NEURAL PERIODIC REOPTIMIZATION FOR DYNAMIC VEHICLE ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic vehicle routing requires repeated route revision as new requests arrive during vehicle execution. Neural construction solvers offer a fast alternative to repeated combinatorial search, but static training does not account for the residual routing states encountered at deployment, which depend jointly on request arrivals, past routing decisions, and irrevocable dispatch commitments. We propose Neural Periodic Reoptimization (NPR), a framework that aligns neural routing solvers with a prescribed periodic execution protocol. NPR represents execution histories as residual routing problems and trains the solver on states encountered during its own protocol-constrained reoptimization rollouts. Its Commitment-Consistent Decoding intersects a commitment mask with the solver's native feasibility mask to preserve committed route prefixes, enabling reoptimization without external search or repair. Experiments with POMO and ReLD on problems with 100-1,000 customers show consistent improvements over static training. For clustered demand with Poisson arrivals at 1,000 customers, NPR reduces ReLD's gap to ALNS from 40.77% to 3.92%. On 122 public DVRP instances, NPR achieves a mean gap of 1.78%. Further experiments demonstrate compatibility with pairwise preference training, transfer across spatial and temporal distributions and degrees of dynamism, and applicability to customer time windows.

1 INTRODUCTION

Dynamic vehicle routing supports same-day delivery and on-demand transportation, where requests arrive while vehicles are operating (Pillac et al., 2013; Zhang & Van Woensel, 2023). Unlike a static vehicle routing problem (VRP), a dynamic vehicle routing problem (DVRP) couples route construction with online decisions: a plan suited to visible requests may leave the fleet poorly positioned for later arrivals. A solver must balance route quality with the computation needed to react.

Periodic reoptimization divides the working day into decision epochs and solves a residual routing problem at each epoch. More frequent epochs allow faster reactions but require more computation. Metaheuristics provide strong solutions through repeated search (Okulewicz & Mańdziuk, 2019; Wang et al., 2024). Learning-enhanced methods predict useful structure or dispatching decisions, but often retain an external optimizer in the online loop (Xiang et al., 2022; Baty et al., 2024).

Neural combinatorial optimization (NCO) provides a complementary opportunity. Autoregressive neural solvers construct complete routes with a sequence of inexpensive forward passes, and methods such as POMO and ReLD offer strong inference efficiency and increasingly robust size generalization (Kwon et al., 2020; Huang et al., 2025; Zhou et al., 2025). These advances have been developed primarily for static problems. Their speed makes repeated neural reoptimization appealing, but dynamic deployment changes the inputs on which the solver operates.

Two technical obstacles prevent a direct transfer from static to dynamic routing. First, the residual problem observed at a later epoch is not an independent sample from the original static distribution. The revealed customer set is determined exogenously by request arrivals, whereas its partition into committed prefixes and free customers, vehicle assignments, timing context, and resulting spatial organization depend on earlier selected plans and the prescribed execution protocol. Training only on independently generated static instances therefore creates a mismatch between training inputs and the rollout-induced states encountered during deployment. Second, dispatched vehicles impose

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.

immutable route prefixes. A static neural solver is free to reconstruct every route and may consequently reorder customers whose dispatch decisions are already fixed. The decoder must therefore preserve committed route order while respecting capacity and time-window constraints.

We address both obstacles with *Neural Periodic Reoptimization* (NPR), a framework for adapting autoregressive neural routing solvers to periodic dynamic execution. Under a prescribed dispatch and commitment protocol, NPR encodes execution histories as residual routing inputs and trains the policy on states visited along its protocol-constrained reoptimization rollouts. This residual representation retains committed route prefixes and their timing as fixed context while exposing uncommitted customers for replanning. Commitment-Consistent Decoding (CCD) preserves the inherited prefixes by intersecting a commitment-admissible mask with the backbone's native feasibility mask. The execution protocol determines which decisions become immutable; NPR learns to construct completion plans under those commitments. Its training uses an epoch-local completion-cost objective on these rollout-induced states.

Across four DVRP sizes and 122 public benchmark instances, NPR consistently improves POMO and ReLD over static training under the same inference pipeline and budget. For ReLD, it reduces the gap to ALNS from 40.77% to 3.92% on 1,000-customer instances with clustered locations and Poisson arrivals, and from 12.52% to 1.78% on the public benchmarks. Controlled experiments with POMO show that the gains persist under both REINFORCE and pairwise preference training. Further evaluations show that the training benefits transfer across separate spatial and temporal distribution shifts and changes in the dynamic-request proportion without retraining. An extension to customer time windows also improves both backbones across all four tested sizes.

Our contributions are: (1) We propose NPR, a framework that adapts neural routing solvers to periodic dynamic execution by training on residual states visited under the current policy and prescribed execution protocol. This procedure exposes the solver to deployment states jointly shaped by request arrivals, earlier selected plans, and execution commitments; (2) We develop a residual-state interface and Commitment-Consistent Decoding (CCD) to carry execution commitments into neural construction. The representation retains historical prefixes and timing, while CCD preserves their order and route boundaries alongside the backbone's native feasibility constraints; (3) We evaluate NPR with two neural backbones and two training objectives. Comparisons under shared execution and decoding mechanisms demonstrate the benefit of residual-state training over static training, with transfer across problem sizes, demand distributions, and degrees of dynamism, and an extension to customer time windows.

2 Related Work

Dynamic vehicle routing. Dynamic routing methods repeatedly update plans while respecting vehicle execution and newly revealed requests (Pillac et al., 2013; Zhang & Van Woensel, 2023). ContDVRP combines time-sliced execution with population-based optimization (Okulewicz & Mańdziuk, 2019). Other metaheuristics adapt genetic search to customer and road-condition changes between working days (Li et al., 2024), support diversion under dynamic traffic (Sze et al., 2024), or combine removal and insertion operators for multi-depot DVRPTW (Wang et al., 2024). These methods provide strong reference solutions, but repeated search can become expensive at large scales. NPR retains periodic execution while replacing online iterative search with neural construction.

Learning-enhanced dynamic routing. PPL-ACO learns pairwise proximity to guide ant-colony search after a change (Xiang et al., 2022). ML-CO learns dispatching decisions while delegating routing to a combinatorial optimization layer (Baty et al., 2024). These approaches use learning to support a downstream optimizer. NPR instead trains the route-construction policy on dynamic residual states and preserves execution commitments inside its decoder.

Neural routing and training objectives. POMO uses multiple rollouts and a shared REINFORCE baseline (Kwon et al., 2020); ReLD improves size generalization by strengthening a light decoder (Huang et al., 2025). Other static routing methods address cross-distribution and cross-scale transfer (Li et al., 2026; Chen et al., 2025). PIP masks decisions that risk future constraint violations (Bi et al., 2024). BOPO introduces objective-guided pairwise preference training, and UCPO adds

constraint-aware preference objectives (Liao et al., 2025; Fang et al., 2026). NPR addresses a complementary issue: the training inputs and immutable decisions created by dynamic deployment. We evaluate it with both REINFORCE and the objective-value preference component of BOPO/UCPO, retaining the same POMO architecture for this comparison.

3 Problem Definition

3.1 Dynamic Routing Problem

We study a single-depot capacitated DVRP in which customer requests are revealed during vehicle operations. For a realized instance, let $G = (V, E)$ be a complete graph with vertex set $V = \{0\} \cup V_c$, where 0 denotes the depot and $V_c = \{1, \ldots, n\}$ is the set of customers. Each request i is revealed at its release time r_i, together with its location $x_i \in \mathbb{R}^2$, demand q_i, and service duration s_i. Travel time and cost between vertices i and j both equal their Euclidean distances d_{ij}, under unit travel speed. An unlimited fleet of identical vehicles with capacity Q is available at the depot. Each customer must be served exactly once, the total demand assigned to each route must not exceed Q, and every dispatched vehicle must return to the depot by the end of the working day H.

The policy is non-anticipative, making decisions based only on the revealed request history and the current execution state. In the extension with time windows (DVRPTW), each request additionally reveals a service window $[a_i, b_i]$. Service must start within this window, with waiting permitted for vehicles arriving before a_i. All vehicles must still return to the depot by H. For DVRP, we set $a_i = 0$ and $b_i = H$. The release time determines when a request becomes observable, whereas the service window constrains when service may begin.

