ALTERNATING EXPLORATION AND REPAIR VIA SPLIT SEARCH FOR HARD-CONSTRAINED ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hard-constrained routing seeks low-cost routes that satisfy operational restrictions, but reducing travel cost can conflict with restoring feasibility. We introduce **ALTER**, a neural solver that repeatedly alternates cost-oriented exploration and feasibility-oriented repair. Each selected route serves as the reference for the other role's next proposals. Accumulated disagreement carries their interaction across iterations, while a learned state-dependent coupling strength controls how strongly the two searches are coordinated. Our analysis characterizes coupling intervals under which finite-pool selection yields a feasible cost improvement. Experiments cover synthetic single- and multi-vehicle routing. On hard 100-customer draft-limit instances, ALTER produces a feasible output for 99.97% of instances and obtains the lowest gap among tested neural methods. It also obtains the lowest reported costs on the multi-vehicle benchmark.

1 INTRODUCTION

Hard-constrained routing requires finding low-cost routes that ultimately satisfy all operational constraints. The difficulty is that cost improvement and feasibility restoration often pull the search in different directions: reducing travel cost can introduce constraint violations, while repairing those violations can undo the cost reduction. Effective search must therefore balance two competing needs: exploring routes that reduce cost, potentially through infeasible regions, and recovering feasibility without discarding the improvements found along the way.

Neural routing solvers construct routes sequentially (Bello et al., 2017; Nazari et al., 2018; Kool et al., 2019) and can address feasibility through masking or backtracking (Bi et al., 2024; Li et al., 2026), or through jointly trained construction and refinement (Bi et al., 2026). Their common design, however, still tends to encode the cost–constraint balance through local feasibility control, hand-specified candidate scoring, or a prescribed construction–repair procedure. Rather than treating feasibility control as a fixed decoding rule or a prescribed post-construction repair stage, we ask a solver-level question: how should a constrained routing solver coordinate cost-oriented exploration and feasibility-oriented repair without limiting exploration beyond the repaired route?

We introduce **ALTER**, a neural solver that alternates cost-oriented exploration and feasibility-oriented repair. Starting from a complete route, exploration generates complete alternatives with greater emphasis on reducing travel cost. Repair then generates alternatives conditioned on the selected exploration route, with greater emphasis on reducing violations. Its selected route guides the next exploration step. Thus, unlike a construction followed by short-horizon refinement, both roles repeatedly generate routes and use the other's selected route as a reference. This is intended to let search pass through a shorter but infeasible route before seeking a better feasible one.

To coordinate these roles, ALTER couples the two searches through a route representation relative to the current reference route. Accumulated disagreement provides feedback across iterations, and a learned controller adjusts the strength of this coupling from the current search state. The coupling therefore compares routes, rather than simply increasing the penalty on operational violations. This allows the interaction between exploration and repair to evolve with the search state while retaining their distinct optimization roles. Section 3 gives the candidate-generation and selection procedure.

Alongside the solver, we analyze feasible improvement under finite candidate pools by characterizing coupling intervals that yield cost-improving feasible selections (Section 4). Routing constructions

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.

show how both weak and strong coupling can exclude a useful exploration–repair step, while the sampling analysis distinguishes generating an improvement from selecting it. Together, these results explain how coupling coordinates cost-oriented exploration with feasible repair.

We evaluate feasibility, cost, and runtime on synthetic single- and multi-vehicle routing benchmarks. On hard 100-customer draft-limit instances, where visit feasibility depends on load, ALTER produces a feasible output for 99.97% of instances and has a 2.75% optimality gap, the lowest among the tested neural methods. On TSPTW Medium, ALTER reports zero instance infeasibility and outperforms all neural baselines in optimality gap at both sizes (Table 1). In the 50- and 100-customer multi-vehicle experiments, ALTER also has the lowest reported costs among the tested neural methods (Table 2).

In summary, our main contributions are as follows:

- We propose **ALTER**, a neural solver that alternates cost-oriented exploration and feasibility-oriented repair, coordinated through accumulated disagreement and learned state-dependent coupling.
- We theoretically characterize when finite-pool selection yields feasible cost improvement by deriving coupling intervals for successful exploration–repair transitions.
- We empirically show that ALTER achieves state-of-the-art neural solution quality across diverse single- and multi-vehicle constrained routing settings.

2 Problem Setup and Related Work

Hard-constrained routing. Let I denote a routing instance and $\mathcal{T}_I$ the route space satisfying basic requirements such as depot visits and customer coverage. For $\tau \in \mathcal{T}_I$, let $C_I(\tau)$ be travel cost and $g_{I,j}(\tau)$ the residual of operational constraint j, with $\mathcal{J}_I$ indexing these constraints. The objective is

$$\begin{aligned} \min_{\tau \in \mathcal{T}_I} \quad & C_I(\tau) \\ \text{s.t.} \quad & g_{I,j}(\tau) \le 0, \qquad j \in \mathcal{J}_I. \end{aligned} \tag{1}$$

Thus, a route in $\mathcal{T}_I$ can be complete but operationally infeasible. We study the Traveling Salesman Problem with Time Windows (TSPTW) or Draft Limits (TSPDL), and the Capacitated Vehicle Routing Problem with Backhauls, route-length limits, and Time Windows (CVRPBLTW). Draft limits impose load-dependent visit restrictions; Appendix C defines all three tasks.

Neural construction and masking. An autoregressive routing policy builds a route by selecting successive nodes from the instance and partial-route state (Kool et al., 2019; Bengio et al., 2021). A feasibility mask removes disallowed next-node choices. For time-window routing, a locally admissible choice need not leave a feasible completion. Proactive Infeasibility Prevention (PIP) addresses this with look-ahead masking (Bi et al., 2024); LMask uses budgeted backtracking to revise earlier choices and masks when decoding reaches a dead end (Li et al., 2026). Both mechanisms act on partial-route decisions.

Construction and refinement. Learned improvement includes local rewriting (Chen & Tian, 2019) and feasible/infeasible k-opt search (Ma et al., 2023). Construct-and-Refine (CaR) generates candidate solutions and improves selected ones with a short-horizon refinement policy (Bi et al., 2026). Its constructor and refiner share an encoder and train jointly, with improved outputs supervising construction.

Variable splitting and coordination. Variable splitting separates objective components while linking their variables through an agreement condition. The alternating direction method of multipliers (ADMM) coordinates alternating subproblems through a quadratic disagreement penalty and multiplier feedback that carries disagreement across iterations (Boyd et al., 2011). Related decomposition is used in classical routing (Yao et al., 2019); RLQP learns to adapt ADMM parameters in a continuous quadratic-programming solver (Ichnowski et al., 2021). These are coordination precedents, not convergence guarantees for our discrete sampled search.

3 METHOD

For a fixed instance I with n customers, write $\mathcal{T} = \mathcal{T}_I$, $C = C_I$, and let $\mathcal{F} \subseteq \mathcal{T}$ be the feasible set in Eq. (1). A route here denotes a complete solution, including all vehicle routes in CVRPBLTW.

