59.8	84.5	23.3	0.7
PolyNet (mixed)	99.23	+15.10 [14.68, 15.54]	2.83	46.6	75.1	23.4	0.9
SymNCO (mixed)	99.23	+15.32 [14.88, 15.77]	2.84	43.5	74.4	23.3	0.6
POMO (mixed)	99.24	+16.84 [16.42, 17.27]	2.68	50.2	77.1	23.3	0.7
ML-CO SL	99.26	−6.93 [−7.61, −6.24]	3.59	21.1	67.4	38.8	39.8
ICD, instance config	99.26	−1.15 [−1.26, −1.05]	3.45	33.8	69.3	23.8	164.1
Release-aware HGS (offline)	100.00	−25.55 [−26.13, −24.97]	4.84	7.8	59.1	1.0	30.0

4.3 METHOD ROBUSTNESS AND DEMAND SENSITIVITY

This section examines sensitivity to scale, DoD, and spatiotemporal demand under Fix-30, using the configurations in Table 4. Training-distribution comparisons appear separately in Appendix D.3.

Problem scale and dynamism. For neural route constructors, the DPC gap to HGS widens with scale: PolyNet's gap increases from 6.43% at $n = 20$ to 21.22% at $n = 500$. Within this family, the leading method is changing among different scenarios. PolyNet has the lowest reported mean gap at $n \leq 200$ and DoD ≤ 0.5; SymNCO leads at $n = 500$ and DoD ≥ 0.8 (Figure 2a).

For learned dispatch, ML-CO improves on HGS in all five scale averages and provides larger gains at high DoD: its DPC reduction is 2.47% at DoD $= 0.2$ and 12.73% at DoD $= 0.95$. Selective postponement can preserve opportunities to combine current requests with later arrivals, providing a plausible explanation for this high-DoD advantage. For anticipatory dispatch, ICD provides smaller gains, reducing DPC relative to HGS by 0.34% and 2.11% at DoD $= 0.2$ and 0.95 (Table 9).

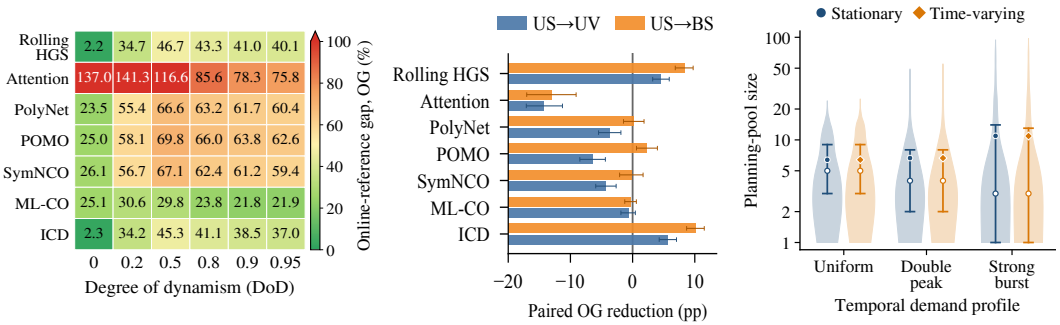
Temporal demand patterns. Using RA-HGS as the baseline, rolling-HGS OG changes from 42.73 to 41.69 and 34.40 across US–RS–BS, and from 47.89 to 47.62 and 35.73 across UV–RV–BV, with most gains occurring under strong bursts. In contrast, for neural constructor OG remains nearly

unchanged under S: 55.36–55.50–55.19 for PolyNet, 54.96–55.24–55.08 for SymNCO, and 58.09–57.70–55.79 for POMO. Under V, their U-to-B OG reductions are only 1.74, 0.46, and 6.55 points, respectively, compared with 12.16 points for rolling HGS. Their relative gaps therefore widen toward burst demand. Strong bursts increase planning-pool variability (Figure 2(c)), which may degrade the performance of neural constructor.

For learned dispatch, ML-CO's OG remains nearly constant across US–RS–BS (23.31–23.27–23.63) and decreases by only 2.83 points from UV to BV, so its advantage over rolling HGS narrows toward B. For anticipatory dispatch, ICD more closely tracks rolling HGS, decreasing from 41.88 to 40.53 and 31.75 under S, and from 45.01 to 44.96 and 33.69 under V (Table 8).

Spatial demand patterns. Using RA-HGS as the common reference, time-varying hotspots reduce rolling-HGS OG by 4.54 percentage points on paired dynamic instances with $n \geq 200$. For neural constructors, OG is increasied by 3.65–6.47 percentage points (Figure 2(b)). With similar planning-pool size distributions (Figure 2(c)), this contrast suggests that neural comstructors adapt less to changes in the geographic structure of concurrently available requests.

For learned dispatch, ML-CO's OG reduction under US→UV is not statistically significant because its 95% CI spans zero. For anticipatory dispatch, ICD's OG falls by 5.68 points, similar to rolling HGS's 4.54-point reduction.



(a) DoD Sensitivity (b) Demand-structure benefits (c) Planning-pool size distribution

Figure 2: Demand-regime effects

4.4 Decision Timing across Routing Methods

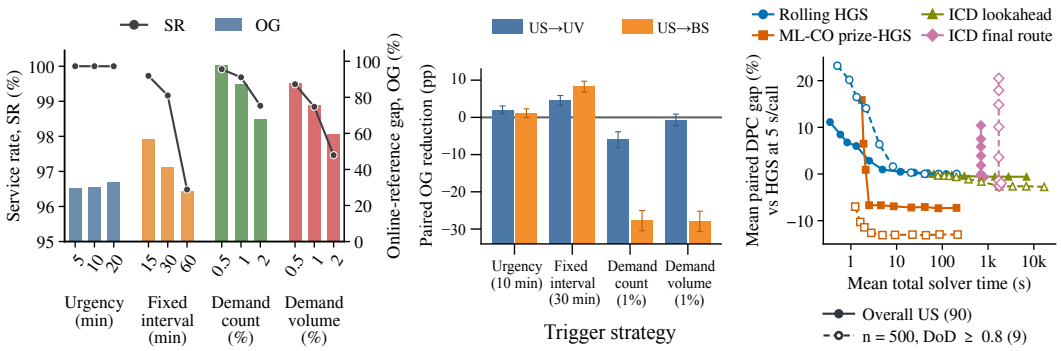
Planning triggers determine when routing is performed. This section first examines their effects on route consolidation, then evaluates whether routing methods respond consistently to changes in decision timing. Full results appear in Appendix E.

Planning triggers shape the trade-off between service responsiveness and route consolidation. Frequent fixed-interval replanning improves service but increases DPC, while Fix-60 lowers DPC at a reduced mean SR of 96.48%. Demand-count and demand-volume triggers respond to accumulated workload, but low thresholds can split concentrated arrivals into small planning batches. Consequently, these rules obtain smaller consolidation gains from strong bursts than fixed-interval and urgency triggers. Urgency-based triggering allow requests to accumulate while responding to approaching deadlines. Urgency-10 achieves full service and reduces DPC by 7.43% relative to Fix-30, with mean customers per trip increasing from 2.86 to 3.17.

The average benefits of Urgency-10 extend to the frozen neural constructors: DPC decreases by 5.78–6.99% relative to each model's Fix-30 baseline, while SR improves to above 99.9% (Table 12 in Appendix E.1). Threshold sensitivity is also consistent across methods: reducing the urgency threshold from 20 to 5 minutes lowers DPC for HGS and all four neural constructors. These shared trends show that adjusting decision timing can improve deployed routing methods without retraining.

4.5 Computational Efficiency

To assess quality-runtime trade-offs, experiments evaluate rolling HGS, ML-CO, and ICD under 0.01-5 s time limits for each routing search. For ICD, there are two steps of route searching. The



(a) Trigger sensitivity (b) Trigger–demand interaction (c) Search-budget sensitivity

Figure 3: Effects of decision timing and search budget. Panel (c) reports mean total solver time per instance. ICD stages are scanned separately; the final-routing scan fixes lookahead at 0.5 s/call.

lookahead routing is solving sampled future-demand scenarios to inform dispatch decisions, and the final-routing is conducting routes over selected requests. So these two routing time is separately experimented. Each limit is varied while the other is held fixed. On 90 US instances, SR remains unchanged across budgets.

Saturation is defined as mean paired DPC degradation not exceeding 0.5% relative to each sweep's max search time result. Firstly, the experiment shows the different saturating speed. ICD lookahead saturates at the lowest search budget, followed by ML-CO; rolling HGS and ICD final routing require larger budgets. After saturation, ML-CO offers the best quality-runtime trade-off: it is faster than rolling HGS (8.6 vs. 12.1 s per instance) and reduces DPC by 6.93% relative to 5 s/call HGS. ICD improve DPC by less than 0.4% against this reference but take over ten times as long as saturated HGS. ICD remains costly because each decision requires 90 lookahead routing solves.

Separately considering instances with $n = 500$ and $\mathrm{DoD} \geq 0.8$, ML-CO saturates at a lower budget than in the overall evaluation, while ICD lookahead needs a higher budget. HGS and ICD final routing retain the same saturation budgets. At saturation, ML-CO's DPC reduction relative to 5 s/call HGS increases from 6.93% to 12.61%, while its runtime falls from 8.6 to 2.8 s per instance. ICD lookahead's DPC reduction also grows, from 0.19% to 2.64%, but its runtime rises from 161.0 to 1,758.7 s per instance. Thus, ML-CO offers larger gains at lower cost on this subset, whereas ICD's increased gains come with much higher runtime.

Detail comparison of other search-based solvers and neural solvers is in Appendix F.

5 Limitations

The experiments focus on settings with sufficient fleet availability and do not examine extreme scenarios where service capacity is insufficient to meet demand. Also, it is limited to delivery tasks with time windows, and the applicability of its findings to other dynamic routing settings remains untested.

6 Conclusion

DVRPTW-BENCH combines controllable demand generation with a modular online protocol. Method performance mainly depends on how models adapt to changing demand scale and distribution. HGS better exploits temporal concentration and large-scale spatial shifts than the evaluated neural constructors, while learned dispatch provides larger gains over original HGS as dynamism increases. ML-CO SL reduces mean DPC relative to rolling HGS by improving route consolidation. Adaptive decision timing further improves online routing: Urgency-10 reduces DPC while maintaining service rates for HGS and frozen neural constructors, with benefits transferring across routing methods without retraining. Neural constructors substantially reduce computation at higher DPC and are more suitable for computational limited scenes.