The objective is to minimize the expected total distance traveled during execution. For a complete route $\rho = (0, u_1, \ldots, u_\ell, 0)$, define

$$\mathrm{cost}(\rho) = d_{0,u_1} + \sum_{j=1}^{\ell-1} d_{u_j,u_{j+1}} + d_{u_\ell,0}, \qquad L(z) = \sum_{\rho \in z} \mathrm{cost}(\rho), \tag{1}$$

where z is a collection of complete routes. Let $\xi \sim \mathcal{D}$ denote a dynamic instance, and let $z^{\mathrm{exec}}(\pi_\theta; \xi)$ denote the routes executed by policy π_θ on ξ under the prescribed execution protocol. The deployment objective is

$$\min_\theta \; J(\theta), \qquad J(\theta) = \mathbb{E}_{\xi \sim \mathcal{D}} \left[L(z^{\mathrm{exec}}(\pi_\theta; \xi)) \right]. \tag{2}$$

The objective measures the total distance actually traveled, without accumulating the costs of tentative plans generated during replanning. Appendix A formalizes the routing, timing, information, and commitment constraints.

3.2 Periodic Reoptimization and Execution Commitments

We use a prescribed periodic execution protocol to translate tentative routing plans into vehicle actions. The working day is divided into K intervals of length $\Delta = H/K$. Replanning occurs at epochs $t_k = k\Delta$ for $k = 0, \ldots, K-1$, and $t_K = H$ is the execution deadline. All requests are revealed by a known cutoff time $T_{\mathrm{co}} \le t_{K-1} < H$, ensuring that every request is observable by the final planning epoch. For $k = 0, \ldots, K-1$, let A_k denote the requests revealed by t_k and N_{k+1} those newly revealed in $(t_k, t_{k+1}]$:

$$A_k = \{i \in V_c : r_i \le t_k\}, \qquad N_{k+1} = \{i \in V_c : t_k < r_i \le t_{k+1}\}. \tag{3}$$

Requests arriving between planning epochs become available for replanning at the next epoch.

Replanning cannot remove, reassign, or reorder customers within a committed route prefix. A committed customer may still be awaiting service. Revealed, unserved customers outside all committed prefixes are termed *free customers* and may be reassigned and reordered within the uncommitted portions of routes, subject to the capacity and time constraints. After completing its committed prefix, a vehicle may wait at the last customer location for further instructions. Tentative return edges remain revisable, but every executed route must ultimately end at the depot.

A route with no existing commitments becomes eligible for dispatch when its projected return time falls within the final $T_{\mathrm{ac}}\Delta$ time units of the working day, where T_{ac} denotes the advanced-commit

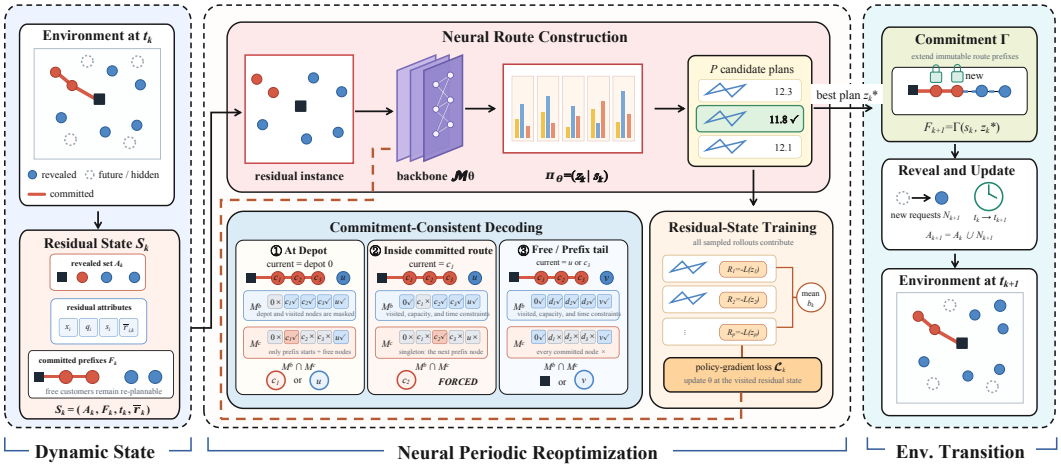

Figure 1: Neural Periodic Reoptimization (NPR). At each decision epoch, the execution history is encoded as a residual routing input, and commitment-consistent decoding (CCD) preserves inherited route prefixes during neural route construction. Executing the retained plan under the prescribed protocol and incorporating newly revealed requests produces the next residual state. Training uses these rollout-induced states shown in the training panel; model parameters remain fixed at inference.

horizon in epochs. Routes with committed prefixes remain eligible. For each eligible route, the commitment operator Γ preserves the existing prefix and extends it through each customer toward whom travel begins before the next planning epoch. In DVRPTW, additional prefixes are committed in advance when delaying departure until the next epoch risks violating a customer's time window or the depot-return deadline.

When replanning terminates, vehicles complete the remaining routes according to the retained feasible plan, termed the *incumbent*. Appendix A.5 details incumbent updates after full request revelation, early termination conditions, and terminal execution.

4 Neural Periodic Reoptimization

Neural Periodic Reoptimization (NPR) adapts autoregressive neural routing solvers to the periodic execution protocol defined in Section 3. It combines two mechanisms: training on residual states encountered during policy rollouts under this protocol, and commitment-consistent decoding (CCD) that preserves inherited route prefixes at each reoptimization epoch. Both mechanisms use a shared residual-state interface that encodes the revealed history, fixed route prefixes, and epoch-dependent timing for the neural solver. At inference time, NPR uses the same interface and decoding constraints with fixed model parameters.

Figure 1 illustrates how residual-state inputs, neural route construction, and execution interact within the reoptimization loop. Sections 4.1–4.3 describe the residual-state representation, CCD, and training procedure.

4.1 Residual-State Interface

At epoch t_k, the residual-state interface maps the revealed execution history h_k to a routing input $I_k = \Phi(h_k)$, combining the revealed request attributes with the dynamic state $S_k = (A_k, F_k, t_k, \bar{\boldsymbol{r}}_k)$. Here, F_k contains the committed prefixes supplied by the execution protocol. Writing $C(c)$ for the customer set of prefix c, the free customers are $U_k = A_k \setminus \bigcup_{c \in F_k} C(c)$. They may be assigned and ordered after existing prefixes or on new depot-to-depot routes, while each prefix remains at the start of its assigned vehicle's route. We use a complete-route representation that retains committed customers, including those already served, to record execution history and

account for demand already assigned to each vehicle. Retaining these customers does not entail serving them again; unrevealed requests are excluded from the input.

Auxiliary departure-time lower bounds $\bar{\boldsymbol{r}}_k$ encode the timing constraints inherited from execution. At each epoch, we set $\bar{r}_{i,k} = t_k$ for free customers and retain its current value once customer i becomes committed. Thus, r_i determines when a request becomes visible, whereas $\bar{r}_{i,k}$ specifies the earliest permitted departure toward customer i. For a route $\rho = (0, u_1, \ldots, u_\ell, 0)$, set $u_0 = 0$, $T_0 = 0$, and $s_0 = 0$. The earliest compatible service-start times are reconstructed as

$$T_j = \max\left\{\max\{T_{j-1} + s_{u_{j-1}}, \bar{r}_{u_j,k}\} + d_{u_{j-1},u_j},\ a_{u_j}\right\}. \tag{4}$$

Under the prescribed execution protocol, fixed prefixes and frozen bounds preserve the inherited prefix schedule. For free customers, the updated bounds ensure that newly planned departures occur no earlier than t_k.

The feasible completion set $\mathcal{Z}(I_k)$ consists of plans that cover every customer in A_k exactly once, preserve each assigned prefix, and satisfy vehicle capacity, service-window, and depot-return constraints. Specifically, each route must satisfy $\sum_{j=1}^{\ell} q_{u_j} \le Q$, $T_j \le b_{u_j}$ for $j = 1, \ldots, \ell$, and $T_\ell + s_{u_\ell} + d_{u_\ell,0} \le H$, with service-start times given by Equation (4). The neural solver generates candidate completion plans to approximately solve

$$z_k^\star \in \arg\min_{z \in \mathcal{Z}(I_k)} L(z). \tag{5}$$

Because every feasible plan retains the same committed prefixes, their travel cost is constant across candidates and does not affect candidate ranking. The resulting epoch-local objective evaluates complete routes over the currently revealed requests and serves as a surrogate for the online objective in Equation (2).