ALTER starts from an initial route and alternates two searches. *Exploration* proposes complete routes with greater emphasis on reducing travel cost, using the current repair route as a reference. *Repair* then proposes complete routes with greater emphasis on reducing violations, using the selected exploration route as its reference. The resulting repair route guides the next exploration step. This allows search to pass through a shorter but infeasible route before seeking a feasible alternative. Both roles compare candidates with their reference through a shared route representation; accumulated disagreement and a learned coupling strength coordinate the two searches. A separate incumbent stores the retained output, so search references need not replace it. After a fixed budget, ALTER returns the incumbent.

Figure 1 illustrates this coordination.

3.1 ALTERNATING PROPOSALS, SELECTION, AND MEMORY

At step k, the incoming repair reference is z_k, the accumulated disagreement is $\mathbf{u}_k$, and the coupling strength is $\rho_k \geq 0$. Both roles use the same ρ_k. Their states $\mathbf{s}_k^x$ and $\mathbf{s}_k^z$ encode $(I, z_k, \mathbf{u}_k, \rho_k)$ and $(I, x_{k+1}, \mathbf{u}_k, \rho_k)$, respectively. Here x_{k+1} is the exploration winner, so repair generation follows its selection. The policies generate complete routes as

$$\tau_x \sim \pi_{\boldsymbol{\theta}_x}(\cdot \mid \mathbf{s}_k^x), \qquad \tau_z \sim \pi_{\boldsymbol{\theta}_z}(\cdot \mid \mathbf{s}_k^z). \tag{2}$$

The parameter subscripts identify roles: both policies share node and reference-route encoders and a role-conditioned autoregressive decoder (Appendix D).

We sample G exploration candidates and select the one minimizing $\mathcal{J}_k^x$; conditioned on this winner, we sample G repair candidates and minimize $\mathcal{J}_k^z$. The scores are

$$\begin{aligned} \mathcal{J}_k^x(\tau) &= \tilde{C}(\tau) + \frac{\rho_k}{2}\|\mathbf{y}(\tau) - \mathbf{y}(z_k) + \mathbf{u}_k\|_2^2, \\ \mathcal{J}_k^z(\tau) &= \tilde{V}(\tau) + \frac{\rho_k}{2}\|\mathbf{y}(x_{k+1}) - \mathbf{y}(\tau) + \mathbf{u}_k\|_2^2. \end{aligned} \tag{3}$$

Reference conditioning influences which routes are proposed; these scores determine how sampled routes are ranked. Memory enters both the policy context and the scores. In representation space, the exploration and repair penalties are centered at $\mathbf{y}(z_k) - \mathbf{u}_k$ and $\mathbf{y}(x_{k+1}) + \mathbf{u}_k$, respectively. The scalar ρ_k weights inter-route agreement. $\mathbf{y}(\tau)$ is the shared representation of a complete route τ, while $\tilde{C}$ and $\tilde{V}$ are the cost-oriented exploration and feasibility-oriented repair objectives, respectively.

After selecting the repair route z_{k+1}, we update

$$\mathbf{r}_{k+1} = \mathbf{y}(x_{k+1}) - \mathbf{y}(z_{k+1}), \qquad \mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{r}_{k+1}. \tag{4}$$

The residual $\mathbf{r}_{k+1}$ records the current disagreement. This additive memory is not rescaled when ρ_k changes; its effective linear coefficient in the score is $\rho_k \mathbf{u}_k$. We therefore use *dual-inspired memory*, rather than a canonical scaled ADMM multiplier Boyd et al. (2011). Appendix B.3 explains this distinction in more detail. The corresponding ideal split formulation is given in Appendix B.4.

3.2 STATE-DEPENDENT PENALTY CONTROL

The penalty controls score pressure toward the memory-shifted targets; overly strong coupling can suppress useful exploration or repair candidates. We learn to choose the penalty from the current search observation:

$$\rho_k \sim p_{\boldsymbol{\psi}}(\cdot \mid \mathbf{s}_k^\rho), \qquad \rho_k \in [\rho_{\min}, \rho_{\max}]. \tag{5}$$

Here $\boldsymbol{\psi}$ denotes controller parameters. The observation $\mathbf{s}_k^\rho$ contains a pooled repair-reference encoding, including resource, violation, and memory features, plus the previous penalty and iteration index. Training samples a scaled Beta action; inference uses its conditional mean (Appendix D).

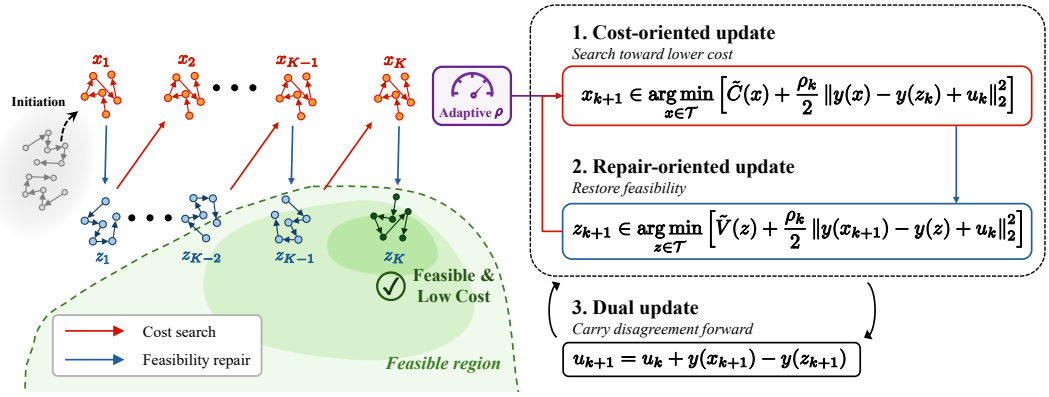


Figure 1: Alternating exploration and repair in ALTER. Each selected route becomes the other role's reference; memory carries their disagreement forward. The boxes show ideal update targets; the implemented solver selects each update from a sampled candidate pool.

The controller learns from the incumbent's progress over *both* role updates. Let g_k and g_{k+1} be the retained routes before and after step k, respectively. We measure reductions in travel cost C and aggregate violation magnitude V by

$$\Delta_k^C = C(g_k) - C(g_{k+1}), \qquad \Delta_k^V = V(g_k) - V(g_{k+1}). \tag{6}$$

The step reward is

$$R_\rho^k = \mathbb{1}\{\Delta_k^C > 0\} + \mathbb{1}\{\Delta_k^V > 0 \text{ or } g_k, g_{k+1} \in \mathcal{F}\} - 1. \tag{7}$$

It gives $+1$ when cost decreases and violations decrease or feasibility is preserved, -1 when neither condition holds, and zero otherwise.

3.3 Route Representation and Role Objectives

Let $v_i(\tau) \geq 0$ be the violation assigned to customer i, such as lateness in TSPTW or excess cumulative load in TSPDL. Define the violation magnitude $V(\tau) = \sum_{i=1}^n v_i(\tau)$, the violating-customer count $N_{\text{inf}}(\tau) = \sum_{i=1}^n \mathbb{1}\{v_i(\tau) > 0\}$, and their sum $\Phi(\tau) = V(\tau) + N_{\text{inf}}(\tau)$, where $\mathbb{1}$ is the indicator. Task-specific evaluation is described in Appendix C.