REPRODUCIBILITY STATEMENT

The benchmark code is available in https://anonymous.4open.science/r/iclr-dvrptw0925. Section 3 describes the benchmark design and online evaluation protocol, while Appendices C and D provide data-generation parameters and method configurations. Section 4.1 reports the hardware and software environment, with computational budgets detailed in Section 4.5 and Appendix G. Appendix A specifies the evaluation subsets, paired comparisons, and bootstrap confidence-interval procedure. Generated benchmark instances, per-instance route records, and trained checkpoints are planned for release with the camera-ready version.

AI USE STATEMENT

ChatGPT assisted with manuscript editing and aggregation of the supplied experimental results.

REFERENCES

Léo Baty, Kai Jungel, Patrick S. Klein, Axel Parmentier, and Maximilian Schiffer. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. *Transportation Science*, 58(4):708–725, 2024. doi: 10.1287/trsc.2023.0107.

Russell W. Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004. doi: 10.1287/opre.1040.0124.

Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, et al. Rl4co: an extensive reinforcement learning for combinatorial optimization benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5278–5289, 2025.

G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958. doi: 10.1287/opre.6.6.791.

Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996. doi: 10.1109/3477.484436.

Ricardo Gama, Ricardo Cunha, Daniel Fuertes, Carlos R del Blanco, and Hugo L Fernandes. Multiagent environments for vehicle routing problems. *INFORMS Journal on Computing*, 2026.

Fred Glover. Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206, 1989. doi: 10.1287/ijoc.1.3.190.

Ahmed Heakl, Yahia Salaheldin Shaaban, Salem Lahlou, Martin Takáč, and Zangir Iklassov. SVRP-Bench: A realistic benchmark for stochastic vehicle routing problem. In *Advances in Neural Information Processing Systems*, volume 38, 2025. Datasets and Benchmarks Track.

Markó Horváth and Tímea Tamási. A general modeling and simulation framework for dynamic vehicle routing. *EURO Journal on Transportation and Logistics*, 14:100159, 2025. doi: 10.1016/j.ejtl.2025.100159.

André Hottung, Mridul Mahajan, and Kevin Tierney. PolyNet: Learning diverse solution strategies for neural combinatorial optimization. In *International Conference on Learning Representations*, 2025.

Lars Magnus Hvattum, Arne Løkketangen, and Gilbert Laporte. Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438, 2006. doi: 10.1287/trsc.1060.0166.

Philip Kilby, Patrick Prosser, and Paul Shaw. Dynamic VRPs: A study of scenarios. Technical Report APES-06-1998, APES Research Group, 1998.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1936–1949, 2022.

Mathias A. Klapp, Alan L. Erera, and Alejandro Toriello. The one-dimensional dynamic dispatch waves problem. *Transportation Science*, 52(2):402–415, 2018. doi: 10.1287/trsc.2016.0682.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Wouter Kool, Laurens Bliek, Danilo Numeroso, Yingqian Zhang, Tom Catshoek, Kevin Tierney, Thibaut Vidal, and Joaquim Gromicho. The EURO meets NeurIPS 2022 vehicle routing competition. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pp. 35–49, 2023. URL `https://proceedings.mlr.press/v220/kool23a.html`.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198, 2020.

Leon Lan, Jasper M. H. van Doorn, Niels A. Wouda, Arpan Rijal, and Sandjai Bhulai. An iterative sample scenario approach for the dynamic dispatch waves problem. *Transportation Science*, 58(4):726–740, 2024. doi: 10.1287/trsc.2023.0111.

Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973. doi: 10.1287/opre.21.2.498.

Snežana Mitrović-Minić and Gilbert Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655, 2004. doi: 10.1016/j.trb.2003.09.002.

George Ninikas and Ioannis Minis. Reoptimization strategies for a dynamic vehicle routing problem with mixed backhauls. *Networks*, 64(3):214–231, 2014. doi: 10.1002/net.21567.

Laurent Perron and Vincent Furnon. OR-Tools. Google Optimization Tools software, 2019.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. doi: 10.1287/trsc.1050.0135.

Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. doi: 10.1137/0206041.

Marlin W. Ulmer, Leonard Heilig, and Stefan Voß. On the value and challenge of real-time information in dynamic dispatching of service vehicles. *Business & Information Systems Engineering*, 59(3): 161–171, 2017. doi: 10.1007/s12599-017-0468-2.

Rinde R. S. Van Lon and Tom Holvoet. Towards systematic evaluation of multi-agent systems in large scale and dynamic logistics. In *PRIMA 2015: Principles and Practice of Multi-Agent Systems*, volume 9387 of *Lecture Notes in Computer Science*, pp. 248–264. Springer, 2015. doi: 10.1007/978-3-319-25524-8_16.

Rinde R. S. Van Lon, Eliseo Ferrante, Ali E. Turgut, Tom Wenseleers, Greet Vanden Berghe, and Tom Holvoet. Measures of dynamism and urgency in logistics. *European Journal of Operational Research*, 253(3):614–624, 2016. doi: 10.1016/j.ejor.2016.03.021.

Barry van Veen, Michael Emmerich, Zhiwei Yang, Thomas Bäck, and Joost Kok. Ant colony algorithms for the dynamic vehicle routing problem with time windows. In *Natural and Artificial Computation in Engineering and Medical Applications*, volume 7931 of *Lecture Notes in Computer Science*, pp. 1–10. Springer, 2013. doi: 10.1007/978-3-642-38622-0_1.

Stacy A. Voccia, Ann Melissa Campbell, and Barrett W. Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019. doi: 10.1287/trsc.2016.0732.

Hai Wang, Lei Zhao, Kaile Yan, Xiaoyi Wang, Yile Liang, and Jinghua Hao. Meituan: Operational-level on-demand food delivery data for the 2025 INFORMS TSL data-driven research challenge. Technical Report 5696423, SSRN, 2025.

Niels A. Wouda, Leon Lan, and Wouter Kool. PyVRP: A high-performance VRP solver package. *INFORMS Journal on Computing*, 36(4):943–955, 2024. doi: 10.1287/ijoc.2023.0055.

Jian Zhang, Kelin Luo, Alexandre M. Florio, and Tom van Woensel. Solving large-scale dynamic vehicle routing problems with stochastic requests. *European Journal of Operational Research*, 306(2):596–614, 2023. doi: 10.1016/j.ejor.2022.07.015.

A Evaluation Roadmap and Statistical Protocol

Table 5 maps each analysis in Section 4 to its evaluation subset and the appendix containing the corresponding detailed results. Method-specific configurations are described in Appendix C.

Table 5: Evaluation roadmap linking the main-text analyses to their evaluation subsets and detailed appendix results.

Main text	Analysis	Evaluation subset / protocol	Details
Sec. 4.2.1	Search-based solver comparison	300 US instances; seven solvers; Fix-30; 1 s/call	Apps. D, F
Sec. 4.2.2	Routing and dispatch comparison	Full 1,350-instance grid; common online protocol	App. D
Sec. 4.3	Method robustness and demand sensitivity	Scale, DoD, and spatiotemporal subsets; Fix-30; main method configurations	App. D.2
Sec. 4.4	Decision-timing sensitivity	1,200 dynamic instances; triggers paired with each method's Fix-30	App. E
Sec. 4.5	Computational efficiency	HGS budget sweep: full grid; seven-solver sweep: US instances	App. F

Relative DPC gaps are likewise computed per instance before aggregation. Paired comparisons align instances by scenario, scale, DoD, and replica; cross-scenario comparisons align scale, DoD, and replica. Reported 95% percentile bootstrap intervals use 4,000 resamples of the per-instance statistics.

B Controlled Data Generator

Section 3.2 summarizes the temporal, spatial, and DoD factors. This appendix defines their distributions, lists model parameters, and illustrates the resulting instances.

Table 6: Model and distribution parameters used by the instance generator. All time quantities are in minutes. $\mathrm{DU}\{a, \dots, b\}$ denotes the discrete uniform distribution over the integers from a to b, inclusive.

Component	Setting
Environment Setting	$\mathcal{A} = [0, 1000]^2$, $H = 1440$, one central depot, capacity $Q = 500$, speed $v = 5$, and a fleet cap of 200
Customer Population	$K = \max(1, \lfloor n/50 \rfloor)$ balanced clusters; $q_i \sim \mathrm{DU}\{1, \dots, 100\}$; $s_i = \mathrm{round}(1 + 0.1 q_i)$
Window Placement	$H_R = 1200$; $\Delta_i \sim \mathrm{DU}\{30, \dots, 120\}$; $w_i \sim \mathrm{DU}\{60, \dots, 120\}$
Strong burst	$K_B = 4$, $\rho = 0.95$, $\sigma_B = 15$; base centers at $\{240, 480, 720, 960\}$ with center-perturbation s.d. 36
Realistic double peak	$W_L = [645, 780]$, $W_D = [1020, 1140]$; $\mu_L \in [675, 735]$, $\sigma_L \in [50, 60]$, $\mu_D \in [1050, 1110]$, $\sigma_D \in [60, 75]$; $\omega_L \in [0.28, 0.32]$, $\omega_D \in [0.19, 0.23]$
Spatial evolution	Four 300-minute periods and hotspot strength $\alpha = 0.8$

B.1 Distribution Details

Customer geometry. Given cluster center μ_c, customer coordinates follow

$$x_i \mid c_i = c \sim \mathcal{N}_{\mathcal{A}}(\mu_c, \sigma_c^2 I), \qquad \sigma_c = \frac{1}{4} \min_{c' \neq c} \|\mu_c - \mu_{c'}\|_2, \tag{10}$$

with $\sigma_c = 250$ when $K = 1$. The stationary and time-varying modes use the same customer population and differ only in the spatial assignment of dynamic temporal records.