Executing the retained plan under the prescribed protocol and incorporating newly revealed requests yields the next residual state, as formalized in Appendix A.4. These successive states form the rollout-induced training inputs used in Section 4.3.

4.2 Commitment-Consistent Decoding

Commitment-Consistent Decoding (CCD) preserves each committed prefix $c = (c_1, \ldots, c_m) \in F_k$ at the start of its assigned route, while allowing free customers to be appended after the prefix. Define the set of prefix entries as $\mathrm{entry}(F_k) = \{c_1 : c \in F_k\}$. A prefix can be entered only from the depot, after which its remaining customers must be selected consecutively. At decoding step j, a controller tracks the next required customer in the active prefix, or remains idle if no prefix is active. The commitment-admissible next-node set is

$$M_{k,j}^{\mathrm{commit}} = \begin{cases} \{\mathrm{next}(c)\}, & \text{inside an active prefix } c, \\ U_k \cup \mathrm{entry}(F_k), & \text{idle at the depot}, \\ \{0\} \cup U_k, & \text{idle away from the depot}, \end{cases} \tag{6}$$

where $\mathrm{next}(c)$ denotes the next customer to be selected in the active prefix. Selecting c_1 activates the prefix; selecting c_m returns the controller to idle. Thus, a depot token must separate any two committed prefixes in the decoded sequence.

CCD combines this structural restriction with the feasibility constraints:

$$M_{k,j} = M_{k,j}^{\mathrm{base}} \cap M_{k,j}^{\mathrm{commit}}, \tag{7}$$

where $M_{k,j}^{\mathrm{base}}$ excludes unavailable or already selected customers and enforces capacity and timing constraints, including customer time windows and the depot-return deadline. Logits outside $M_{k,j}$ are set to $-\infty$ before the softmax. Provided that an inherited prefix remains feasible under the reconstructed load and schedule, its forced successors satisfy the base constraints. CCD preserves the prefix at the start of its assigned route without interruption and requires a depot token between distinct prefixes. Any free customers appended after a prefix remain subject to the capacity and timing constraints.

CCD requires no additional trainable parameters or network evaluations and integrates through the next-node masking interface. Beyond the prefix representation shared across rollouts, the controller stores one integer pointer per rollout. Constructing and intersecting the commitment mask requires $O(|A_k|)$ work per decoding step per rollout. For backbones that already perform linear-time feasibility masking over the residual nodes, CCD does not increase the asymptotic decoding complexity.

Algorithm 1 Dynamic residual-state training

Require: Policy π_θ, dynamic-instance distribution $\mathcal{D}$
Require: Horizon H, epochs K, rollouts P
while the training budget is not exhausted **do**
  Sample a dynamic instance from $\mathcal{D}$
  Initialize history h_0 and state S_0 at t_0
  for $k = 0, \ldots, K-1$ **do**
    Construct residual input $I_k = \Phi(h_k)$
    Sample P candidate plans $\{z_{k,p}\}_{p=1}^{P}$ using π_θ with CCD
    Update θ using all sampled plans (Equation (8) by default)
    Select the shortest feasible candidate $z_k^\star$
    $\hat{z}_k \leftarrow z_k^\star$
    if the termination condition is met **then**
      Exit the replanning loop
    end if
    Execute $\hat{z}_k$ under the prescribed protocol until t_{k+1}
    Update h_{k+1} and construct S_{k+1} using Equation (22)
  end for
  Complete the remaining execution according to $\hat{z}_k$
end while

4.3 Training on Rollout-Induced Residual States

NPR trains the solver on residual states generated along its own reoptimization trajectories (Algorithm 1). Unlike static training on independently sampled complete instances, NPR samples dynamic instances from $\mathcal{D}$ and advances each instance through the prescribed execution protocol. At each epoch, the current policy generates candidate completion plans, and the protocol selects and retains a plan for execution. Subsequent training states therefore reflect both exogenous request arrivals and the consequences of earlier retained plans, including vehicle assignments, committed prefixes, and the partition of revealed customers into committed and free sets.

Our default objective is REINFORCE (Williams, 1992) with the shared multi-start baseline of POMO (Kwon et al., 2020). At residual state S_k, the solver samples P candidate completion plans $\{z_{k,p}\}_{p=1}^{P}$ using CCD. Each plan receives the epoch-local reward $R_{k,p} = -L(z_{k,p})$, with shared baseline $b_k = P^{-1} \sum_{p=1}^{P} R_{k,p}$. The policy is updated using

$$\mathcal{L}_k(\theta) = -\frac{1}{P} \sum_{p=1}^{P} (R_{k,p} - b_k) \log \pi_\theta(z_{k,p} \mid S_k). \tag{8}$$

Rewards and the baseline are treated as fixed targets during differentiation. All sampled plans contribute to the update, whereas only the shortest feasible candidate is proposed to the execution protocol. Let $\hat{z}_k$ denote the plan retained by the execution protocol, including incumbent comparison when applicable. Executing $\hat{z}_k$ and incorporating newly revealed requests yields the next residual state.

The update uses the epoch-local completion cost rather than a return accumulated over the full dynamic trajectory. Thus, Equation (8) does not directly optimize the online objective $J(\theta)$ through credit assignment across reoptimization epochs. NPR instead exposes the solver to the consequences of its earlier retained plans by training on the resulting residual states. This separates the residual-state training procedure from the choice of local loss and preserves compatibility with standard POMO-style training. We also evaluate objective-guided pairwise preference training, defined in Appendix E, while keeping the candidate-generation and state-transition procedures unchanged.

Backbone integration. NPR connects to autoregressive neural construction solvers through the residual-state inputs and next-node feasibility masks. We instantiate the framework with POMO and ReLD (Kwon et al., 2020; Huang et al., 2025), and evaluate pairwise preference training on POMO. At each epoch, customer features comprise coordinates, demand, the auxiliary departure-time lower bound, and service duration; DVRPTW additionally includes customer time-window bounds. The

decoder context contains the remaining vehicle capacity, current route time, and accumulated route length. CCD is integrated by intersecting its commitment mask with the backbone's native feasibility mask. Integration requires adapting the input and context projections to encode the residual state, without introducing backbone-specific attention modules or additional reward terms.

5 EXPERIMENTS

We evaluate residual-state training on synthetic and public DVRP benchmarks, test its effectiveness with customer time windows (DVRPTW), and examine its compatibility with an alternative training objective. Additional experiments in Appendix F assess transfer across spatial and temporal distributions and degrees of dynamism using unchanged checkpoints. Data-generation and implementation details appear in Appendices B and C.

Datasets. We evaluate synthetic DVRP instances with $n \in \{100, 200, 500, 1000\}$ under Uniform–Uniform (U–U) and Cluster–Poisson (C–P) distributions, denoting customer locations and request arrivals, respectively. Each setting contains 1,000 instances for $n = 100$ and 64 for larger sizes. Public benchmarks include 22 classical instances from Kilby et al. (1998) and 100 CVRPLIB Set-X instances (Uchoa et al., 2017), with release times generated following ContDVRP (Okulewicz & Mańdziuk, 2019). See Appendices B.3 and C for further details.

Baselines. We compare against metaheuristics ALNS (Wang et al., 2024) and ContDVRP (Okulewicz & Mańdziuk, 2019), the learning-guided metaheuristic PPL-ACO (Xiang et al., 2022), and neural construction solvers POMO (Kwon et al., 2020) and ReLD (Huang et al., 2025). For each backbone, static and NPR training share the same inference pipeline, including CCD, and the same inference budget. DVRPTW comparisons exclude ContDVRP and PPL-ACO because their evaluated implementations do not support time windows. See Appendix B.2 for details.

Training and inference. DVRP models are trained on U–U instances with $n = 100$ and evaluated without retraining. All neural models are trained from scratch for 450,000 updates using REINFORCE unless otherwise specified. Neural inference uses up to 100 greedy starts per decision epoch on a single NVIDIA RTX 4090; see Appendix B for details.