The two roles rank cost and violation differently:

$$\tilde{C}(\tau) = \alpha_C C(\tau) + \alpha_V \Phi(\tau), \qquad \tilde{V}(\tau) = \beta_C C(\tau) + \beta_V \Phi(\tau). \tag{8}$$

The coefficients are fixed across experiments with $\alpha_C \gg \alpha_V$ and $\beta_V \gg \beta_C$. Hence, $\tilde{C}$ is dominated by route cost and drives the x-update toward low-distance structure, whereas $\tilde{V}$ is dominated by the violation penalty and drives the z-update toward feasibility restoration. The smaller cross weights retain the complementary signal, keeping the two objectives comparable and preventing either update from ignoring the other side of the split.

To coordinate the roles, we compare routes through

$$\mathbf{y}(\tau) = [C(\tau), v_1(\tau), \ldots, v_n(\tau)]^\top \in \mathbb{R}^{n+1}. \tag{9}$$

3.4 Training and Inference

Algorithm 1 summarizes one inference chain. The lexicographic update $\text{Inc}(g, \tau)$ maintains the incumbent separately from score-based search references. The incumbent update prioritizes lower violation magnitude V, then lower cost C, retaining the current route on exact ties. As in the detailed algorithms, only selected role winners enter this update (Appendix B.2).

Algorithm 1 ALTER: one inference chain

Require: Instance I, trained policies, pool sizes G, horizon K, initial previous penalty ρ_{-1}
1: Initialize random depot-start route z_0; set $x_0 = g_0 = z_0$ and $\mathbf{u}_0 = \mathbf{0}$
2: **for** $k = 0, \dots, K-1$ **do**
3: Set $\rho_k = \mathbb{E}_{p_{\boldsymbol{\psi}}(\cdot|\mathbf{s}_k^\rho)}[\rho]$
4: Sample G routes $\mathcal{X}_k$ from $\pi_{\boldsymbol{\theta}_x}(\cdot \mid \mathbf{s}_k^x)$
5: Select $x_{k+1} \in \arg\min_{\tau \in \mathcal{X}_k} \mathcal{J}_k^x(\tau)$; set $g_{k+1/2} = \mathrm{Inc}(g_k, x_{k+1})$
6: Sample G routes $\mathcal{Z}_k$ from $\pi_{\boldsymbol{\theta}_z}(\cdot \mid \mathbf{s}_k^z)$, using x_{k+1} as reference
7: Select $z_{k+1} \in \arg\min_{\tau \in \mathcal{Z}_k} \mathcal{J}_k^z(\tau)$; set $g_{k+1} = \mathrm{Inc}(g_{k+1/2}, z_{k+1})$
8: Update $\mathbf{u}_{k+1}$ by Eq. (4)
9: **end for**
10: **return** g_K

Training uses the same rollout with sampled penalties. The route rewards are $R_x^k(\tau) = -\mathcal{J}_k^x(\tau)$ and $R_z^k(\tau) = -\mathcal{J}_k^z(\tau)$. Policy-gradient losses with component-specific baselines are accumulated as $\mathcal{L} = \lambda_x \mathcal{L}_x + \lambda_z \mathcal{L}_z + \lambda_\rho \mathcal{L}_\rho$, where $\lambda_x, \lambda_z, \lambda_\rho > 0$ weight the exploration, repair, and controller losses. Appendix B gives the losses and detailed training procedure.

4 Analysis of Feasible Improvement

Reference conditioning shapes the candidate pools; we ask which coupling values make their selection yield a better feasible route. For a feasible incumbent g, an infeasible exploration x, and a feasible repair f, define the saving $s = C(g) - C(x)$, repair premium $h = C(f) - C(x)$, and violation penalty $P = \Phi(x) > 0$. The feasible gain is $s - h$; usefulness requires $h < s$.

Why different role weights? Ignoring coupling, the scores prefer x to g and f to x, respectively, when

$$\frac{\alpha_V}{\alpha_C} < \frac{s}{P}, \qquad \frac{\beta_V}{\beta_C} > \frac{h}{P}. \tag{10}$$

For a family of such excursions, one fixed score $C + \lambda\Phi$ instead requires $\max_j h_j/P_j < \lambda < \min_j s_j/P_j$, which can be empty even when every excursion improves feasible cost. This motivates different ranking pressures; the derivation is in Appendix A.2.

When does coupling preserve the improvement? To isolate score-based selection, fix the incoming reference, memory, and candidate pools. Write the scores in Eq. (3) as $J_x^\rho = b_x + \rho q_x$ and $J_z^\rho(\cdot; x) = b_z + \rho q_z(\cdot; x)$, where $b_x = \tilde{C}$, $b_z = \tilde{V}$, and q_x, q_z are the half-squared disagreement terms. Let $\mathcal{G}_\Delta(g) = \{f \in \mathcal{F} : C(f) \le C(g) - \Delta\}$, for $\Delta > 0$.

Theorem 1 (Coupling intervals for feasible improvement)**.** Fix finite nonempty exploration and conditional repair pools $\mathcal{X}$ and $\mathcal{Z}(x)$. For $x \in \mathcal{X}$ and $f \in \mathcal{Z}(x) \cap \mathcal{G}_\Delta(g)$, define $\mathcal{I}(x, f) \subseteq [\rho_{\min}, \rho_{\max}]$ by

$$\begin{aligned} J_x^\rho(x) &< J_x^\rho(a) \quad \text{for all } a \in \mathcal{X} \setminus \{x\}, \\ J_z^\rho(f; x) &< J_z^\rho(b; x) \quad \text{for all } b \in \mathcal{Z}(x) \setminus \mathcal{G}_\Delta(g). \end{aligned} \tag{11}$$

For every $\rho \in \mathcal{I}(x, f)$, exploration selects x and every repair minimizer is a feasible improvement of at least Δ. If the incumbent retains the cheapest selected feasible route, its gain is at least Δ. Each $\mathcal{I}(x, f)$ is a computable interval, possibly empty.

Each comparison is $A_j + \rho B_j < 0$, with A_j, B_j the role-objective and disagreement-score differences, respectively, taken as desired candidate minus competitor. It imposes $\rho < -A_j/B_j$ for $B_j > 0$, $\rho > -A_j/B_j$ for $B_j < 0$, and $A_j < 0$ for $B_j = 0$. Intersecting these bounds with the permitted range gives $\mathcal{I}(x, f)$, excluding comparison ties (Appendix A.3). These bounds expose the tension between rejecting competitors and preserving useful exploration and repair. The certificate fixes the pools; changing ρ can also change what is proposed.

The improvement claim starts from a feasible incumbent. First-feasible discovery instead uses target set $\mathcal{F}$, provided the incumbent accepts a selected feasible route (Appendix A.1).