Temporal profiles and service windows. The densities in Equation 3 use a temporal horizon of $H_R = 1200$ minutes. The strong-burst profile uses $K_B = 4$, $\rho = 0.95$, and $\sigma_B = 15$ minutes; peak centers are approximately evenly spaced with small random jitter. The realistic double-peak profile

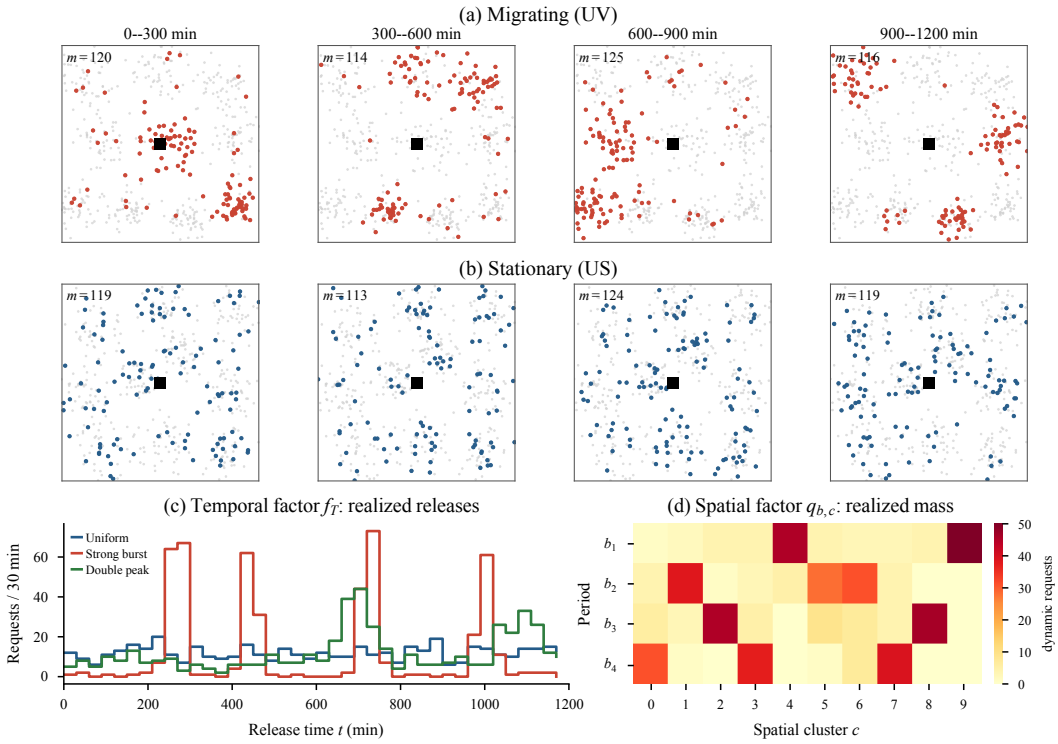


Figure 4: Generator factors at $n = 500$ and DoD 0.95. (a,b,d) show the stationary and time-varying cluster masses used for dynamic-request placement; (c) shows the three latent-time densities.

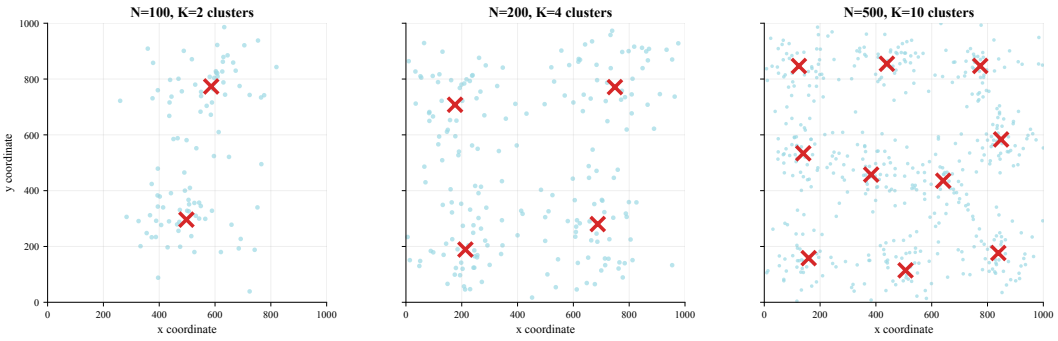


Figure 5: Fixed Gaussian-mixture customer populations at $n = 100$, 200, and 500. Red crosses denote cluster centers.

uses lunch and dinner windows $W_L = [645, 780]$ and $W_D = [1020, 1140]$, with mixture shares $\omega_L \in [0.28, 0.32]$ and $\omega_D \in [0.19, 0.23]$.

For each temporal record, $\tilde{r}_i \sim f_T$ and

$$\begin{aligned} \Delta_i &\sim \mathrm{DU}\{30, \ldots, 120\}, \quad & w_i &\sim \mathrm{DU}\{60, \ldots, 120\}, \\ e_i &= \tilde{r}_i + \Delta_i, & l_i &= e_i + w_i. \end{aligned} \tag{11}$$

The distribution parameters are summarized in Table 6. Dynamic requests use $r_i = \tilde{r}_i$, while static requests use $r_i = 0$; both retain the service window anchored to $\tilde{r}_i$. This construction reflects on-demand delivery, where service windows follow request creation by a fulfillment lead time (Wang et al., 2025).

Dynamic spatial distribution. Equation 5 defines the cluster probabilities for four equal periods, with hotspot strength $\alpha = 0.8$. When $K \geq 4$, each cluster is active in exactly one period, with the numbers of active clusters approximately balanced across periods.

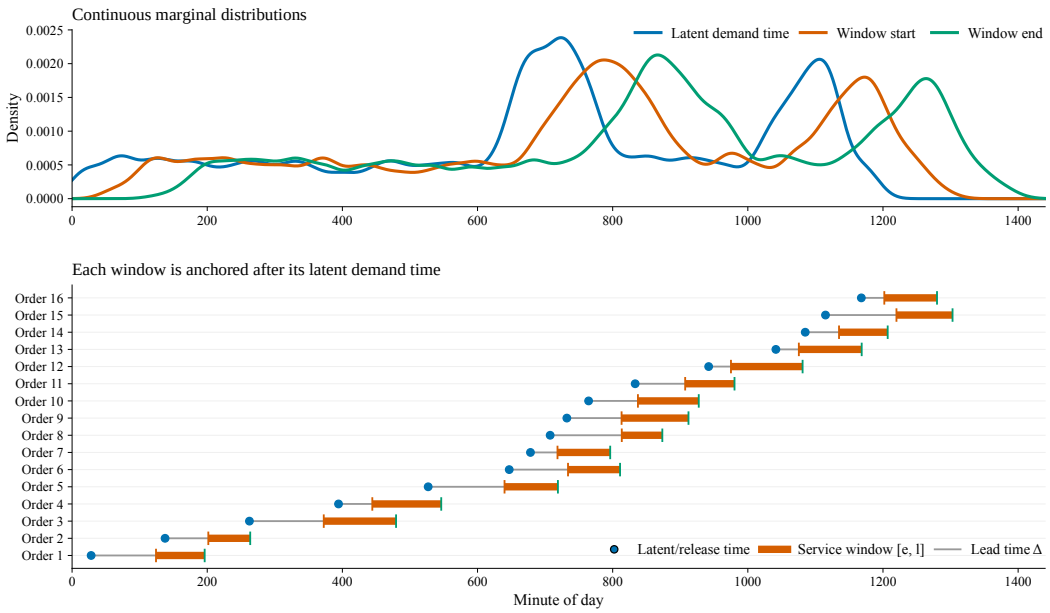


Figure 6: Latent demand times and service-window placement. The upper panel compares their marginal densities; the lower panel shows sampled lead times and window widths.

After a cluster is selected, the temporal record is assigned without replacement to an unused customer node in that cluster.

For dynamic records, the temporal–spatial distribution is

$$p_{\mathrm{dyn}}(t, c, x) = f_T(t) q_{b(t),c} p(x \mid c), \tag{12}$$

where $b(t)$ is the period containing latent demand time t. Static records are randomly paired with the remaining customer nodes.

Depot reachability. Each customer satisfies

$$r_i + \left\lceil \frac{d_{\min}(i)}{v} \right\rceil \leq l_i, \tag{13}$$

where $d_{\min}(i)$ is the distance from customer i to the depot.

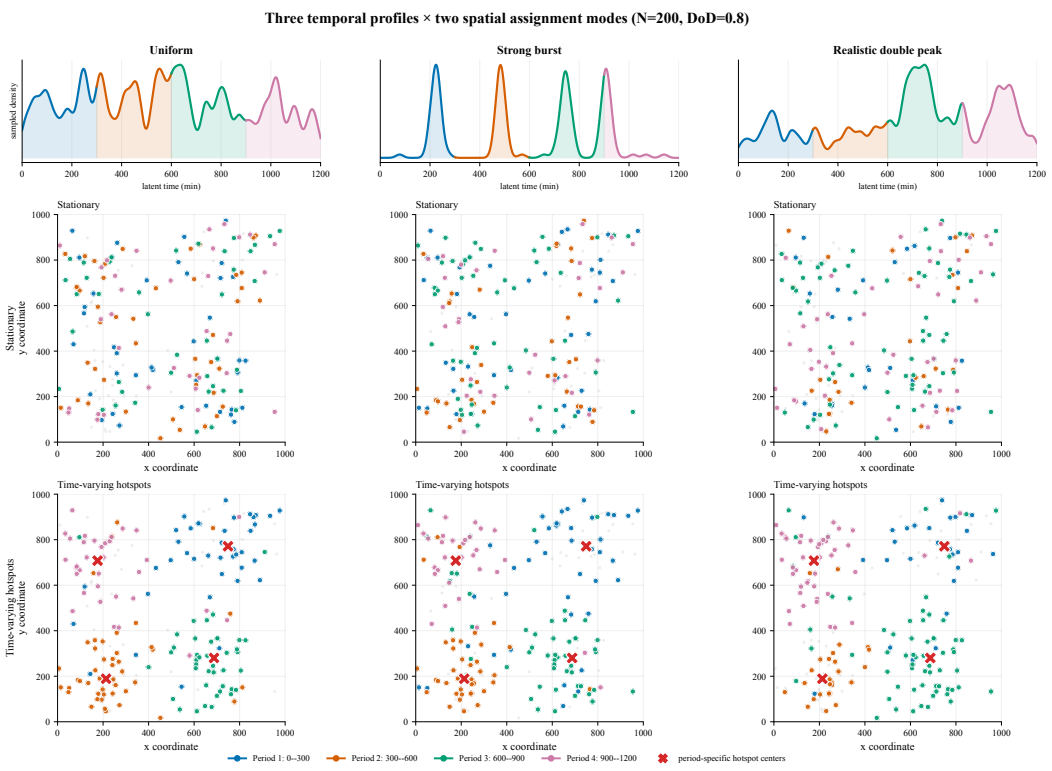


Figure 7: The 3×2 temporal–spatial design at $n = 200$ and DoD 0.8. Columns show the three latent-time profiles; the lower rows visualize stationary and time-varying assignments of dynamic records. Colors denote the four temporal periods.