Evaluation and metrics. All methods use $K = 40$ decision epochs, a request cutoff at $0.5H$, and $T_{\text{ac}} = 8$ epochs ($0.2H$). Neural methods report the minimum final distance over eight-augmentation trajectories, while ALNS and ContDVRP use the minimum over eight parallel runs. We report test-set mean distances, gaps to ALNS, and total runtime; ALNS serves as a reference rather than an optimality certificate.

5.1 RESULTS ON SYNTHETIC DVRP INSTANCES

Table 1: Results on U–U DVRP instances. Obj. denotes the average per-instance minimum distance; Gap % is relative to ALNS (lower is better); Time is total time. Bold indicates the best non-ALNS objective value at each problem size.

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
Method	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time
ALNS	17.91	0.00	19.3h	25.22	0.00	8.0h	52.92	0.00	29.8h	97.35	0.00	340.7h
ContDVRP	17.76	-0.84	16.8h	28.64	13.56	6.3h	67.21	27.00	16.7h	133.71	37.35	65.2h
PPL-ACO	20.11	12.28	3.7h	30.01	18.99	32.0m	59.91	13.21	18.8h	113.53	16.62	200.0h
POMO	21.92	22.39	2.1m	33.76	33.86	0.8m	78.77	48.85	4.5m	187.76	92.87	18.7m
NPR (POMO)	**17.33**	**-3.24**	1.9m	26.10	3.49	1.0m	59.46	12.36	4.6m	121.08	24.38	18.7m
ReLD	20.65	15.30	2.6m	30.83	22.24	1.1m	64.67	22.20	5.5m	128.60	32.10	21.3m
NPR (ReLD)	17.34	-3.18	2.3m	**25.65**	**1.70**	1.0m	**54.64**	**3.25**	5.0m	**103.42**	**6.24**	18.8m

We evaluate whether the benefits of residual-state training extend across problem sizes and demand distributions, using POMO and ReLD trained on U–U instances with $n = 100$ and tested without

Table 2: Results on C–P DVRP instances. Obj. denotes the average per-instance minimum distance; Gap % is relative to ALNS (lower is better); Time is total time. Bold indicates the best non-ALNS objective value at each problem size.

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
Method	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time
ALNS	16.14	0.00	16.8h	21.63	0.00	4.2h	48.74	0.00	92.7h	92.88	0.00	348.6h
ContDVRP	**15.08**	**-6.57**	12.0h	25.30	16.97	3.6h	62.34	27.90	16.2h	126.42	36.11	66.1h
PPL-ACO	17.49	8.36	5.1h	25.17	16.37	1.3h	54.60	12.02	22.6h	104.87	12.91	246.5h
POMO	20.04	24.16	2.0m	29.32	35.55	0.8m	76.50	56.96	4.6m	166.50	79.26	18.8m
NPR (POMO)	15.83	-1.92	1.8m	22.91	5.92	0.9m	55.53	13.93	4.2m	114.22	22.98	17.9m
ReLD	18.29	13.32	2.8m	25.93	19.88	1.0m	60.17	23.45	5.4m	130.75	40.77	21.7m
NPR (ReLD)	15.86	-1.73	2.5m	**22.37**	**3.42**	0.9m	**50.46**	**3.53**	4.8m	**96.52**	**3.92**	18.3m

retraining on U–U and C–P instances with up to 1000 customers. NPR consistently improves both POMO and ReLD over static training across all tested problem sizes and both demand distributions, using the same inference pipeline and budget (Tables 1 and 2). On U–U instances with $n = 1000$, NPR reduces the gap to ALNS from 92.87% to 24.38% for POMO and from 32.10% to 6.24% for ReLD. Thus, residual-state training benefits both backbones at larger scales, although ReLD retains a substantially smaller gap.

The gains also persist when both customer locations and request arrivals shift from the U–U training distribution to C–P. At $n = 1000$, NPR reduces ReLD's average route distance by 26.18% relative to static training, leaving a 3.92% gap to ALNS. NPR-trained ReLD achieves the lowest distance among the evaluated non-ALNS methods at $n = 200$, 500, and 1000 under both distributions. Its advantage is not universal, however: ContDVRP achieves the lowest distance on C–P at $n = 100$.

5.2 Results on Public DVRP Benchmarks

Table 3: Mean instance-wise gap (%) to ALNS on 122 public DVRP instances. Size bins are disjoint; Overall is instance-weighted. Bold marks the lowest non-ALNS gap.

Method	$n \leq 200$	$n \leq 500$	$n \leq 1000$	Overall
ALNS	0.00	0.00	0.00	0.00
ContDVRP	12.76	12.79	11.85	12.54
PPL-ACO	5.83	9.29	12.05	8.79
POMO	17.04	32.24	39.40	28.76
NPR (POMO)	8.22	11.06	14.84	11.05
ReLD	8.27	13.60	16.64	12.52
NPR (ReLD)	**0.39**	**2.96**	**1.93**	**1.78**

To assess whether the gains from residual-state training extend beyond synthetic test distributions, we evaluate both backbones on 122 public DVRP instances without fine-tuning. As shown in Table 3, NPR reduces the mean instance-wise gap to ALNS from 28.76% to 11.05% for POMO and from 12.52% to 1.78% for ReLD. NPR-trained ReLD achieves the lowest gap among all evaluated non-ALNS methods in every size group, with group-wise mean gaps below 3%. These results support the transfer of NPR's training benefits to public benchmarks across both backbones.

5.3 Results on Dynamic Routing with Time Windows

We next examine whether residual-state training remains effective when dynamic routing also requires satisfying customer time windows. We evaluate synthetic DVRPTW instances at the same four problem sizes and with the same test-set sizes as in the DVRP experiments, comparing ALNS with static and NPR-trained POMO and ReLD. Each static/NPR pair uses the same deadline-triggered commitment rules, decoding masks, and inference budget, allowing us to assess the training benefit under a shared execution protocol. Request releases are conditioned on service deadlines,

and the generator differs from that used for DVRP; this experiment therefore assesses effectiveness under time-window constraints rather than relative task difficulty. Appendix C provides the full generation and execution details.

NPR reduces average route distance for both backbones at every tested size (Table 4). At $n = 1000$, it lowers POMO's gap to ALNS from 23.07% to 11.17% and ReLD's gap from 3.93% to -0.50%. NPR-trained ReLD achieves distances close to those of ALNS at both $n = 500$ and $n = 1000$, with respective gaps of 0.24% and -0.50%. These improvements are achieved with total test-set runtimes comparable to those of static ReLD. Together, these matched comparisons show that the benefit of residual-state training extends to the time-window setting without requiring a larger inference budget.

Table 4: Synthetic DVRPTW results. Obj. denotes the average per-instance minimum distance; Gap % is relative to ALNS (lower is better); Time is total time. Bold indicates the best non-ALNS objective value at each problem size.

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
Method	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time
ALNS	23.78	0.00	14.6h	40.14	0.00	3.7h	94.74	0.00	72.5h	169.45	0.00	140.6h
POMO	28.61	20.31	1.6m	45.13	12.43	0.5m	104.08	9.86	2.1m	208.54	23.07	8.2m
NPR (POMO)	**24.63**	**3.57**	1.5m	42.36	5.53	0.5m	98.97	4.46	2.2m	188.38	11.17	7.7m
ReLD	27.52	15.73	2.0m	44.92	11.91	0.6m	99.62	5.15	2.6m	176.11	3.93	9.6m
NPR (ReLD)	24.67	3.74	1.8m	**41.96**	**4.53**	0.5m	**94.97**	**0.24**	2.5m	**168.61**	**-0.50**	9.0m

5.4 Compatibility with Training Objectives

Table 5: Training-objective comparison with the POMO backbone on the same 122 public DVRP instances. Entries are mean gaps (%) to ALNS. Architecture, masks, sampling width, training updates, and inference budget are held fixed. Bold marks the lower gap within each objective.

Training objective	Regime	$n \leq 200$	$n \leq 500$	$n \leq 1000$	Overall
REINFORCE	Static	17.04	32.24	39.40	28.76
REINFORCE	NPR	**8.22**	**11.06**	**14.84**	**11.05**
Pairwise preference	Static	11.41	14.28	18.24	14.31
Pairwise preference	NPR	**3.63**	**8.36**	**9.11**	**6.89**

We examine whether the gains from residual-state training depend on the training objective. On the same POMO backbone, we compare REINFORCE with a shared multi-start baseline against objective-guided pairwise preference optimization, each under static and NPR training. The preference loss follows BOPO (Liao et al., 2025); its relation to UCPO (Fang et al., 2026) is detailed in Appendix E. All configurations are trained from scratch for 450,000 updates under the same protocol. NPR improves every size group under both objectives (Table 5). The overall gap to ALNS decreases from 28.76% to 11.05% with REINFORCE and from 14.31% to 6.89% with pairwise preference training, showing that residual-state training remains beneficial with a stronger training objective.