Routing interpretation. A five-customer Euclidean TSPDL construction (Appendix A.4) exhibits a bounded coupling interval in which an infeasible cost-saving exploration is followed by selection of a feasible improvement over the incumbent. Weaker coupling selects a cheaper reference whose specified repair pool cannot improve the incumbent; stronger coupling rejects the useful exploration. Spatial rescaling produces a disjoint interval; another scale leaves no useful shared penalty. Appendix A.4 supplies the routes, resource checks, and exact bounds. This motivates state-dependent coupling. Appendix A.5 separately distinguishes generating an improvement from selecting it under finite sampling. The ideal-consensus result is provided in Appendix A.7.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Problems and datasets. We evaluate on two hard-constrained routing problems, TSPTW and TSPDL, following the synthetic benchmark protocol of PIP Bi et al. (2024). To further assess applicability to multi-vehicle settings, we additionally evaluate on CVRPBLTW Bi et al. (2026); Zhou et al. (2024).

Baselines. We compare against four groups of baselines. The heuristic baselines are LKH-3 Helsgaun (2017) and OR-Tools Furnon & Perron (2026). For TSPTW and TSPDL, the neural constructive baseline is POMO Kwon et al. (2020), while for CVRPBLTW, we compare against MVMoE/4E, MVMoE/4E-L Zhou et al. (2024), and ReLD-MoEL Huang et al. (2025). The masking-based feasibility-control baselines are PIP Bi et al. (2024) and LMask Li et al. (2026). The search-and-repair baseline is CaR Bi et al. (2026). Detailed implementation settings for baselines are provided in Appendix E.

Evaluation metrics. We report feasibility, solution quality, and runtime. *Sol. Infsb.* is the fraction of evaluated candidates violating at least one constraint; *Inst. Infsb.* is the fraction of instances with no feasible output. For successful instances, solution quality is evaluated using $\mathrm{Obj}_{\mathrm{succ}}$ and the reference gap

$$\mathrm{Gap}_{\mathrm{succ}}(\%) = \frac{100}{|\mathcal{S}_m|} \sum_{I \in \mathcal{S}_m} \frac{C_m(I) - C_{\mathrm{ref}}(I)}{C_{\mathrm{ref}}(I)}.$$

Here, $\mathcal{S}_m$ denotes the set of test instances for which method m produces at least one feasible solution, and $|\mathcal{S}_m|$ denotes its cardinality. For each $I \in \mathcal{S}_m$, $C_m(I)$ is the best feasible cost reported by method m, and $C_{\mathrm{ref}}(I)$ is its reference cost. $\mathrm{Obj}_{\mathrm{succ}}$ is the average of $C_m(I)$ over $\mathcal{S}_m$, and the factor 100 expresses the gap as a percentage. *Time* denotes test-set wall-clock inference time. Negative gaps are relative to heuristic references, not certified optima. Runtime comparisons use the reported native configurations.

Implementation details. Our static node encoder builds on the attention-based encoder design used in POMO (Kwon et al., 2020), with task-specific input features (Appendix D). We trained the model with a learning rate of 1×10^{-4} and a batch size of 128. For both training and validation we fixed $G = 20$, $\rho_{\min} = 0$, $\rho_{\max} = 3$, $\alpha_C = \beta_V = \lambda_x = \lambda_z = 1$, and $\alpha_V = \beta_C = \lambda_\rho = 0.1$. During validation, sampling temperature was set to 1.4. The split updates were unrolled for $K = 5$ iterations. All experiments were run using an NVIDIA RTX 6000 Ada GPU and two AMD EPYC 9354 32-core processors.

5.2 RESULTS ON TSPTW AND TSPDL

Table 1 shows 0.00% instance infeasibility and the lowest reported neural gaps for ALTER on TSPTW Medium at both sizes. On TSPDL Hard, ALTER leads the tested neural methods in both measures: 0.00% and 0.03% instance infeasibility, with gaps of 1.32% and 2.75% at 50 and 100 customers.

On TSPTW Hard at 100 customers, ALTER reports lower instance infeasibility than CaR (5.02% versus 21.12%), a smaller gap (0.16% versus 0.76%), and shorter runtime (49.4 s versus 155.7 s). LMask achieves 0.00% instance infeasibility with a 0.20% gap and shorter runtime at this size; it also has lower reported infeasibility, gap, and runtime than ALTER at 50 customers.

Table 1: Results on TSPTW and TSPDL. Bold indicates the lowest reported infeasibility or gap among tested learning-based methods. Time is test-set wall-clock time.

Problem	Group	Method	$n = 50$					$n = 100$				
			Infeasible %↓		Obj.↓	Gap↓	Time↓	Infeasible %↓		Obj.↓	Gap↓	Time↓
			Sol.↓	Inst.↓				Sol.↓	Inst.↓			
TSPTW Medium	Heuristic method	LKH3	0.00%	0.00%	13.02	–	3.9h	0.00%	0.00%	18.74	–	9.2h
		ORTools	15.77%	15.77%	13.04	0.45%	2.9h	0.52%	0.52%	19.31	3.07%	6.9h
	Constructive baseline	POMO*	14.92%	3.77%	13.68	5.23%	3.1s	18.77%	0.12%	20.78	10.93%	8.0s
	Masking-based feasibility control	POMO*+PIP	4.53%	0.90%	13.40	2.91%	5.9s	3.88%	0.19%	19.61	4.65%	31.5s
		POMO*+PIP-D	3.83%	0.65%	13.45	3.32%	5.8s	3.34%	0.03%	19.79	5.64%	31.4s
		LMask	**0.04%**	0.01%	13.24	1.68%	5.3s	0.03%	**0.00%**	19.47	3.92%	18.5s
	Search & repair	CaR	0.15%	**0.00%**	13.36	2.61%	65.4s	0.11%	**0.00%**	20.41	8.97%	155.2s
		Ours	0.13%	**0.00%**	13.23	**1.58%**	19.1s	**0.02%**	**0.00%**	19.31	**3.04%**	49.6s
TSPTW Hard	Heuristic method	LKH3	0.00%	0.00%	25.61	–	6.6h	0.00%	0.00%	51.24	–	26.1h
		ORTools	65.72%	65.72%	25.76	0.00%	1.2h	89.07%	89.07%	51.61	0.00%	46.1m
	Constructive baseline	POMO*	39.26%	35.25%	26.21	1.61%	3.1s	100.00%	100.00%	–	–	8.1s
	Masking-based feasibility control	POMO*+PIP	5.54%	2.67%	25.66	0.18%	5.9s	31.49%	16.27%	51.42	0.37%	31.6s
		POMO*+PIP-D	6.76%	3.07%	25.69	0.28%	5.9s	13.18%	6.48%	51.39	0.31%	31.5s
		LMask	**0.01%**	**0.00%**	25.63	0.10%	5.0s	**0.00%**	**0.00%**	51.34	0.20%	18.7s
	Search & repair	CaR	0.07%	0.01%	25.61	**0.01%**	66.1s	57.70%	21.12%	51.57	0.76%	155.7s
		Ours	1.83%	0.21%	25.64	0.13%	19.1s	19.14%	5.02%	51.29	**0.16%**	49.4s
TSPDL Hard	Heuristic method	LKH3	0.00%	0.00%	13.30	–	6.1h	0.00%	0.00%	20.70	–	17.6h
		ORTools	100.00%	100.00%	/	/	1.0m	100.00%	100.00%	/	/	7.0m
	Constructive baseline	POMO*	37.01%	29.25%	13.03	4.11%	3.0s	99.98%	99.85%	20.95	15.87%	8.0s
	Masking-based feasibility control	POMO*+PIP	4.53%	2.10%	13.66	3.13%	4.5s	28.54%	20.66%	22.30	12.67%	19.7s
		POMO*+PIP-D	3.89%	0.82%	13.80	3.95%	4.5s	12.84%	7.91%	22.83	12.31%	19.6s
		LMask	0.15%	0.03%	13.58	2.21%	5.1s	0.72%	0.20%	21.54	4.16%	11.6s
	Search & repair	CaR	9.27%	0.53%	14.12	6.39%	64.3s	39.92%	3.92%	22.67	10.31%	155.6s
		Ours	**0.14%**	**0.00%**	13.47	**1.32%**	19.1s	**0.24%**	**0.03%**	21.26	**2.75%**	50.4s