C Method Configurations

This part introduces the learning-based models and offline reference.

Neural constructors. The neural constructors introduced in Section 3.4 are evaluated under the same online protocol. Training uses the RL4CO `MTVRPGenerator vrptw` preset (Berto et al., 2025), with vehicle capacity 500 and demands sampled from $\{1, \ldots, 100\}$. All four policies use 128-dimensional embeddings, three encoder layers, and eight attention heads. PolyNet uses 40 policy heads, POMO uses eight starts, SymNCO uses eight augmentations, and Attention uses a rollout baseline.

Mixed-scale models are trained over $n \in \{20, 50, 100, 200\}$. Each epoch uses the same 100,000 training instances, with 25,000 instances per scale. Each balanced update contains 32 samples per scale per GPU. Training uses Adam with learning rate 10^{-4} for up to 400 epochs. Mixed-scale checkpoints are selected by maximizing the equally weighted mean raw reward on an independent validation set of 180 instances spanning scales 20–200.

Fixed-scale models are trained separately at $n \in \{20, 50, 100, 200, 500\}$, with the training scale matching the nominal benchmark size at evaluation. Each model uses 100,000 training instances of the corresponding scale, with separate validation and test sets containing 200 instances each. Training runs for up to 400 epochs using the same network configurations as the mixed-scale models. The 1,350 formal benchmark instances are excluded from checkpoint selection for both training settings.

These models are therefore evaluated as constructive snapshot solvers rather than end-to-end dynamic policies: they receive neither previous plans nor vehicle states.

Learned dispatch policy. The evaluated ML-CO policy follows the implementation introduced in Section 3.4. The prize-collecting HGS used to determine the final dispatch set receives a one-second budget per call.

Besides the released checkpoint, we train four variants using different training distributions. The SL variant is trained on small scales and low dynamism: $n \in \{20, 50\}$ and $\mathrm{DoD} \in \{0.2, 0.5\}$. The LH variant focuses on large-scale and high-DoD instances. The GEN variant covers all five scales and multiple DoD levels. The GEN-BAL2 variant additionally balances the observation quota across problem scales.

Anticipatory dispatch with instance configuration. The evaluated ICD configuration follows the iterative sample-scenario policy introduced in Section 3.4. At each decision epoch, the policy evaluates dispatch decisions using future demand sampled from the instance's demand-generation parameters. This configuration gives the sampler privileged access to instance-level distributional information. Actual unreleased requests remain hidden. Each decision consists of three sampling rounds with 30 scenarios per round.

For each sampled scenario, we solve the corresponding routing problem using HGS initialized with complete feasible nearest-neighbor (NN) solutions. The per-scenario search budget is limited to $3/90$ seconds, resulting in a total lookahead budget of three seconds across all 90 scenario solves at each decision epoch. The route selected for actual execution is optimized separately using benchmark HGS with NN initialization and a one-second budget per call.

The original ICD implementation uses a much larger scenario-routing budget, which is not directly comparable with the computational scale of the other online methods. Therefore, we adopt the above short-budget configuration to provide a consistent online evaluation setting.

Offline reference. The release-time-aware HGS provided by the EURO/NeurIPS dynamic VRPTW benchmark is used as the offline reference. Unlike online methods, it receives the complete realized request set and solves a static problem under a 30-second budget. Each route satisfies $\delta_R \geq \max_{i \in R} r_i$, together with the same release-time, time-window, and capacity constraints used by the online methods. Its DPC is reported as a heuristic full-information reference.

D Complete Results and Demand-Regime Analysis

This appendix brings together the detailed evidence for the method comparison and demand-regime analysis in Sections 4.2.2 and 4.3.

D.1 Overall Method Performance

Table 7 reports all 22 evaluated configurations. Service rate, DPC, the paired gap to rolling HGS, and measured solver time summarize their overall quality and computational cost.

Table 7: All 22 configurations on the 1,350-instance grid under the main evaluation protocol. ΔDPC is the mean per-instance gap relative to rolling HGS.

Overall comparison

Method	SR (%)	DPC	ΔDPC (%) [95% CI]	Time (s)
Rolling HGS	99.26	313.8	+0.00 [0.00, 0.00]	23.0
Rolling OR-Tools	99.26	320.5	+3.72 [3.29, 4.16]	23.9
Rolling ALNS	99.26	325.2	+5.77 [5.31, 6.24]	24.9
Rolling LK	99.26	346.6	+15.57 [14.54, 16.61]	22.9
Rolling Tabu Search	99.26	348.9	+16.53 [15.48, 17.58]	22.9
Rolling ACO	99.26	349.1	+17.82 [16.45, 19.20]	24.2
Rolling NN+2-opt	99.26	384.5	+27.70 [26.75, 28.66]	< 0.1
PolyNet (mixed)	99.23	355.4	+15.10 [14.68, 15.54]	0.9
SymNCO (mixed)	99.23	355.5	+15.32 [14.88, 15.77]	0.6
POMO (mixed)	99.24	361.3	+16.84 [16.42, 17.27]	0.7
Attention (mixed)	98.99	434.8	+51.23 [48.72, 53.82]	0.7
PolyNet (fixed-scale)	99.24	353.5	+14.00 [13.63, 14.36]	0.6
SymNCO (fixed-scale)	99.24	369.5	+19.33 [18.78, 19.88]	0.6
POMO (fixed-scale)	99.23	370.9	+19.67 [19.14, 20.18]	0.6
Attention (fixed-scale)†	98.98	443.7	+52.14 [49.91, 54.36]	0.6
ML-CO official	99.26	308.9	+3.56 [2.02, 5.07]	42.9
ML-CO SL	99.26	283.4	−6.93 [−7.61, −6.24]	39.8
ML-CO GEN-BAL2	99.26	297.0	−1.36 [−2.41, −0.32]	41.1
ML-CO GEN	99.26	291.9	−2.09 [−3.39, −0.82]	42.3
ML-CO LH	99.26	293.4	−1.77 [−3.17, −0.39]	43.8
ICD, instance config	99.26	309.8	−1.15 [−1.26, −1.05]	164.1
Release-aware HGS (offline)	100.00	229.1	−25.55 [−26.13, −24.97]	30.0

Fixed-scale selects the checkpoint matching the nominal test scale. †For Attention, matched checkpoints are used through $n = 200$. A single-scale $n = 500$ model was also trained under the same training budget, but exhibited severe performance degradation and was therefore excluded from the reported comparison; the completed mixed-scale checkpoint is used at $n = 500$ instead. Time is measured solver time per instance. Release-aware HGS is the offline reference.

D.2 Demand-Regime Sensitivity

Table 8(a) compares methods within each dynamic scenario, while panel (b) measures each method's own response to a change in demand regime. Under US→BS, rolling HGS reduces DPC by 19.91%, compared with 15.20–16.33% for PolyNet, SymNCO, and POMO. Their relative gaps consequently increase even though their own DPC decreases. The spatial contrasts show a similar difference in the gains from time-varying hotspots on large instances.

Table 8: Method sensitivity to spatiotemporal demand. Panel (a) compares each method with rolling HGS within a scenario. Panel (b) reports the paired change in each method's own DPC between scenarios.

(a) Gap relative to rolling HGS (%)

Method	US	UV	BS	BV	RS	RV
Rolling OR-Tools	+1.62	+2.71	+1.99	+4.38	+1.54	+2.88
Rolling ALNS	+2.57	+4.67	+3.40	+7.33	+2.56	+5.09
PolyNet (mixed)	+8.70	+16.06	+15.39	+25.33	+9.59	+17.63
SymNCO (mixed)	+8.48	+15.79	+15.26	+25.91	+9.45	+17.55
POMO (mixed)	+10.62	+19.84	+15.90	+25.99	+11.12	+20.80
Attention (mixed)†	+22.00	+39.44	+38.85	+70.99	+34.41	+60.83
ML-CO official	−5.45	−9.74	+0.92	−0.16	−5.63	−9.01
ML-CO SL	−12.99	−11.82	−7.72	−6.20	−12.55	−11.52
ML-CO GEN-BAL2	−8.29	−8.93	−5.10	−3.41	−8.33	−8.79
ML-CO GEN	−10.58	−12.20	−5.79	−4.60	−10.84	−11.62
ML-CO LH	−10.25	−13.91	−4.28	−5.66	−10.26	−13.20
ICD, instance config	−0.55	−1.91	−1.88	−1.48	−0.77	−1.75
Release-aware HGS (offline)	−29.02	−32.01	−25.04	−26.11	−28.63	−31.90

(b) Paired scenario-transition change in DPC (%)

Method	US→BS	UV→BV	US→RS	UV→RV	US→UV	BS→BV	RS→RV
Rolling HGS	-19.91	-24.73	-3.83	-3.74	-15.61	-13.50	-14.49
Rolling OR-Tools	-19.69	-23.52	-3.89	-3.57	-15.51	-12.25	-13.87
Rolling ALNS	-19.37	-22.83	-3.85	-3.37	-14.72	-11.23	-13.09
PolyNet (mixed)	-15.33	-18.82	-3.09	-2.48	-11.17	-9.14	-9.98
SymNCO (mixed)	-15.20	-18.20	-2.99	-2.35	-10.84	-8.45	-9.51
POMO (mixed)	-16.33	-20.80	-3.44	-2.96	-9.82	-8.51	-8.83
Attention (mixed)†	-10.06	-7.90	+4.75	+10.00	-6.29	-1.21	-3.91
ML-CO official	-14.30	-16.62	-3.67	-3.03	-10.24	-8.69	-10.68
ML-CO SL	-15.21	-19.83	-3.25	-3.43	-12.57	-12.22	-12.23
ML-CO GEN-BAL2	-17.21	-20.07	-3.69	-3.63	-11.71	-11.12	-11.95
ML-CO GEN	-15.60	-18.15	-3.80	-3.18	-10.31	-9.87	-10.06
ML-CO LH	-14.34	-17.35	-3.42	-2.94	-9.92	-8.80	-9.88
ICD, instance config	-21.02	-24.41	-4.04	-3.54	-16.27	-13.12	-15.17
Release-aware HGS (offline)	-15.39	-18.06	-3.10	-3.54	-12.94	-10.49	-12.96

Panel (a) averages per-instance method-to-HGS gaps; panel (b) averages paired per-instance percentage changes from the first scenario to the second. All entries use dynamic instances. In (a), US/BS/RS each contain 250 instances; UV/BV/RV each contain 150 with $n \geq 100$. In (b), temporal contrasts use 250 stationary or 150 time-varying pairs; spatial contrasts use 100 pairs at $n \geq 200$. Pairs match scale, DoD, and replica. Negative values in (a) favor the listed method; negative values in (b) favor the second scenario. †Attention uses its mixed-scale checkpoint.