6 Conclusion

NPR adapts neural construction solvers to periodic dynamic routing by training on rollout-induced residual states and preserving immutable execution commitments through CCD. Across synthetic and public DVRP benchmarks, NPR consistently improves POMO and ReLD under matched inference protocols, achieving a 1.78% mean gap to ALNS on 122 public instances with ReLD, with additional gains under time windows and pairwise preference training. The current formulation assumes deterministic travel, homogeneous vehicles, and fixed decision epochs; extending NPR to stochastic travel, finite heterogeneous fleets, event-driven replanning, and long-horizon training objectives is promising future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with literature retrieval and discovery, brainstorming and refinement of research ideas, and selected aspects of research execution, including discussing methodological and implementation alternatives. Generative AI tools were also used to draft portions of the manuscript and to improve the clarity, grammar, and presentation of the writing.

All AI-assisted research ideas, methodological suggestions, implementation-related outputs, references, and generated text were reviewed and verified by the authors. References suggested by AI tools were checked against the original sources, and AI-assisted research or implementation suggestions were independently evaluated and validated through our own analysis and experiments. The authors made all final decisions regarding the research methodology, experimental design, analysis, interpretation of results, and conclusions.

We take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of our results. The complete formulation and methodological details of the proposed approach are provided in the main paper, with additional algorithmic and implementation details presented in the appendix. We report the experimental settings, model architectures, training procedures, hyperparameters, evaluation metrics, and baseline configurations used in our experiments. Detailed descriptions of the datasets and instance-generation procedures are also provided to facilitate independent reproduction. Additional experimental results and analyses are included in the appendix. We provide anonymous source code, configuration files, and scripts for training and evaluation in the supplementary materials. Unless otherwise specified, reported results are obtained using the experimental protocol and random seeds described in the appendix.

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.

Jieyi Bi, Yining Ma, Jianan Zhou, Wen Song, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Learning to handle complex constraints for vehicle routing problems. In *Advances in Neural Information Processing Systems*, volume 37, pp. 93479–93509, 2024. doi: 10.52202/079017-2964.

Yuanyao Chen, Rongsheng Chen, Fu Luo, and Zhenkun Wang. Improving generalization of neural combinatorial optimization for vehicle routing problems via test-time projection learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/6edd46d69ef91f4555d67f7b321d6902-Abstract-Conference.html.

Zhanhong Fang, Debing Wang, Jinbiao Chen, Jiahai Wang, and Zizhen Zhang. UCPO: A universal constrained combinatorial optimization method via preference optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36900–36908, 2026. doi: 10.1609/aaai.v40i43.41017.

Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/447ac93bf22099aa346a45577376492d-Abstract-Conference.html.

Philip Kilby, Patrick Prosser, and Paul Shaw. Dynamic VRPs: A study of scenarios. Technical Report APES-06-1998, University of Strathclyde, 1998. URL https://users.cecs.anu.edu.au/~pjk/pjkbib.htm.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

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. URL https://proceedings.neurips.cc/paper/2020/hash/f231f2107df69eab0a3862d50018a9b2-Abstract.html.

Jianxia Li, Ruochen Liu, and Ruinan Wang. Handling dynamic capacitated vehicle routing problems based on adaptive genetic algorithm with elastic strategy. *Swarm and Evolutionary Computation*, 86:101529, 2024. doi: 10.1016/j.swevo.2024.101529.

Ning Li, Peng Lin, Peng Zhang, and Ruichen Tian. Elite pattern reinforcement for vehicle routing problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36298–36306, 2026. doi: 10.1609/aaai.v40i43.40949.

Zijun Liao, Jinbiao Chen, Debing Wang, Zizhen Zhang, and Jiahai Wang. BOPO: Neural combinatorial optimization via best-anchored and objective-guided preference optimization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37456–37475, 2025. URL https://proceedings.mlr.press/v267/liao25a.html.

Michał Okulewicz and Jacek Mańdziuk. A metaheuristic approach to solve dynamic vehicle routing problem in continuous search space. *Swarm and Evolutionary Computation*, 48:44–61, 2019. doi: 10.1016/j.swevo.2019.03.008.

Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013. doi: 10.1016/j.ejor.2012.08.015.

Jeeu Fong Sze, Said Salhi, and Niaz Wassan. An adaptive variable neighbourhood search approach for the dynamic vehicle routing problem. *Computers & Operations Research*, 164:106531, 2024. doi: 10.1016/j.cor.2024.106531.

Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017. doi: 10.1016/j.ejor.2016.08.012.

Sihan Wang, Wei Sun, and Min Huang. An adaptive large neighborhood search for the multi-depot dynamic vehicle routing problem with time windows. *Computers & Industrial Engineering*, 191:110122, 2024. doi: 10.1016/j.cie.2024.110122.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Xiaoshu Xiang, Ye Tian, Xingyi Zhang, Jianhua Xiao, and Yaochu Jin. A pairwise proximity learning-based ant colony algorithm for dynamic vehicle routing problems. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5275–5286, 2022. doi: 10.1109/TITS.2021.3052834.

Jian Zhang and Tom Van Woensel. Dynamic vehicle routing with random requests: A literature review. *International Journal of Production Economics*, 256:108751, 2023. doi: 10.1016/j.ijpe.2022.108751.

Fangting Zhou, Attila Lischka, Balázs Kulcsár, Jiaming Wu, Morteza Haghir Chehreghani, and Gilbert Laporte. Learning for routing: A guided review of recent developments and future directions. *Transportation Research Part E: Logistics and Transportation Review*, 202:104278, 2025. doi: 10.1016/j.tre.2025.104278.

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Chi Xu. MVMoE: Multi-task vehicle routing solver with mixture-of-experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61804–61824, 2024. URL https://proceedings.mlr.press/v235/zhou24c.html.

A Formal Problem Formulation

This section formalizes the constraints in Section 3 using the same route representation as the residual subproblem in Section 4.1. We first define static CVRP feasibility, then introduce service timing, request visibility, and execution commitments. The capacity constraints follow standard neural CVRP formulations (Kool et al., 2019); the time-window extension follows formulations used in multi-variant neural routing (Zhou et al., 2024). The dynamic execution rules are those of the scheduling protocol in Section 3.2.

A.1 Routes and Capacity

For a realized instance, let $V = \{0\} \cup V_c$, where 0 is the depot and $V_c = \{1, \dots, n\}$ is the full customer set. A nonempty route is an ordered sequence $\rho = (0, u_1, \dots, u_\ell, 0)$ with distinct customer indices, and $C(\rho) = \{u_1, \dots, u_\ell\}$ denotes its customer set. A collection of routes z serves every customer exactly once and respects vehicle capacity if

$$\sum_{\rho \in z} \mathbf{1}\{i \in C(\rho)\} = 1, \qquad \forall i \in V_c, \tag{9}$$

$$\sum_{i \in C(\rho)} q_i \le Q, \qquad \forall \rho \in z. \tag{10}$$

Every route begins and ends at the depot by construction. The fleet is unlimited, so no additional bound on the number of routes is imposed. Let $\mathcal{Z}_{\mathrm{CVRP}}$ be the set of route collections satisfying these conditions. Using the distance function L in Equation (1), the static CVRP is

$$\min_{z \in \mathcal{Z}_{\mathrm{CVRP}}} L(z). \tag{11}$$

This static formulation assumes that all customers are known before planning. The working-day deadline and service durations used in our experiments introduce the additional timing constraints below.