Table 2: Results on CVRPBLTW. Bold indicates the lowest reported infeasibility or gap among tested learning-based methods. Gaps use OR-Tools reference solutions.

Group	Method	$n = 50$					$n = 100$				
		Infeasible %↓		Obj.↓	Gap↓	Time↓	Infeasible %↓		Obj.↓	Gap↓	Time↓
		Sol.↓	Inst.↓				Sol.↓	Inst.↓			
Heuristic	OR-Tools	0.00%	0.00%	14.68	–	15.3m	0.00%	0.00%	25.34	–	34.9m
Constructive baseline	MVMoE/4E	**0.00%**	**0.00%**	15.95	8.78%	2.1s	**0.00%**	**0.00%**	27.14	7.33%	8.5s
	MVMoE/4E-L	**0.00%**	**0.00%**	15.96	8.88%	2.0s	**0.00%**	**0.00%**	27.16	7.41%	8.0s
	ReLD-MoEL	**0.00%**	**0.00%**	16.29	11.19%	2.0s	**0.00%**	**0.00%**	27.04	6.91%	8.5s
Search & repair	CaR	15.81%	2.30%	14.83	2.13%	12.3s	21.81%	2.90%	24.45	−1.77%	24.7s
	Ours	1.84%	0.40%	14.22	**−2.15%**	4.9s	2.38%	0.20%	23.66	**−5.37%**	14.4s

5.3 Results on CVRPBLTW

Table 2 reports CVRPBLTW results. ALTER obtains the lowest reported gaps among the tested neural methods, -2.15% and -5.37% at 50 and 100 customers, relative to the selected OR-Tools reference solutions. It also reports lower instance infeasibility and shorter runtime than CaR at both sizes.

5.4 Qualitative analysis

Figure 2 illustrates how the two route processes specialize and interact during split search. At x_1, the cost-oriented update produces a low-cost route with 49 violations ($C = 24.46$), after which the repair-oriented update z_1 reduces the violations to one with a moderate cost increase to 25.87. Conditioned on this repaired state, x_2 again explores a substantially lower-cost route ($C = 19.00$) while temporarily increasing violations to 15, and z_2 restores feasibility with cost 25.93. The displayed x_3 and z_3 updates remain feasible at the same reported cost. This trajectory illustrates feasibility restoration followed by stabilization of the reported cost and violation count. Additional qualitative examples across diverse instances and problem settings are provided in Appendix F.

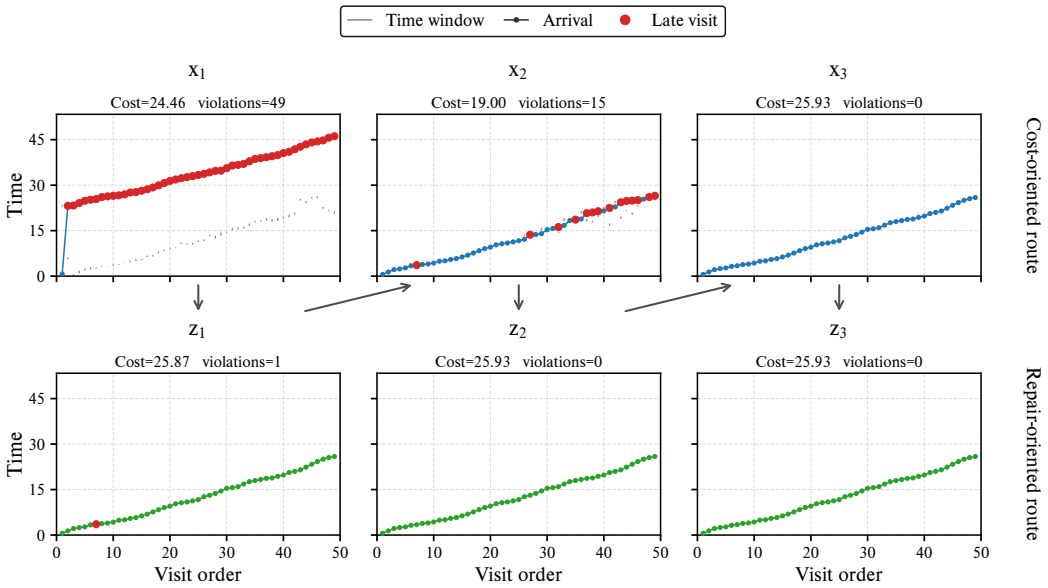


Figure 2: Qualitative split-search dynamics on a TSPTW-50 Hard instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red markers indicate late visits.

5.5 Ablation Study

Table 3 highlights the contributions of the proposed coordination components. The exploration-only configuration produces no feasible solution, with 100.00% solution and instance infeasibility, whereas the repair-only configuration achieves low infeasibility (1.67% solution and 0.19% instance infeasibility) but a relatively larger gap of 0.180%. Combining cost-oriented exploration and feasibility-oriented repair with disagreement memory preserves low infeasibility at 1.83% and 0.20%, respectively, while reducing the gap to 0.168%. This supports the benefit of alternating the two complementary search roles while carrying their disagreement across iterations. The full model with adaptive penalty control further reduces the gap to 0.126% while maintaining low solution and instance infeasibility of 1.83% and 0.21%, respectively. Together, these results show that alternating exploration and repair, disagreement feedback, and adaptive coupling contribute complementary effects to balancing solution quality and feasibility.

5.6 Transfer to Larger Instances

Table 4 evaluates scalability by directly applying the same learning-based models used in the 100-node setting to 200-node CVRPBLTW instances without retraining. ALTER reports 0.00% instance infeasibility with 1.46% solution infeasibility and achieves a reference gap of -7.86%. The result demonstrates within-family size transfer.

Table 3: Ablation on TSPTW-50 Hard. $\mathbf{u}$, ρ, and ψ denote disagreement memory, quadratic consensus, and adaptive penalty control. Fixed consensus uses $\rho = 1$.