Table 9 separates nominal scale and DoD as two marginal summaries. The neural constructors' gaps to HGS grow with scale, while ML-CO SL and ICD improve on HGS in every positive-DoD slice. Rolling HGS has lower DPC than all evaluated ML-CO variants on the static slice. Comparisons between training configurations are discussed in Appendix D.3.

Table 9: Mean per-instance DPC gap relative to rolling HGS (%) by (a) nominal scale and (b) DoD on the 1,350-instance grid. Scale columns contain 180 instances each for $n = 20, 50$ and 330 each for $n = 100, 200, 500$. The static DoD column contains 150 instances; each positive-DoD column contains 240. The two panels are separate marginal summaries.

(a) By nominal scale

Method	n=20	50	100	200	500
Rolling OR-Tools	+0.11	+0.80	+1.74	+3.36	+9.61
Rolling ALNS	+0.75	+2.83	+4.44	+6.49	+10.72
PolyNet (mixed)	+6.43	+9.31	+15.32	+16.67	+21.22
SymNCO (mixed)	+7.26	+10.10	+16.89	+16.69	+19.63
POMO (mixed)	+8.50	+10.65	+18.87	+19.02	+20.57
Attention (mixed)	+16.45	+32.91	+53.25	+58.29	+71.10
ML-CO official	+19.86	+17.38	+3.46	−0.87	−8.34
ML-CO SL	−1.47	−4.52	−9.41	−8.79	−6.89
ML-CO GEN-BAL2	+7.07	+4.92	−3.07	−3.73	−5.28
ML-CO GEN	+9.65	+6.51	−5.00	−4.88	−7.48
ML-CO LH	+14.94	+10.89	−4.14	−6.23	−10.95
ICD, instance config	−0.50	−0.76	−2.09	−1.01	−0.91
Release-aware HGS (offline)	−15.04	−20.73	−29.69	−29.19	−26.11

(b) By degree of dynamism

Method	0	0.2	0.5	0.8	0.9	0.95
Rolling OR-Tools	+14.90	+7.56	+2.34	+0.71	+0.64	+0.35
Rolling ALNS	+20.61	+10.77	+4.96	+1.74	+1.26	+0.84
PolyNet (mixed)	+20.78	+15.09	+13.53	+13.90	+14.80	+14.67
SymNCO (mixed)	+23.35	+16.09	+13.88	+13.34	+14.41	+13.88
POMO (mixed)	+22.21	+17.15	+15.61	+15.76	+16.20	+16.13
Attention (mixed)	+131.02	+77.30	+47.02	+29.61	+26.64	+25.70
ML-CO official	+67.90	+15.55	−5.33	−10.68	−11.08	−10.87
ML-CO SL	+22.57	−2.47	−11.16	−13.41	−13.33	−12.73
ML-CO GEN-BAL2	+45.14	+6.78	−8.24	−11.64	−11.86	−10.88
ML-CO GEN	+54.95	+8.71	−9.81	−14.98	−15.16	−14.86
ML-CO LH	+58.14	+9.35	−10.04	−15.04	−15.46	−15.11
ICD, instance config	+0.14	−0.34	−0.88	−1.54	−1.69	−2.11
Release-aware HGS (offline)	−2.07	−24.95	−31.19	−29.76	−28.52	−27.99

Solver sensitivity to scale and dynamism. Figures 8 and 9 visualize the marginal results in Table 9. Among neural constructors, PolyNet has the lowest mean gap at $n \leq 200$, while SymNCO has the lowest at $n = 500$. PolyNet also leads at DoD ≤ 0.5, with SymNCO leading at DoD ≥ 0.8; POMO remains close, while Attention has a substantially larger gap. For dispatch, SL improves on HGS at every scale in the marginal comparison, and ICD provides smaller gains across these scale averages. These rankings describe the reported slice means; they do not establish statistical differences between closely performing methods.

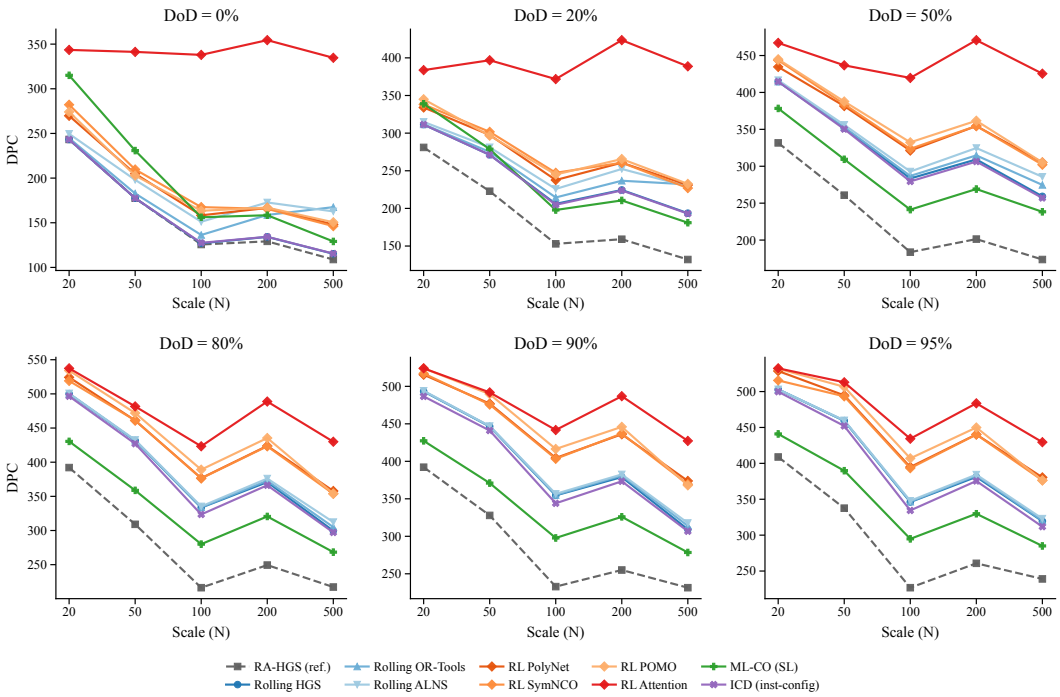


Figure 8: Method performance across nominal instance sizes at fixed DoD levels, visualizing Table 9(a). Each panel fixes DoD, and points show mean DPC over replicas at each instance size. Neural constructors use mixed-scale checkpoints; ML-CO uses SL, and ICD uses instance-configuration sampling. The gray dashed curve denotes the offline RA-HGS reference. Lower values are better.

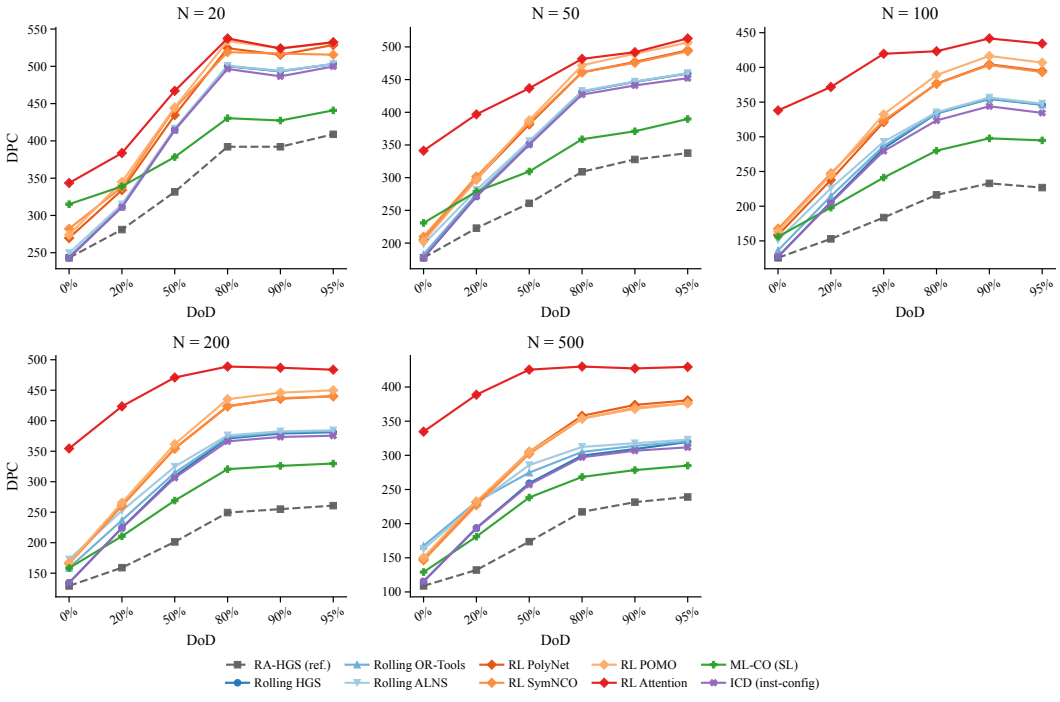


Figure 9: Method performance across DoD levels at fixed nominal instance sizes, visualizing Table 9(b). Each panel fixes the instance size, and points show mean DPC over replicas at each DoD level. Method configurations match Figure 8; the gray dashed curve denotes the offline RA-HGS reference. Lower values are better.

D.3 Training Distribution and Transfer

This subsection isolates training-distribution comparisons from the fixed-configuration demand analysis in Section 4.3.