A.2 Service Timing and Time Windows

For route ρ, let D_j^ρ be the departure time from u_{j-1} toward u_j, and let T_j^ρ be the service-start time at u_j. Set $u_0 = 0$, $T_0^\rho = 0$, and $s_0 = 0$. With unit travel speed, feasible timing satisfies

$$D_j^\rho \ge T_{j-1}^\rho + s_{u_{j-1}}, \qquad j = 1, \dots, \ell, \tag{12}$$

$$T_j^\rho \ge D_j^\rho + d_{u_{j-1}, u_j}, \qquad j = 1, \dots, \ell, \tag{13}$$

$$a_{u_j} \le T_j^\rho \le b_{u_j}, \qquad j = 1, \dots, \ell, \tag{14}$$

$$T_\ell^\rho + s_{u_\ell} + d_{u_\ell, 0} \le H. \tag{15}$$

Equation (12) allows waiting after service, including at a committed-prefix endpoint. Equations (13)–(14) allow early arrival followed by waiting until the service window opens. The upper bound b_i applies to service start, while Equation (15) includes service completion and the final trip to the depot. It ensures that a timely return remains possible without requiring immediate return after the last service.

For DVRP, we set $a_i = 0$ and $b_i = H$. DVRPTW additionally requires the customer-specific windows $[a_i, b_i]$ revealed with each request. Setting all release times to zero gives the corresponding static problems. In particular, retaining Equation (15) yields a static problem with a working-day deadline, whereas Equation (11) alone defines the capacity-constrained baseline.

A.3 Dynamic Information and Non-Anticipativity

The full customer set specifies a realized instance, but each request becomes observable only upon release. At time t, the policy observes requests in

$$A(t) = \{i \in V_c : r_i \le t\}, \qquad 0 \le r_i \le T_{\mathrm{req}} < H, \tag{16}$$

together with the execution history. A vehicle can start traveling toward a customer only after that customer's request has been revealed, so an executed route must also satisfy

$$D_j^\rho \geq r_{u_j}, \qquad j = 1, \ldots, \ell. \tag{17}$$

This departure constraint does not by itself ensure non-anticipativity, since an offline planner could still use future requests to choose earlier routes. Let $h_k(\xi)$ contain the revealed request history and execution state at epoch t_k for instance ξ. Non-anticipativity requires

$$h_k(\xi) = h_k(\xi') \implies \pi_\theta(\cdot \mid h_k(\xi)) = \pi_\theta(\cdot \mid h_k(\xi')). \tag{18}$$

Thus, indistinguishable histories induce the same distribution of routing decisions, even when future requests differ. For a deterministic policy, the decisions themselves coincide. In NPR, the encoded input $I_k = \Phi(h_k)$ and the scheduling protocol depend only on this observable history. Requests arriving between decision epochs become available for replanning at the next epoch, as specified in Section 3.

A.4 Committed Prefixes and Feasible Replanning

At epoch k, let $A_k = A(t_k)$ and let F_k contain the committed customer prefixes, each associated with its assigned vehicle. Write a prefix as $c = (c_1, \ldots, c_m)$, and let $C(c)$ be its customer set. The free customers are

$$U_k = A_k \setminus \bigcup_{c \in F_k} C(c). \tag{19}$$

Every revised plan must retain each prefix at the start of its assigned vehicle's route. If $\rho_{c,k} = (0, u_1, \ldots, u_\ell, 0)$ denotes that route, then

$$\ell \geq m, \qquad (u_1, \ldots, u_m) = (c_1, \ldots, c_m), \qquad \forall c \in F_k. \tag{20}$$

Consequently, no customer can be inserted into a committed prefix, and its customers cannot be removed, reassigned, or reordered. Free customers may be assigned and ordered after these prefixes or on new routes. Only the customer prefix is fixed by Equation (20), so a tentative return edge after its endpoint does not prevent later extensions. Once travel toward a customer begins, the prefix through that customer is committed. The protocol may also commit additional customers in advance. Historical departure and service events remain fixed, and a trip already in progress must be completed as dispatched.

For consistency with the decoder, a candidate plan represents complete routes over A_k, including committed customers already served. Their presence records execution history and does not require repeated service. Coverage is imposed by Equation (9) with V_c replaced by A_k, and unrevealed customers are excluded. Capacity is checked over each complete route, so demand already assigned to its prefix remains accounted for.

The auxiliary departure-time bounds $\bar{r}_{i,k}$ in Section 4.1 encode when a departure toward customer i may occur in the residual plan. They equal t_k for free customers and retain their previously assigned values for committed customers. For each candidate route, the earliest compatible schedule is reconstructed as

$$D_j^\rho = \max\{T_{j-1}^\rho + s_{u_{j-1}}, \bar{r}_{u_j,k}\}, \qquad T_j^\rho = \max\{D_j^\rho + d_{u_{j-1},u_j}, a_{u_j}\}, \tag{21}$$

which is Equation (4) with departure times made explicit. A vehicle whose prefix ended before t_k therefore waits until at least t_k before departing toward a newly appended customer. The original release times r_i determine visibility, while $\bar{r}_{i,k}$ preserves the timing of commitments and prevents newly planned departures from occurring in the past.

Under the prescribed execution protocol, the reconstructed schedule remains unchanged on each inherited prefix: its customer order, travel times, service durations, time windows, and auxiliary departure-time lower bounds are all fixed. Starting from the same depot initialization, induction along the prefix in Equation (21) therefore yields the same departure and service-start times. Updating the bounds of free customers affects only the uncommitted suffix.

The feasible completion set $\mathcal{Z}(I_k)$ consists of route collections covering A_k exactly once, preserving the assigned prefixes, and satisfying capacity, service windows, and depot return under Equation (21). The epoch optimization is Equation (5). Its feasible set depends on earlier execution

decisions, so successive subproblems are coupled rather than independent static instances. Instance-generation checks are described in Sections B.3 and C.1. These checks do not imply that every sequence of online decisions remains feasible.

The deployment objective remains Equation (2), evaluated on the routes ultimately executed under the scheduling protocol. If the deployed policy is stochastic, the expectation also includes its internal randomness. Historical travel contributes the same constant to all feasible completion plans at a given epoch, but summing those plan costs across epochs would count that travel repeatedly. We therefore distinguish the epoch-local completion objective from total executed distance.

Let $\hat{z}_k$ denote the plan retained for execution after candidate selection and, when applicable, incumbent comparison. When replanning continues, the execution protocol and newly revealed requests induce the transition

$$F_{k+1} = \Gamma(S_k, \hat{z}_k), \qquad A_{k+1} = A_k \cup N_{k+1},$$
$$\bar{r}_{i,k+1} = \begin{cases} \bar{r}_{i,k}, & i \in \bigcup_{c \in F_{k+1}} C(c), \\ t_{k+1}, & i \in A_{k+1} \setminus \bigcup_{c \in F_{k+1}} C(c). \end{cases} \tag{22}$$

Together with t_{k+1}, these quantities define S_{k+1}.

A.5 Dispatch and Execution Commitments

We use a prescribed execution protocol to translate tentative completion plans into vehicle actions. At each planning epoch, the routing solver proposes a plan for the revealed requests. The commitment operator Γ determines which parts of the retained plan become irrevocable before the next epoch. For a planned route ρ, let $\tau_{\mathrm{return}}(\rho)$ be its projected return time. The route is dispatch-eligible if it already begins with a previously committed prefix or satisfies the advanced-commit rule

$$\tau_{\mathrm{return}}(\rho) + T_{\mathrm{ac}}\Delta \geq H, \tag{23}$$

where T_{ac} is the advanced-commit horizon measured in epochs. Thus, a previously uncommitted route becomes eligible for dispatch when its projected return falls within the final $T_{\mathrm{ac}}\Delta$ time units of the working day. Routes with existing commitments remain eligible at subsequent epochs.

Let F_k denote the committed customer prefixes at epoch k. The operator Γ maps the current execution state and retained plan to F_{k+1}. For a dispatch-eligible route, it preserves its existing prefix and commits the next customer whenever travel toward that customer begins before t_{k+1}. Commitment is determined by departure, so a customer may become committed even if its service starts or finishes after the next epoch. Uncommitted suffixes, including tentative return edges, remain available for subsequent reoptimization. A vehicle that reaches the end of its committed prefix may wait there until a later dispatch decision; its planned return edge does not close the route.

DVRPTW additionally requires commitment before postponement makes a customer urgent. For each customer i in the current plan, we check

$$t_{k+1} + d_{0i} \geq b_i \quad \text{or} \quad t_{k+1} + 2d_{0i} + s_i \geq H. \tag{24}$$

If either condition holds, we commit the route prefix through and including i. Multiple triggers on one route extend the prefix to the last triggering customer. These checks trigger commitment when direct travel from the depot starting at the next epoch would reach or exceed the service-start or return-time threshold. These rules determine which decisions become immutable. Section 4.2 describes how the neural decoder preserves the resulting commitments during subsequent replanning.