Components					Infeasible %↓		Gap↓
x	z	$\mathbf{u}$	ρ	ψ	Sol.↓	Inst.↓	
✓					100.00%	100.00%	–
	✓				1.67%	0.19%	0.180%
✓	✓	✓			1.83%	0.20%	0.168%
✓	✓	✓	✓		14.35%	1.17%	0.131%
✓	✓	✓	✓	✓	1.83%	0.21%	0.126%

Table 4: Scalability on CVRPBLTW-200. All learning-based models are trained on 100-node instances and evaluated without retraining.

Method	Infeasible %↓		Obj.↓	Gap↓	Time↓
	Sol.↓	Inst.↓			
MVMoE/4E	**0.00%**	**0.00%**	50.94	8.98%	4.7s
MVMoE/4E-L	**0.00%**	**0.00%**	50.76	8.62%	4.6s
ReLD-MoEL	**0.00%**	**0.00%**	49.63	6.19%	4.7s
CaR	44.04%	6.25%	47.15	3.72%	11.8s
Ours	1.46%	**0.00%**	42.37	**−7.86%**	5.5s

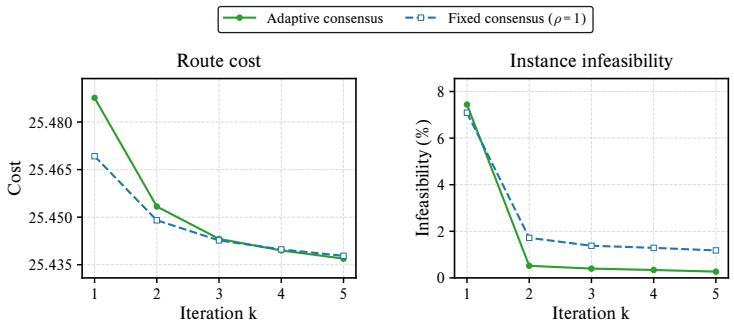


Figure 3: Effect of adaptive penalty control on TSPTW-50 Hard. Left: route cost across split iterations. Right: instance infeasibility. Fixed consensus uses $\rho = 1$.

5.7 Analysis of Split Iterations

Figure 3 shows how route cost and instance infeasibility evolve over the split iterations. For both adaptive and fixed consensus, route cost decreases rapidly during the first few iterations and changes only marginally after $k = 3$. The feasibility trajectories differ more substantially: adaptive consensus sharply reduces instance infeasibility by the second iteration and continues to improve through $k = 5$, whereas fixed consensus remains at a noticeably higher infeasibility level.

These trajectories suggest that the early iterations account for most of the cost reduction, while later iterations continue to contribute to feasibility restoration under adaptive coupling. We therefore use $K = 5$ throughout the experiments, which retains these later feasibility improvements after route cost has largely stabilized within the evaluated horizon.

6 Conclusion

We introduced ALTER, an alternating search framework that coordinates cost-oriented exploration and feasibility-oriented repair through a cost–violation representation, accumulated disagreement, and learned penalty control. The analysis identifies finite-pool conditions for feasible improvement and separates generating a useful repair from selecting it. Experiments show low instance infeasibility and favorable gaps on draft-limit routing, together with the lowest reported costs among the tested neural methods on CVRPBLTW.

Limitations. Our evaluation primarily focuses on synthetic routing benchmarks, which provide controlled settings for isolating the interaction between cost-oriented exploration and feasibility-oriented repair. While these benchmarks cover multiple hard-constraint structures and include transfer to larger problem sizes, they do not capture the full variability of real-world routing environments, such as heterogeneous operational rules, noisy inputs, or changing demand patterns.

AI Use Statement

We used generative AI tools only for language polishing, including grammatical correction and improvements to clarity, phrasing, and readability. All research ideas, methods, experiments, analyses, scientific claims, and conclusions were developed by the authors. All AI-assisted text was reviewed by the authors, who take full responsibility for the final content of this work.

REFERENCES

Maria Battarra, Artur Alves Pessoa, Anand Subramanian, and Eduardo Uchoa. Exact algorithms for the traveling salesman problem with draft limits. *European Journal of Operational Research*, 235 (1):115–128, 2014.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations, Workshop Track*, 2017.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

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*, 2024.

Jieyi Bi, Zhiguang Cao, Jianan Zhou, Wen Song, Yaoxin Wu, Jie Zhang, Yining Ma, and Cathy Wu. Towards efficient constraint handling in neural solvers for routing problems. In *International Conference on Learning Representations*, 2026.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2019.

Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and François Soumis. Time constrained routing and scheduling. In *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pp. 35–139. Elsevier, 1995.

Christophe Duhamel, Jean-Yves Potvin, and Jean-Marc Rousseau. A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science*, 31(1):49–59, 1997.

Yvan Dumas, Jacques Desrosiers, Eric Gélinas, and Marius M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

Vincent Furnon and Laurent Perron. OR-Tools routing library, 2026. Version 9.15.6755.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017.

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.

Jeffrey Ichnowski, Paras Jain, Bartolomeo Stellato, Goran Banjac, Michael Luo, Francesco Borrelli, Joseph E. Gonzalez, Ion Stoica, and Ken Goldberg. Accelerating quadratic optimization with reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

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

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*, 2020.

Tianyou Li, Haijun Zou, Jiayuan Wu, and Zaiwen Wen. LMask: Learn to solve constrained routing problems with lazy masking. In *International Conference on Learning Representations*, 2026.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-Opt. In *Advances in Neural Information Processing Systems*, 2023.

Mohammadreza Nazari, Afshin Oroojlooy, Martin Takáč, and Lawrence V. Snyder. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, 2018.

Jørgen Glomvik Rakke, Marielle Christiansen, Kjetil Fagerholt, and Gilbert Laporte. The traveling salesman problem with draft limits. *Computers & Operations Research*, 39(9):2161–2167, 2012.

Martin W. P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1–4):285–305, 1985.

Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

Sam R. Thangiah, Jean-Yves Potvin, and Tong Sun. Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research*, 23(11):1043–1057, 1996.

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

Yu Yao, Xiaoning Zhu, Hongyu Dong, Shengnan Wu, Hailong Wu, Lu (Carol) Tong, and Xuesong Zhou. ADMM-based problem decomposition scheme for vehicle routing problem with time windows. *Transportation Research Part B: Methodological*, 129:156–174, 2019.

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 International Conference on Machine Learning*, 2024.

A Feasible-Improvement Analysis and Proofs

This appendix proves the statements in Section 4 for the written scores and candidate-selection procedure. The routing constructions are mathematical examples.

A.1 Notation and assumptions

Let $\mathcal{T}$ be a finite nonempty set of complete routes, with finite cost $C(\tau) \geq 0$, nonnegative customer-indexed violations $\mathbf{v}(\tau)$, and

$$\mathcal{F} = \{\tau : \mathbf{v}(\tau) = \mathbf{0}\}, \qquad \Phi(\tau) = \sum_i v_i(\tau) + \sum_i \mathbb{1}\{v_i(\tau) > 0\}, \qquad \mathbf{y}(\tau) = (C(\tau), \mathbf{v}(\tau)).$$

Here and below, $\mathbb{1}$ denotes the indicator. Zero violation must characterize feasibility under the task evaluator; a constraint not represented by $\mathbf{v}$ must already be enforced in $\mathcal{T}$. No normalization of neural features is assumed to change the numerical coordinates used in the score. The finite consensus diameter is $D_y = \max_{\tau,\sigma\in\mathcal{T}} \|\mathbf{y}(\tau) - \mathbf{y}(\sigma)\|_2$.