Neural route constructors. Table 10 compares scale-matched checkpoints with mixed-scale checkpoints trained on $n \in \{20, 50, 100, 200\}$ using paired US instances. Pooled results for PolyNet, SymNCO, and POMO favor scale-matched training at DoD $= 0$ and mixed-scale training at DoD $= 0.95$. At the latter endpoint, scale-matched checkpoints have 0.23%, 4.46%, and 2.76% higher DPC, respectively. Mixed-scale training therefore benefits SymNCO and POMO more than PolyNet in this comparison. The reversal is small for PolyNet and varies across individual scales. Attention favors mixed-scale training at both endpoints over its available scales. These results show that training-scale sensitivity differs across models and DoD levels.

Table 10: Scale-matched versus mixed-scale checkpoints on US instances: mean paired DPC difference (%) of the scale-matched checkpoint relative to the mixed-scale checkpoint. Negative values favor scale-matched training. Ten replicas per cell; Pooled averages the available scales. All same-instance pairs contribute.

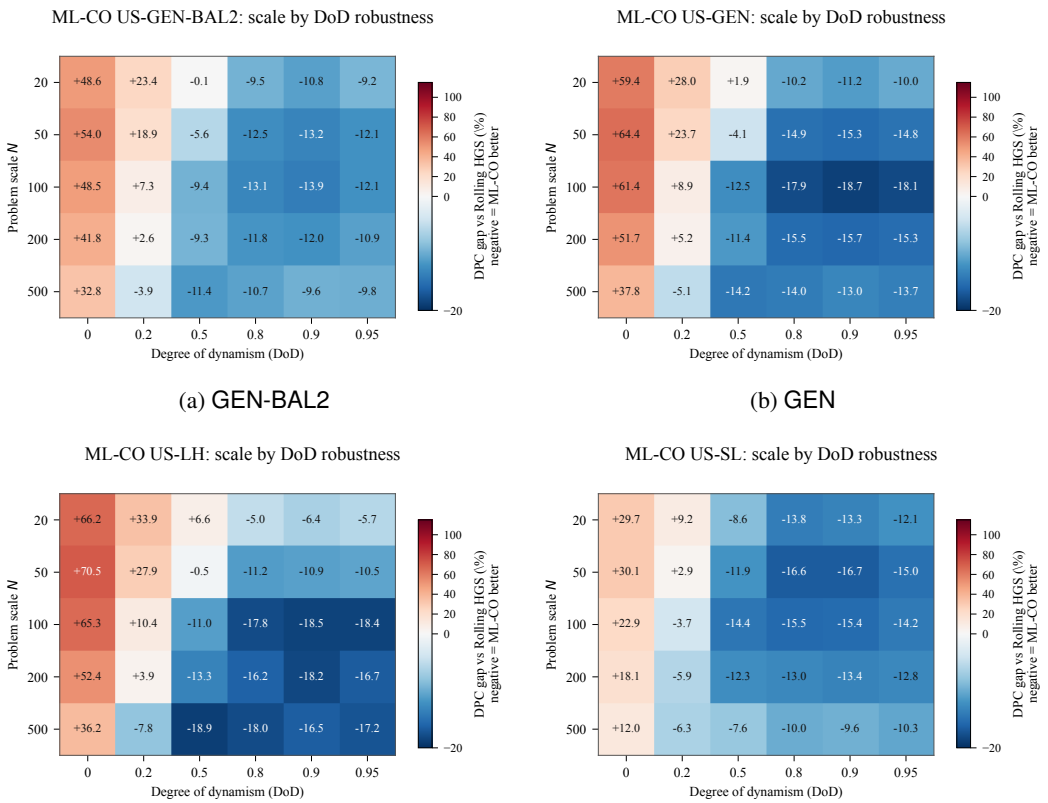
Model	DoD	n=20	50	100	200	500	Pooled
PolyNet	0	+1.69	-3.05	+0.20	-2.06	-7.86	-2.21
	0.95	+0.19	+0.36	+1.21	-0.91	+0.32	+0.23
SymNCO	0	-6.39	-2.01	+0.64	+0.88	-5.94	-2.56
	0.95	+1.17	+3.32	+5.18	+6.79	+5.87	+4.46
POMO	0	-6.45	-0.38	+0.42	+0.31	-4.45	-2.11
	0.95	-0.75	+0.17	+4.32	+3.28	+6.80	+2.76
Attention[†]	0	+5.67	+6.89	-14.07	+27.84	–	+6.58
	0.95	+0.00	-0.73	+2.01	+13.03	–	+3.58

[†]No single-scale 500-customer Attention checkpoint is available; its pooled column covers $n \leq 200$.

Learned dispatch. ML-CO exhibits a trade-off between performance in specific regimes and consistency across demand conditions. Among the four retrained variants, SL has the lowest pooled DPC gap to HGS (-6.93%; Table 7). The LH variant, trained on large-scale and high-DoD instances, achieves larger reductions at $n = 500$ (10.95%, compared with 6.89% for SL) and at DoD $= 0.95$ (15.11%, compared with 12.73%). These are separate marginal comparisons. At $n = 20$, LH has 14.94% higher DPC than HGS, while SL reduces DPC by 1.47% (Table 9).

SL is the only evaluated ML-CO variant with lower mean DPC than HGS in every nominal-scale slice and every positive-DoD slice. It also improves on HGS in every reported dynamic-scenario average (Tables 9 and 8). These results support broader performance consistency for SL across the evaluated conditions, alongside stronger regime-specific gains for LH.

Figure 10 preserves the joint scale–DoD structure for the four ML-CO training distributions on US instances. Together with the marginal results in Table 9, these heatmaps show where each training distribution transfers well and where its pooled performance hides weaker cells. Negative gaps indicate lower DPC than rolling HGS.



(a) GEN-BAL2 (b) GEN

(c) LH (d) SL

Figure 10: ML-CO transfer on US instances across training distributions. Each panel reports the DPC gap relative to rolling HGS across scale and DoD (%): (a) GEN-BAL2, (b) GEN, (c) LH, and (d) SL.

Figure 11 complements the rolling-HGS comparison with online–reference gaps to RA-HGS on the same US grid. All displayed cell averages remain above this full-information heuristic reference. At DoD $= 0.95$, SL has the smallest gap at $n = 20$ (8.0%), whereas LH has the smallest gap at $n = 500$ (9.8%, compared with 19.1% for SL). The residual gap therefore varies with both problem scale and training distribution.

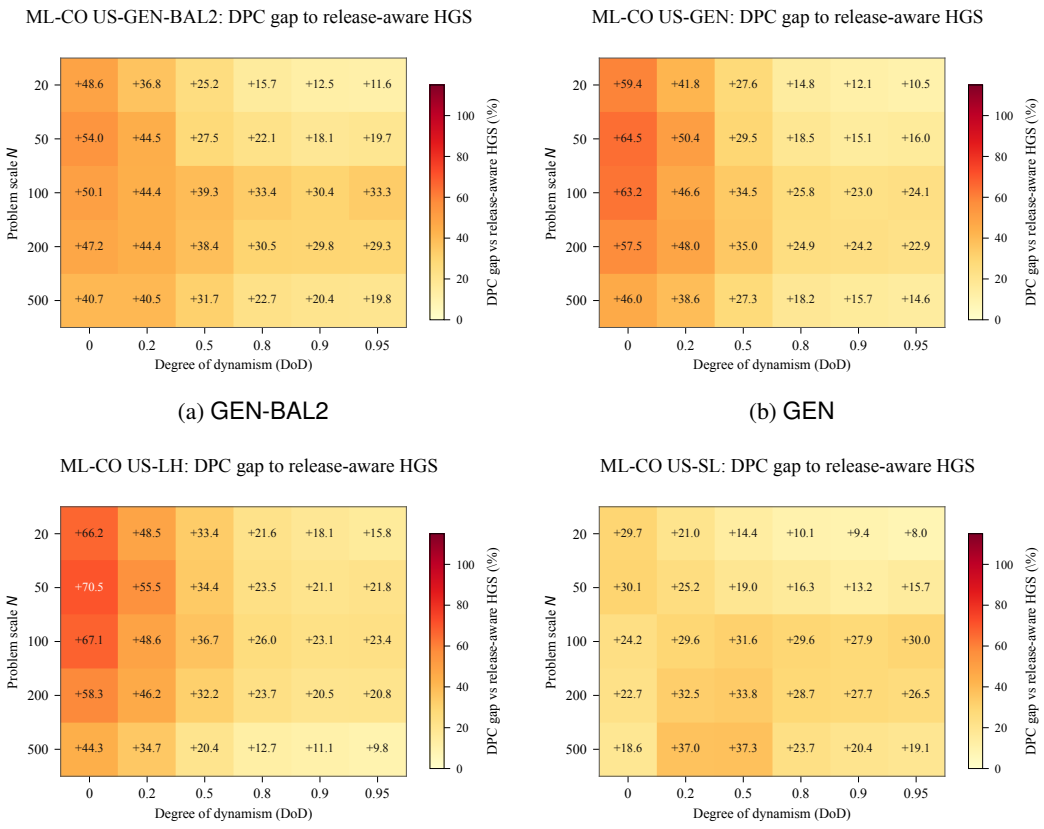


Figure 11: ML-CO online–reference gaps on US instances across training distributions. Each panel reports OG relative to the full-information RA-HGS reference across scale and DoD (%): (a) GEN-BAL2, (b) GEN, (c) LH, and (d) SL. Smaller values indicate lower DPC relative to this heuristic reference.

D.4 Online Workloads and Route Structure

Table 11 fixes rolling HGS and Fix-30 to characterize the planning inputs associated with the demand-regime results. At DoD 0.95, nominal daily sizes of 20–500 customers yield mean post-initial planning pools of only 1.63–13.63 requests. On large dynamic instances, strong bursts increase the mean pool from 6.16 to 9.99 requests and customers per trip from 2.97 to 4.13. US and UV have similar pool sizes but different customers per trip, showing that pool size alone does not describe the realized route structure. These workload summaries help interpret the method-specific responses in Table 8(b).

Table 11: Online planning workloads and realized batching under rolling HGS with Fix-30. Panel (a) varies nominal scale at DoD 0.95; panel (b) compares dynamic demand regimes at $n \in \{200, 500\}$.