Once all requests have been revealed, periodic reoptimization continues while retaining an incumbent feasible completion plan. A candidate replaces the incumbent only if it is feasible, respects existing execution commitments, and has a lower total distance; execution and subsequent commitments follow the retained incumbent. Reoptimization ends after the final planning epoch or earlier if the residual feasibility guard is triggered. This guard tests whether an uncommitted customer's updated departure bound would make direct service and return exceed H; triggering it stops further replanning rather than invalidating the retained execution plan. The remaining incumbent suffixes are executed to completion with return by H. Retaining the plan itself, rather than only its best distance, connects the reported objective to an executable solution.

B Additional Implementation Details

B.1 Training and Inference Configuration

Table 6 summarizes training and inference settings. All models start from random initialization and use Adam (Kingma & Ba, 2015). Static/NPR pairs retain identical architectures and masks, including backbone-specific normalization and logit clipping. Within each geometric-augmentation branch, inference selects the shortest feasible plan across greedy multi-start candidates at each epoch. The eight branches maintain separate dynamic trajectories, and the reported instance score is the minimum final distance across their completed trajectories. Validity masks exclude padded customers from encoder attention and decoder selection.

Table 6: Neural training and inference configurations. The multi-start width is reduced when fewer than 100 customers are available.

Configuration	Value
Training problem size	100
Number of training updates	450,000
Training batch size	120
Optimizer	Adam
Initial learning rate	1×10^{-4}
Learning-rate milestones	[350,000; 400,000]
Learning-rate decay factor	0.1
Number of encoder layers	6
Embedding dimension	128
Number of attention heads	8
Query/key dimension per head	16
Feed-forward hidden dimension	512
Maximum multi-start width	100
Test-time augmentation	8-fold
Training hardware	NVIDIA RTX 4090 GPU

B.2 Baseline Implementations and Evaluation

ALNS and PPL-ACO are our implementations of the methods described by Wang et al. (2024) and Xiang et al. (2022), respectively. ContDVRP is reproduced using the original authors' released source code and the parameter settings reported by Okulewicz & Mańdziuk (2019).

ALNS uses the `paper` preset and eight independent runs per instance, with eight parallel workers. The reported instance distance is the minimum final distance across these eight runs. ContDVRP likewise uses eight parallel runs and reports their minimum final distance. For neural methods, the eight candidates for final selection are the completed dynamic trajectories associated with the eight geometric augmentations, as described above. This best-of-eight aggregation is applied per instance before computing the dataset-level distance or gap.

B.3 Synthetic DVRP Data Generation

B.3.1 Customer Locations and Release Times.

For the Uniform–Uniform (U–U) distribution, customer locations are sampled independently from $[0, 1]^2$, and preliminary request release times are sampled uniformly over $[0, H]$. Following the one-day cut-off convention of Okulewicz & Mańdziuk (2019), the cut-off operation makes requests with preliminary times beyond $0.5H$ available at time zero; the remaining requests retain their release times. Thus, the uniform generator has a nominal dynamic fraction of 0.5, while individual random draws can vary. This differs from the exact-count DoD construction described below.

For the Cluster–Poisson (C–P) distribution, five cluster centers are sampled uniformly from $[0, 1]^2$. Every customer is assigned to a randomly selected cluster center and perturbed by isotropic Gaussian noise with standard deviation 0.1. Coordinates outside the unit square are clipped to its boundary.

Poisson request arrivals are generated through cumulative exponential interarrival times. The Poisson rate parameters are 10, 20, 50, and 100 for problem sizes 100, 200, 500, and 1,000, respectively. The same $0.5H$ cut-off operation is applied to the generated release times.

DVRP neural models are trained on U–U instances with training DoD set to 0.5. The U–P and C–U extensions combine the same spatial and temporal generators to separate their effects. All distribution extensions use 1,000 instances of size 100 and unchanged U–U-trained checkpoints. No distribution-specific fine-tuning is performed.

B.3.2 Demands, Capacity, and Working Day.

Customer demands are sampled uniformly from

$$q_i \in 1, 2, \ldots, 9. \tag{25}$$

Following the varying-capacity generator used by ReLD (Huang et al., 2025), based on CVRPLIB Set-X (Uchoa et al., 2017), an expected route size ν is sampled from

$$\nu \sim \text{Triangular}(3, 6, 25), \tag{26}$$

and vehicle capacity is defined as

$$Q = \left\lceil \nu \frac{\sum_{i=1}^{n} q_i}{n} \right\rceil. \tag{27}$$

Let d_{ij} denote the Euclidean distance between customers i and j. The working-day length is

$$H = 8 \max_{i,j \in V_c} d_{ij}. \tag{28}$$

The common customer service duration is the average pairwise customer distance:

$$s_i = \bar{d} = \frac{1}{n^2} \sum_{i \in V_c} \sum_{j \in V_c} d_{ij}, \qquad \forall i \in V_c. \tag{29}$$

An instance is regenerated if a customer fails the individual reachability condition

$$r_i + 2d_{0i} + s_i < H. \tag{30}$$

This condition ensures that each customer can be served by a dedicated depot–customer–depot route after its request is revealed. It is a necessary instance-level screening condition rather than a guarantee that every sequence of online routing decisions remains feasible.

The held-out synthetic test sets contain 1,000 instances for $n = 100$ and 64 instances for each of $n = 200$, $n = 500$, and $n = 1000$.

C Synthetic DVRPTW Protocol

C.1 Instance Generation

The DVRPTW evaluation uses synthetic instances with $n \in \{100, 200, 500, 1000\}$. Customer and depot coordinates are sampled uniformly from $[0, 1]^2$, and travel speed is one. The depot is open over $[0, H]$ with $H = 4.6$, and each customer has service duration $s_i = 0.2$. Raw demands are sampled uniformly from $\{1, \ldots, 9\}$ and normalized by vehicle capacity. Capacities are 50, 70, 100, and 150 for the four sizes, respectively. DVRPTW uses these fixed capacities rather than the DVRP capacity generator.

For each customer, let $d_i = d_{0i}$. We sample a time-window center and half-width independently as

$$c_i \sim \mathcal{U}(d_i, H - d_i - s_i), \qquad w_i \sim \mathcal{U}(s_i/2, H/3), \tag{31}$$

and set $a_i = \max\{0, c_i - w_i\}$ and $b_i = \min\{H, c_i + w_i\}$. To generate releases, define

$$u_i = \max\{0, \min\{b_i - d_i, H - s_i - 2d_i\}\}. \tag{32}$$

We draw $\tilde{r}_i \sim \mathcal{U}(u_i/2, u_i)$, shift the release back by one decision interval, $r'_i = \max\{0, \tilde{r}_i - H/40\}$, and apply the cut-off: $r_i = 0$ if $r'_i > 0.5H$, and $r_i = r'_i$ otherwise. Hence the temporal generator is

conditioned on individual service deadlines; it is not an independent uniform draw over the working day.

The generator accepts an instance only if, for every customer,

$$v_i = \max\{r_i + d_i, a_i\} \le b_i, \qquad v_i + s_i + d_i \le H, \tag{33}$$

up to a numerical tolerance of 10^{-5}. This checks direct service after release, including waiting for the time window. Test generation uses seed 12 and creates 1,000 instances at $n = 100$ and 64 at each larger size. Generated instances are stored and reused across methods.

C.2 Execution and Comparison

DVRPTW uses 40 epochs, an advanced-commit horizon of eight epochs, and a cut-off of $0.5H$. In addition to ordinary dispatch commitment, a customer triggers commitment if $t_{k+1} + d_{0i} \ge b_i$ or $t_{k+1} + 2d_{0i} + s_i \ge H$. The entire current route prefix through the last triggering customer is retained. This rule addresses the risk that waiting until the next epoch makes direct service too late. CCD preserves the resulting prefixes, while the backbone's native mask enforces capacity, customer windows, and depot return. Vehicles may wait at committed-prefix endpoints, and tentative return edges remain revisable. The periodic decision protocol is defined in Section 3, and the dispatch and terminal execution rules are specified in Appendix A.5; in particular, full request visibility does not itself end reoptimization.