Fix an incoming repair reference z, memory $\mathbf{u}$, and $\rho \in \mathcal{R} = [\rho_{\min}, \rho_{\max}] \subseteq \mathbb{R}_{\geq 0}$. All four role weights are positive. Suppressing the iteration index, define

$$\begin{aligned} b_x(\tau) &= \alpha_C C(\tau) + \alpha_V \Phi(\tau), \qquad q_x(\tau) = \tfrac{1}{2}\|\mathbf{y}(\tau) - \mathbf{y}(z) + \mathbf{u}\|_2^2, \\ b_z(\tau) &= \beta_C C(\tau) + \beta_V \Phi(\tau), \quad q_z(\tau; x) = \tfrac{1}{2}\|\mathbf{y}(x) - \mathbf{y}(\tau) + \mathbf{u}\|_2^2. \end{aligned} \tag{12}$$

Thus $J_x^\rho = b_x + \rho q_x$ and $J_z^\rho(\cdot; x) = b_z + \rho q_z(\cdot; x)$ are exactly the scores in Eq. (3).

Incumbent assumption. For output claims, assume a feasible incumbent g is retained and updated to the cheapest feasible route among itself and the selected exploration and repair routes. This is an explicit condition on the lexicographic rule in Algorithms 2–3. The reference z need not equal g. Score-selection conclusions do not require this assumption; the exact output-probability formula additionally follows the written winner-only updates, rather than retaining all unselected proposals.

For $\Delta > 0$, the target set is $\mathcal{G}_\Delta(g) = \{f \in \mathcal{F} : C(f) \leq C(g) - \Delta\}$. A target can exist without being generated, or be generated without being selected. Since the repair kernel generates complete routes, the intermediate reference does not change the global feasible optimum; it changes conditional proposal probabilities and candidate rankings. First-feasible discovery can instead use target set $\mathcal{F}$, provided the incumbent accepts a selected feasible route.

A.2 Role-weight conditions

Consider finitely many excursions (g_j, x_j, f_j), where $g_j, f_j \in \mathcal{F}$, $P_j = \Phi(x_j) > 0$, and

$$s_j = C(g_j) - C(x_j), \qquad h_j = C(f_j) - C(x_j), \qquad 0 \leq h_j < s_j.$$

The final gain is $s_j - h_j$. This identity alone does not justify alternation, since $C(x_j)$ cancels; generation and selection remain necessary.

Derivation of the role-weight conditions. **Step 1: exploration.** Since $\Phi(g_j) = 0$, subtracting role scores gives

$$b_x(x_j) - b_x(g_j) = -\alpha_C s_j + \alpha_V P_j.$$

The difference is negative exactly when $\alpha_V/\alpha_C < s_j/P_j$.

Step 2: repair. Since $\Phi(f_j) = 0$,

$$b_z(f_j) - b_z(x_j) = \beta_C h_j - \beta_V P_j,$$

which is negative exactly when $\beta_V/\beta_C > h_j/P_j$.

Step 3: a shared exchange rate. For one score $C + \lambda\Phi$, the two differences are $-s_j + \lambda P_j$ and $h_j - \lambda P_j$. Intersecting the strict inequalities over all excursions gives

$$\max_j h_j/P_j < \lambda < \min_j s_j/P_j.$$

Separate role ratios only require $\alpha_V/\alpha_C < \min_j s_j/P_j$ and $\beta_V/\beta_C > \max_j h_j/P_j$. For example, ratios $(s_1/P_1, h_1/P_1) = (0.5, 0.4)$ and $(s_2/P_2, h_2/P_2) = (5, 4)$ give disjoint common intervals, whereas role ratios 0.1 and 10 satisfy both conditions. □

The calculation concerns pairwise preferences without coupling.

A.3 Finite-pool selection and coupling intervals

Proof of Theorem 1. **Step 1: exploration selection.** The first line of Eq. (11) makes x strictly better than every other exploration candidate. Therefore it is the unique exploration minimizer, fixing the repair context and pool $\mathcal{Z}(x)$.

Step 2: repair selection. The second line gives a feasible target f with lower score than every non-target repair. If a non-target b minimized the repair score, it would satisfy $J_z^\rho(b;x) \le J_z^\rho(f;x)$, contradicting that inequality. Hence every minimizer belongs to $\mathcal{G}_\Delta(g)$; another, better target is allowed to beat f.

Step 3: retention. Let z^+ be the selected repair. Under the incumbent assumption, g^+ is feasible and $C(g^+) \le C(z^+) \le C(g) - \Delta$. Thus the retained gain is at least Δ.

Step 4: interval computation. For an exploration competitor a, put

$$A_a = b_x(x) - b_x(a), \qquad B_a = q_x(x) - q_x(a).$$

For a non-target repair b, put $A_b = b_z(f) - b_z(b)$ and $B_b = q_z(f;x) - q_z(b;x)$. Each required comparison is $A_j + \rho B_j < 0$, equivalent to

$$\begin{cases} \rho < -A_j/B_j, & B_j > 0, \\ \rho > -A_j/B_j, & B_j < 0, \\ A_j < 0, & B_j = 0. \end{cases} \tag{13}$$

Intersect these half-lines with $\mathcal{R}$. The result is an interval, possibly empty, and uses only the evaluated candidates and current state. Strict endpoints exclude score ties; endpoints of $\mathcal{R}$ are retained only when all strict comparisons hold. □

Exactness for a fixed exploration winner. If x is the unique exploration minimizer and every repair minimizer is a target, choose any such repair minimizer f. Finiteness yields strict inequalities against all non-target repairs, so $\rho \in \mathcal{I}(x,f)$. Consequently, $\bigcup_{f \in \mathcal{Z}(x) \cap \mathcal{G}_\Delta(g)} \mathcal{I}(x,f)$ characterizes this tie-independent success event exactly. The union need not be one interval. Favorable tied selections can succeed outside it; the theorem does not assume such tie handling.

Scoring margin. For a nonempty comparison family, let $m_{x,f}(\rho)$ be the smallest competitor-minus-witness score gap in Eq. (11). It is the minimum of finitely many affine functions, hence concave and piecewise affine. The certificate is $m_{x,f}(\rho) > 0$. If each score has evaluation error at most ε, every gap changes by at most 2ε; therefore $m_{x,f}(\rho) > 2\varepsilon$ preserves selection.