(a) By nominal scale at DoD 0.95

n	Inst.	Calls/day	Pool/call	Cust./trip
20	30	13.63	1.63	1.30
50	30	22.33	2.43	1.61
100	60	28.83	4.09	2.25
200	60	34.72	6.43	2.50
500	60	38.60	13.63	3.12

(b) By scenario on large dynamic instances

Scenario	Inst.	Calls/day	Pool/call	Median	IQR	Cust./trip
US	100	39.4	6.15	5	3–9	2.73
UV	100	39.7	6.17	5	3–9	3.21
BS	100	23.6	10.01	3	1–14	3.84
BV	100	23.7	9.97	3	1–13	4.42
RS	100	38.2	6.34	4	2–8	2.86
RV	100	38.2	6.31	4	2–8	3.33
Uniform (US+UV)	200	39.6	6.16	5	3–9	2.97
Strong burst (BS+BV)	200	23.6	9.99	3	1–13	4.13
Realistic (RS+RV)	200	38.2	6.32	4	2–8	3.10

Pool/call is the instance-macro mean size of the complete pending input recorded before each actual solver call, excluding the initial time-zero call. Median and IQR are call-weighted over the same post-initial calls. Calls/day counts all calls; Cust./trip summarizes executed routes. The last three rows of (b) pool the corresponding spatial variants.

E Decision Timing Sensitivity

This appendix compares planning triggers across routing methods and examines how their effects depend on demand.

E.1 Trigger Comparison

Table 12 summarizes service rates and DPC changes for representative triggers across rolling HGS and the four neural constructors.

Table 12: Decision timing across solvers on 1,200 dynamic instances.

	SR (%)				ΔDPC vs Fix-30 (%)		
Solver	Fix-30	Urgency-10	Count 1%	Volume 1%	Urgency-10	Count 1%	Volume 1%
Rolling HGS	99.17	100.00	99.68	98.84	−7.43	+32.43	+24.56
Attention	99.06	99.93	99.59	98.75	−6.99	+17.79	+14.63
POMO	99.16	99.96	99.68	98.84	−6.54	+24.66	+20.48
SymNCO	99.15	99.94	99.68	98.84	−5.78	+24.04	+18.96
PolyNet	99.15	99.94	99.68	98.84	−6.45	+23.79	+18.78

Table 13(a) compares twelve trigger settings for rolling HGS with a one-second per-call budget on 1,200 dynamic instances. Panel (b) summarizes transfer to four frozen neural constructors using mixed-scale checkpoints trained under Fix-30, without trigger-specific retraining. All relative DPC changes are computed per instance against the same solver's or model's Fix-30 run within its respective sweep.

For HGS, Urgency-10 lowers paired DPC by 7.43% relative to Fix-30 while achieving full service and using fewer planning calls. Fix-60 yields lower mean DPC but reduces mean SR to 96.48%. The neural constructors also benefit from Urgency-10: paired DPC decreases by 5.78–6.99% and mean SR exceeds 99.9% for every model. Count-1% and Volume-1% increase DPC across all five methods. Urgency therefore provides a favorable quality–service trade-off across the evaluated methods.

Table 13: Trigger comparison on 1,200 dynamic instances per method and setting. Panel (a) reports the rolling-HGS sweep; panel (b) summarizes neural trigger transfer. SR is the mean instance service rate; Calls and Cust./trip are mean planning calls per instance and mean customers per executed trip. Each ΔDPC is the mean per-instance percentage change from the same model's paired Fix-30 run; negative values favor the tested trigger.

(a) Rolling-HGS trigger comparison

Trigger	DPC	SR (%)	Calls	Cust./trip
Fixed clock				
15 min	369.6	99.73	41.2	2.50
30 min	333.1	99.17	26.7	2.86
60 min	302.4	96.48	16.5	3.18
Deadline-driven (urgency), slack				
5 min	304.7	100.00	14.4	3.18
10 min	305.5	100.00	15.7	3.17
20 min	311.9	100.00	19.0	3.11
Load-driven (demand count)				
0.5 %	458.6	99.91	83.9	1.85
1 %	433.7	99.68	63.6	1.97
2 %	390.7	98.87	40.1	2.29
Load-driven (demand volume)				
0.5 %	436.1	99.49	70.4	1.98
1 %	408.1	98.84	51.4	2.17
2 %	372.2	97.46	33.7	2.47

(b) Neural trigger transfer: paired ΔDPC (%)

Method	Urgency-10	Count-1%	Volume-1%
Attention	-6.99	$+17.79$	$+14.63$
POMO	-6.54	$+24.66$	$+20.48$
SymNCO	-5.78	$+24.04$	$+18.96$
PolyNet	-6.45	$+23.79$	$+18.78$

Urgency thresholds are remaining slack in minutes; count and volume thresholds are percentages of the total count and total demand of dynamic requests, respectively.

E.2 Trigger–Demand Interaction

Table 14 separates the HGS results by DoD. At DoD 0, each trigger makes one planning call and yields similar DPC with full service. At every positive DoD, urgency achieves lower DPC than Fix-30 while retaining full service; the count- and volume-based rules have higher DPC. These patterns are consistent with the role of remaining slack in balancing request accumulation and timely dispatch.

Temporal concentration also changes the benefit of each trigger. For HGS, the paired US-to-BS DPC reduction is 14.98–19.92% across fixed-interval and urgency settings, compared with 2.04–3.79% across count- and volume-based settings. Frozen neural constructors show a related pattern: averaging equally across the four models, Count-1% increases DPC relative to Fix-30 by 14.69% under uniform arrivals and 35.18% under strong bursts; Volume-1% increases it by 10.51% and 30.59%, respectively. Under strong bursts, mean customers per trip fall from 2.56 with Fix-30 to 1.71 with Count-1%, consistent with reduced request consolidation under the load-driven trigger.

Urgency gains also depend on scale and demand structure. On the 100 strong-burst instances at $n = 500$ (BS/BV, DoD > 0), Urgency-10 increases paired DPC by 1.98–4.01% for PolyNet, SymNCO, and POMO, while reducing it by 4.42% for Attention; all four retain mean SR above 99.9%. On the 240-instance HGS subset with $n \in \{200, 500\}$ and DoD $\in \{0.9, 0.95\}$, Urgency-10 retains full service and lowers paired DPC by 2.90% relative to Fix-30. These results support urgency as an effective trigger across the evaluated methods on average, with gains that depend on the solver and demand regime.

Table 14: Rolling-HGS trigger sensitivity by DoD on all 1,350 instances. Each cell is DPC / SR (%): mean distance per served customer followed by mean instance service rate. The static column contains 150 instances; each positive-DoD column contains 240. Urgency thresholds are slack in minutes; count and volume thresholds are percentages.

Rule	0	0.2	0.5	0.8	0.9	0.95
Fixed 15 min	159.4 / 100.00	237.4 / 99.95	335.9 / 99.78	412.2 / 99.65	428.1 / 99.57	434.4 / 99.68
Fixed 30 min	159.5 / 100.00	228.9 / 99.75	308.8 / 99.35	367.5 / 98.96	378.2 / 98.84	382.0 / 98.92
Fixed 60 min	159.5 / 100.00	218.0 / 98.94	283.8 / 97.28	330.6 / 95.69	338.8 / 95.22	340.6 / 95.29
Urgency 5 min	160.4 / 100.00	217.5 / 100.00	282.1 / 100.00	335.5 / 100.00	342.7 / 100.00	345.9 / 100.00
Urgency 10 min	160.0 / 100.00	217.1 / 100.00	282.4 / 100.00	336.3 / 100.00	343.3 / 100.00	348.5 / 100.00
Urgency 20 min	159.9 / 100.00	218.3 / 100.00	285.7 / 100.00	343.7 / 100.00	353.6 / 100.00	358.3 / 100.00
Count 0.5 %	159.4 / 100.00	254.0 / 100.00	393.4 / 99.91	525.6 / 99.93	552.0 / 99.86	568.0 / 99.87
Count 1 %	159.6 / 100.00	254.3 / 100.00	387.0 / 99.82	486.4 / 99.58	513.7 / 99.50	527.0 / 99.52
Count 2 %	159.6 / 100.00	249.5 / 99.91	362.3 / 99.44	429.8 / 98.33	452.4 / 98.37	459.6 / 98.32
Volume 0.5 %	159.7 / 100.00	249.9 / 99.85	381.9 / 99.58	492.8 / 99.39	522.8 / 99.35	533.2 / 99.26
Volume 1 %	159.7 / 100.00	245.5 / 99.62	363.6 / 98.85	458.3 / 98.62	482.9 / 98.65	490.3 / 98.48
Volume 2 %	159.7 / 100.00	238.5 / 99.23	338.1 / 97.78	415.3 / 96.98	432.3 / 96.67	436.6 / 96.65

F Computational Budgets and Efficiency

The seven-solver US study evaluates 300 instances at each of ten per-call time budgets. Extending the HGS sweep to the five non-US scenarios gives 1,350 HGS instances per budget. Table 15 presents routing quality and service. Figure 12 compares measured runtime and routing quality across search-based and neural methods. Table 16 decomposes the paired HGS gains by scale and DoD.

F.1 Quality and Service across Budgets

Longer search lowers HGS DPC, with diminishing gains at longer budgets. OR-Tools loses substantial service at the shortest settings, so short-budget DPC should be read jointly with SR.

Table 15: Solution quality and service across per-call time budgets. Each column summarizes its stated evaluation subset.

Budget (s/call)	HGS DPC US $N = 300$	HGS DPC Full grid $N = 1350$	OR-Tools SR (%) US $N = 300$
0.01	399.4	379.2	35.04
0.02	389.1	–	71.30
0.03	–	–	80.07
0.05	379.2	344.3	88.79
0.10	369.2	331.4	93.94
0.20	360.8	322.1	97.45
0.50	357.0	316.2	99.35
1.00	356.6	314.3	99.44
2.00	356.3	–	99.44
5.00	356.0	313.1	99.44

HGS has mean SR 99.44% and 24.29 calls per instance at every US budget setting. Budget-sweep and main-grid runs are separate evaluations. – denotes an unreported DPC statistic; missing values are not interpolated.