We compare ALNS with static and NPR-trained POMO and ReLD. ContDVRP and PPL-ACO are excluded because their evaluated implementations do not support customer time windows. Each neural static/NPR pair shares the simulator, deadline-commitment rule, masks, multi-start width, and augmentation. Table 4 reports distance, gap to ALNS, and total test-set runtime for this synthetic evaluation.

D Public DVRP Benchmark Composition

The public DVRP benchmark contains the 22 classical Christofides, Fisher, and Taillard instances used by Kilby et al. and the 100 Uchoa Set-X instances (Kilby et al., 1998; Uchoa et al., 2017). Dynamic release times follow the ContDVRP construction (Okulewicz & Mańdziuk, 2019). The disjoint size groups contain 43 instances with $n \le 200$, 47 with $200 < n \le 500$, and 32 with $500 < n \le 1000$. U–U checkpoints are evaluated without fine-tuning.

For a group $\mathcal{I}$, the reported score is

$$\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} 100 \, \frac{L_{m,i} - L_{\mathrm{ALNS},i}}{L_{\mathrm{ALNS},i}}, \tag{34}$$

where $L_{m,i}$ is method m's total distance on instance i. Overall is calculated over all 122 instances, not by assigning equal weight to the three size groups.

E Training Objective Comparison

We cross REINFORCE and objective-guided pairwise preference with static and NPR training. All four POMO configurations use random initialization and 450,000 updates, with architecture, masks, rollout count, and inference budget fixed.

The default objective is REINFORCE with the mean multi-start return as its baseline, as defined in the main paper. The alternative uses the objective-value preference component shared by BOPO and UCPO's primal refinement loss (Liao et al., 2025; Fang et al., 2026). For sampled plans $z_1, \dots, z_P$ with positive distances $L_1, \dots, L_P$, let $p^\star \in \arg\min_p L_p$. Its pairwise form is

$$\mathcal{L}_{\mathrm{pref}}(\theta) = -\frac{1}{P-1} \sum_{p \ne p^\star} \log \sigma(\beta_p \left[\log \pi_\theta(z_{p^\star} \mid S) - \log \pi_\theta(z_p \mid S)\right]), \qquad \beta_p = \frac{L_p}{L_{p^\star}}, \tag{35}$$

where σ is the logistic sigmoid, and costs and pair selection are treated as fixed targets during differentiation. The best sampled plan anchors the preference pairs. We omit UCPO's feasibility-margin

and dual-exploration terms, and retain the same hard masks as in the REINFORCE comparison. The experiment therefore evaluates an alternative loss on POMO rather than the complete UCPO training framework. NPR uses the rollout-induced residual state visited under the current policy and execution protocol as S and proposes the best sampled plan to the same scheduling and incumbent-retention protocol under either objective.

F Distribution and Dynamism Extensions

These experiments assess whether the benefit of residual-state training transfers across spatial and temporal distributions and degrees of dynamism. All tests use unchanged U–U-trained checkpoints without retraining, with 1,000 instances at $n = 100$ per setting.

F.1 Separate Spatial and Temporal Shifts

The distribution experiment crosses uniform or clustered locations with uniform or Poisson request arrivals, giving U–U, U–P, C–U, and C–P. Each setting contains 1,000 instances with 100 customers. Every method retains its U–U-trained checkpoint, with no retraining or setting-specific fine-tuning. The data generators and cut-off are those specified for synthetic DVRP above.

F.2 Paired Degree-of-Dynamism Datasets

Following the request-count definition of degree of dynamism (Pillac et al., 2013; Okulewicz & Mańdziuk, 2019), we define $\mathrm{DoD} = |\{i : r_i > 0\}|/n$. Models retain the checkpoints from U–U training at nominal DoD 0.5. The sensitivity experiment uses 1,000 base U–U instances with $n = 100$ and sets the dynamic customer count to exactly 25, 50, or 75. Customer and depot locations, demands, service durations, capacities, and working-day lengths are unchanged across the three variants of each instance.

Using a random generator with seed 12, we sample a permutation of the 100 customer indices and assign independent release times in $(0, 0.5H]$ to its first 75 entries. The first 25, 50, or 75 entries form the dynamic set for the corresponding DoD. All other customers have release time zero. Dynamic sets are thus nested, and a customer that is dynamic in multiple variants retains the same release time. The regenerated releases mean that the DoD-0.5 set need not reproduce the main U–U table numerically.

The construction also checks direct reachability at the first decision epoch at which each customer becomes visible. With $\Delta = H/40$, let $t_i^{\mathrm{vis}} = \Delta\lceil r_i/\Delta \rceil$ for $r_i > 0$ and zero otherwise. Each instance must satisfy $t_i^{\mathrm{vis}} + 2d_{0i} + s_i \le H$ up to tolerance 10^{-6}. The three datasets are saved as fixed pickle files. A manifest records the source checksum, source indices, seed, customer permutation, release times, and dataset checksums; reuse validates these records. The cut-off $0.5H$ controls the latest release, independently of the dynamic-request proportion.

F.3 Results across Distributions and Degrees of Dynamism

Table 7: Distribution transfer at $n = 100$, with 1,000 instances per setting and the same U–U-trained checkpoint for each method. Metrics and boldface follow Table 1.

	U–U			U–P			C–U			C–P		
Method	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time
ALNS	17.91	0.00	19.3h	17.86	0.00	15.2h	16.30	0.00	15.0h	16.14	0.00	16.8h
POMO	21.92	22.39	2.1m	22.15	24.02	2.1m	19.78	21.35	1.6m	20.04	24.16	2.0m
NPR (POMO)	**17.33**	**-3.24**	1.9m	**17.21**	**-3.64**	1.8m	**16.01**	**-1.78**	1.4m	**15.83**	**-1.92**	1.8m
ReLD	20.65	15.30	2.6m	20.68	15.79	2.1m	18.46	13.25	2.1m	18.29	13.32	2.8m
NPR (ReLD)	17.34	-3.18	2.3m	17.22	-3.58	2.0m	16.03	-1.66	2.0m	15.86	-1.73	2.5m

Table 8: Degree-of-dynamism sensitivity on 1,000 paired U–U instances with $n = 100$ per setting. Checkpoints are unchanged from training at DoD 0.5. Metrics and boldface follow Table 1.

	DoD $= 0.25$			DoD $= 0.50$			DoD $= 0.75$		
Method	Obj.	Gap%	Time	Obj.	Gap%	Time	Obj.	Gap%	Time
ALNS	15.62	0.00	16.0h	18.02	0.00	14.7h	18.16	0.00	14.3h
POMO	20.39	30.54	2.2m	21.51	19.37	1.9m	22.64	24.67	1.7m
NPR (POMO)	**16.96**	**8.58**	2.1m	**17.34**	**-3.77**	1.6m	**17.55**	**-3.36**	1.4m
ReLD	20.37	30.41	2.5m	20.59	14.26	2.1m	20.69	13.93	1.7m
NPR (ReLD)	16.98	8.71	2.5m	17.35	-3.72	2.0m	17.60	-3.08	1.7m

Spatial and temporal shifts. The controlled distribution tests show that the training benefit persists when spatial and temporal shifts are introduced separately (Table 7). NPR improves both backbones on U–P, C–U, and C–P using unchanged U–U-trained checkpoints. The gains therefore extend across both arrival-process and location-distribution changes in the tested $n = 100$ settings. These tests complement the joint-shift results at larger sizes without establishing the same behavior for every size and distribution combination.

Degree of dynamism. The benefit also persists when the dynamic-request proportion changes from the training value of 0.5 to 0.25 or 0.75 (Table 8). The paired instances hold locations, demands, service durations, capacities, and the working day fixed; dynamic customer sets are nested, and shared dynamic customers retain the same release times. Both backbones improve with NPR at all three DoD values without retraining. Relative to ALNS, however, performance depends on the dynamic-request proportion: at DoD 0.25, NPR-trained POMO and ReLD retain positive gaps of 8.58% and 8.71%, respectively; at DoD 0.50 and 0.75, both achieve negative gaps, ranging from -3.77% to -3.08%. These results support robustness to the tested changes in dynamism at $n = 100$, but do not establish a monotonic relationship between DoD and NPR's advantage.