Runtime–quality trade-offs. Figure 12 relates measured total solver time to mean OG relative to release-aware HGS. The plotted neural configurations run in less than one second per instance. HGS reaches OG values close to POMO, PolyNet, and SymNCO with a short search, then achieves lower OG as its budget increases. Its gains diminish at longer runtimes. The practical advantage of fast neural construction therefore depends on the search method and budget used for comparison. These quality comparisons should also account for the service losses observed at short budgets.

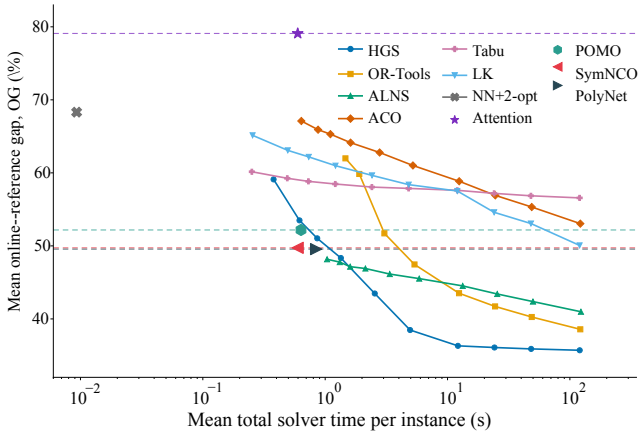


Figure 12: Runtime–quality trade-offs across routing methods. The horizontal axis shows mean total solver time per instance on a logarithmic scale; the vertical axis shows mean OG relative to release-aware HGS. Connected markers show search results at different per-call budgets. Standalone markers show NN+2-opt and the neural constructors. Horizontal dashed lines extend the neural constructors' OG values for visual comparison. Lower runtime and lower OG are preferable.

F.2 Returns to Additional Search

The paired gains in Table 16 increase with nominal scale, whereas the full-grid DoD pattern is non-monotonic. The small additional improvement beyond one second supports the reference setting used in the main comparison. Suitable budgets still depend on the workload. Paired improvements are calculated per instance before averaging.

Table 16: Paired HGS DPC reductions from additional search on the full grid. Each reduction is the mean of the per-instance percentages $100(1 - \mathrm{DPC}_{b_1}/\mathrm{DPC}_{b_0})$ over the stated subset, where $b_1 > b_0$. Positive values indicate lower DPC with the longer budget.

Comparison / subset	Instances	DPC reduction (%)
Overall budget comparisons		
10 ms → 5 s	1,350	14.45
1 s → 5 s	1,350	0.48
By nominal scale: 10 ms → 5 s		
$n = 20$	180	1.20
$n = 50$	180	4.22
$n = 100$	330	6.56
$n = 200$	330	20.23
$n = 500$	330	29.37
By DoD: 10 ms → 5 s		
$\mathrm{DoD} = 0$	150	11.88
$\mathrm{DoD} = 0.2$	240	19.49
$\mathrm{DoD} = 0.5$	240	11.20
$\mathrm{DoD} = 0.8$	240	14.18
$\mathrm{DoD} = 0.9$	240	14.22
$\mathrm{DoD} = 0.95$	240	14.78

G Qualitative Route Visualizations

The aggregate tables report distance and consolidation summaries, but they do not show what the executed plans look like. Figures 13 and 14 therefore visualize executed rolling-HGS plans on matched instances. Figure 13 holds scale, DoD, and replica fixed and varies only the spatiotemporal regime, making the mechanisms of Section 4.3 directly visible: the uniform profile produces many short depot-anchored trips, the strong-burst profile yields fewer and longer routes, and time-varying

hotspots shorten within-route segments relative to the stationary counterpart at the same temporal profile. In this example, DPC decreases from 448.9 under US to 266.9 under BV, and the executed route count falls from 102 to 53, while service remains near saturation. This example is consistent with the DPC reductions under strong bursts and time-varying hotspots in Table 8(b). Figure 14 instead fixes the scenario and shows the state of the same run at four decision times, which illustrates the online protocol of Section 3.3: only released requests are visible to the method, completed routes are immutable, and the planned or active routes at any instant cover a small snapshot rather than the whole day. These route-state counts are descriptive states of one run and are not used as substitutes for the complete planning-pool measurement. The corresponding state counts are given directly in the figure caption.

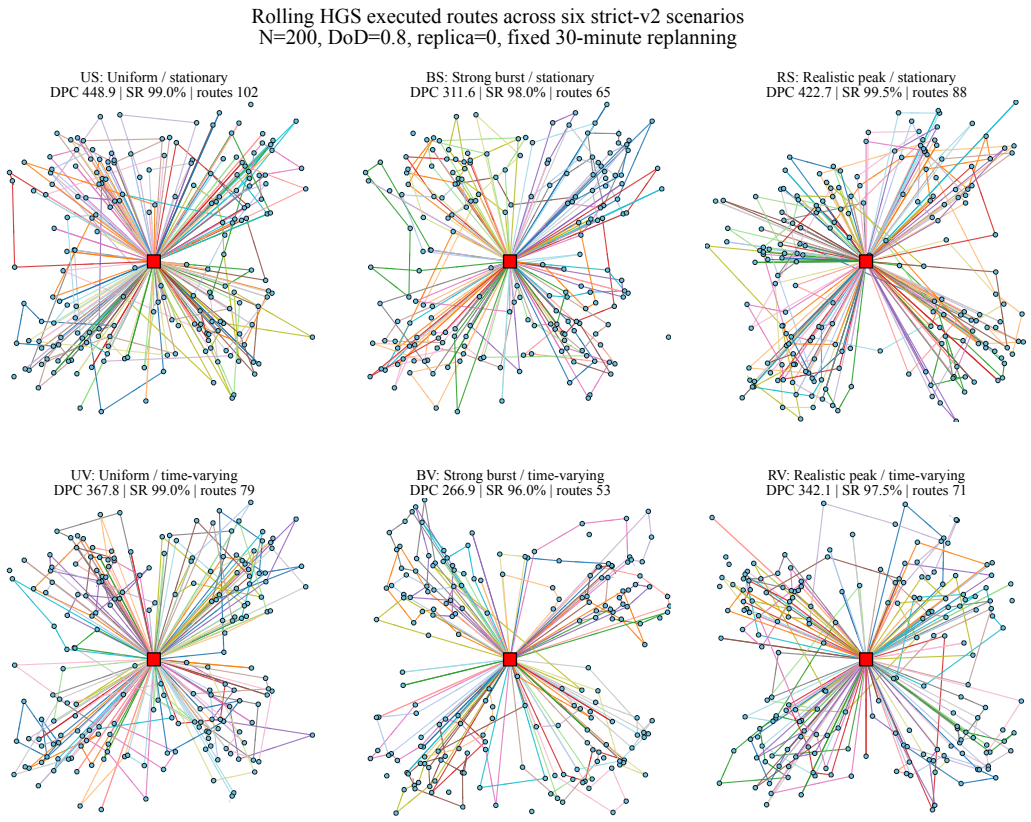


Figure 13: Executed rolling-HGS routes across the six spatiotemporal scenarios at $n = 200$, DoD $= 0.8$, replica 0, under the fixed 30-minute trigger. Light markers are customers, the red square is the depot, and each color is one executed depot-to-depot trip. Panel annotations give DPC, service rate, and executed route count for that instance. For US versus BV, DPC is 448.9 versus 266.9 and the executed route count is 102 versus 53. These single-instance examples illustrate the regime effects; population-level scenario contrasts are reported in Table 8(b).

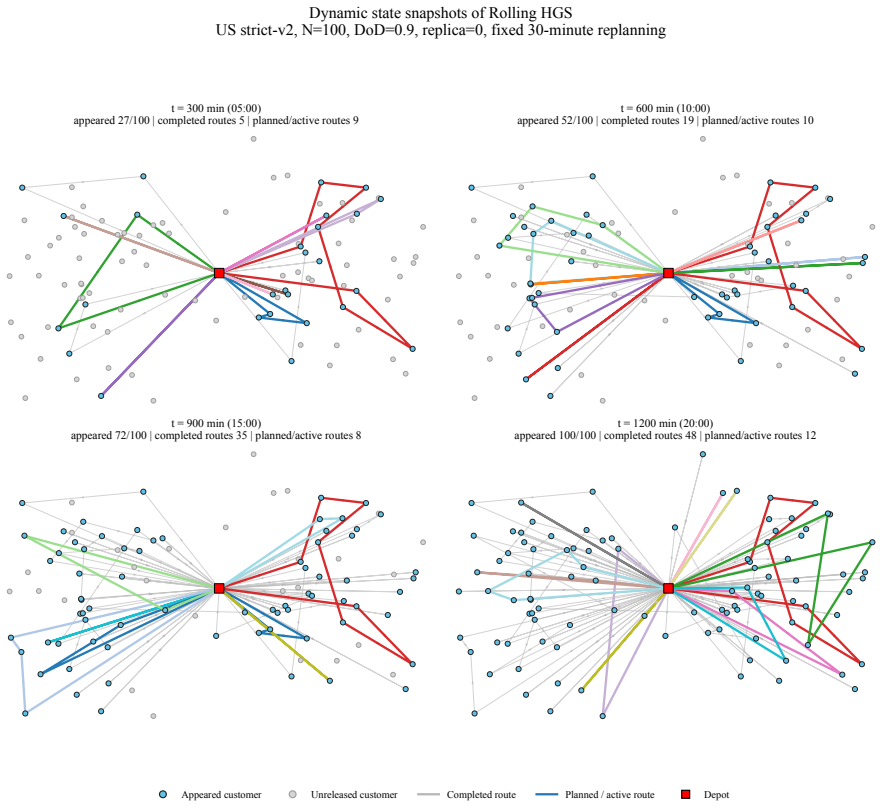


Figure 14: Dynamic state of one rolling-HGS run at four decision times on the US instance with $n = 100$, DoD $= 0.9$, replica 0, under the fixed 30-minute trigger. Gray markers are customers whose release time has not yet arrived, so they are invisible to the method; light markers are released customers; gray lines are completed immutable routes; colored lines are planned or active routes; the red square is the depot. Across the four displayed states, released requests increase from 27 to 100 and completed routes from 5 to 48; simultaneously planned or active routes range from 8 to 12. These are endpoint counts and an observed range, not a reconstruction of intermediate snapshot counts. These variables are distinct from the complete pending input pool in Table 11: the panel illustrates the sequence of online states rather than measuring pool size itself.