How saving, repair premium, and memory enter. Set $z = g$ only for this calculation, and write $\mathbf{u} = (u^C, \mathbf{u}^V)$, $\mathbf{v} = \mathbf{v}(x)$. Since $\mathbf{y}(x) - \mathbf{y}(g) = (-s, \mathbf{v})$, expansion and cancellation of $\|\mathbf{u}\|_2^2/2$ give

$$J_x^\rho(x) - J_x^\rho(g) = -\alpha_C s + \alpha_V P + \rho \left[\frac{s^2 + \|\mathbf{v}\|_2^2}{2} - s u^C + \langle \mathbf{v}, \mathbf{u}^V \rangle \right]. \tag{14}$$

Similarly, $\mathbf{y}(x) - \mathbf{y}(f) = (-h, \mathbf{v})$ and $q_z(x;x) = \|\mathbf{u}\|_2^2/2$, so

$$J_z^\rho(f;x) - J_z^\rho(x;x) = \beta_C h - \beta_V P + \rho \left[\frac{h^2 + \|\mathbf{v}\|_2^2}{2} - h u^C + \langle \mathbf{v}, \mathbf{u}^V \rangle \right]. \tag{15}$$

For $\mathbf{u} = \mathbf{0}$ and positive numerators, both comparisons hold when

$$0 \le \rho < \min \left\{ \frac{2(\alpha_C s - \alpha_V P)}{s^2 + \|\mathbf{v}\|_2^2}, \frac{2(\beta_V P - \beta_C h)}{h^2 + \|\mathbf{v}\|_2^2} \right\}.$$

Other candidates can also impose a positive lower bound, as the next construction shows. These pairwise comparisons require neither that g nor that the unrepaired x is inserted into an actual pool; full-pool selection is governed by Theorem 1. Changing ρ or the reference can change the neural proposals themselves, which are frozen for interval analysis.

A.4 A TSPDL EXAMPLE WITH BOUNDED COUPLING INTERVALS

The following construction uses the cumulative-load TSPDL convention in Appendix C. All routes start and end at depot 0. There are five unit-demand customers; customer 1 has draft limit 2, and the others have limit 5. Coordinates are

$$\mathbf{p}_0 = (0,0), \quad \mathbf{p}_1 = (0.25, 1.75), \quad \mathbf{p}_2 = (0.50, 1.50),$$
$$\mathbf{p}_3 = (0.50, 0.75), \quad \mathbf{p}_4 = (2.25, 1.50), \quad \mathbf{p}_5 = (1.25, 1.75).$$

Distances are Euclidean. The candidate customer sequences and their directly evaluated quantities are

Route	Customer sequence	C	Φ
g	$(1,2,4,3,5)$	9.175845	0
x	$(3,2,1,4,5)$	7.201863	2
d	$(3,2,1,5,4)$	6.739881	2
f	$(1,3,2,4,5)$	8.479901	0

These are constructed candidates, not benchmark measurements. Both x and d have violation $v_1 = 1$, with every other coordinate zero.

Step 1: certify the repair by resource accounting. More generally, moving customer j before a block $(b_1, \ldots, b_m)$, $m \geq 1$, changes

$$x = (0, \pi, b_1, \ldots, b_m, j, \sigma, 0) \quad \text{to} \quad f = (0, \pi, j, b_1, \ldots, b_m, \sigma, 0).$$

Here π, σ are unchanged sequences. Use a_i for customer demand, to distinguish it from distance d_{ij}, and let L_i^x denote cumulative load at customer i. Removing the block demand from the prefix of j, and adding a_j to each block prefix, yields

$$L_j^f = L_j^x - \sum_{r=1}^{m} a_{b_r}, \qquad L_{b_r}^f = L_{b_r}^x + a_j, \qquad L_i^f = L_i^x \quad (i \notin \{j, b_1, \ldots, b_m\}). \tag{16}$$

After the block, the same total demand has been accumulated, proving the last equality. Feasibility is therefore equivalent to these new loads not exceeding their draft limits and all unchanged customers remaining feasible.

For $j = 1$ and block $(3,2)$, customer 1's load drops from 3 to 1; the block loads increase from $(1,2)$ to $(2,3)$, below their limits 5. The remaining loads are $4, 5$. Thus f is feasible. If p is the predecessor of the block and q the successor of j, with the depot used at empty ends, only three edges change, giving exactly

$$h = C(f) - C(x) = d_{pj} + d_{jb_1} + d_{b_m q} - d_{pb_1} - d_{b_m j} - d_{jq}. \tag{17}$$

Here $s \simeq 1.973982$, $h \simeq 1.278038$, and $s - h \simeq 0.695944 > 0$. The relocation certifies a candidate.

Step 2: compute the selection interval. Set $z = g$, $\mathbf{u} = \mathbf{0}$, and use the reported weights $(\alpha_C, \alpha_V, \beta_C, \beta_V) = (1, 0.1, 0.1, 1)$. Fix

$$\mathcal{X} = \{g, x, d\}, \qquad \mathcal{Z}(x) = \{x, f, g\}, \quad \mathcal{Z}(d) = \{d, g\}, \quad \mathcal{Z}(g) = \{g\}.$$

The conditional pools are stipulated for this construction. The score lines, rounded here but evaluated from the coordinates, are

$$J_x^\rho(g) = 9.175845, \qquad J_z^\rho(x; x) = 2.720186,$$
$$J_x^\rho(x) = 7.401863 + 2.448302\rho, \qquad J_z^\rho(f; x) = 0.847990 + 1.316690\rho,$$
$$J_x^\rho(d) = 6.939881 + 3.466960\rho, \qquad J_z^\rho(g; x) = 0.917584 + 2.448302\rho.$$

Let $A = C(g) - C(x) > 0$, $B = C(g) - C(d) > A$, and $H = C(f) - C(x) > 0$, all defined by the exact Euclidean sums. Comparing x with d, x with g, and f with x gives, respectively,

$$\rho > \frac{2}{A + B}, \qquad \rho < \frac{2(A - 0.2)}{A^2 + 1}, \qquad \rho < \frac{2(2 - 0.1H)}{H^2 + 1}.$$

The comparison of f with g holds for all $\rho \geq 0$, since $C(f) < C(g)$ and $0 < H < A$ give both a smaller intercept and a smaller slope. The third bound is approximately 1.421896, leaving the exact useful interval

$$\left(\frac{2}{A+B}, \frac{2(A-0.2)}{A^2+1}\right) \simeq (0.453520, 0.724576). \tag{18}$$

For any target $0 < \Delta \leq C(g) - C(f)$, these strict inequalities certify selection and retention of a feasible improvement. Below the lower endpoint d wins; above the upper endpoint g wins, and their stipulated repair pools cannot improve the incumbent. Ties at the endpoints are excluded.

Step 3: distinguish adaptive opportunity from a guarantee. Multiply every spatial coordinate by $t > 0$, keeping demands, limits, weights, and pools fixed. Travel costs, A, B, H, scale by t, but violations do not. The lower bound and the two upper bounds become

$$L_t = \frac{2}{t(A+B)}, \qquad U_{x,t} = \frac{2(tA-0.2)}{t^2A^2+1}, \qquad U_{z,t} = \frac{2(2-0.1tH)}{t^2H^2+1}.$$

For $t = 3$, the useful interval is $(L_3, \min\{U_{x,3}, U_{z,3}\}) \simeq (0.151173, 0.205929)$, disjoint from Eq. (18). No fixed penalty serves both constructions. For $t = 4$, $L_4 \simeq 0.113380 > U_{z,4} \simeq 0.109735$, so no shared scalar admits the desired transition for these pools. This demonstrates possible state dependence and scalar limitations.

A.5 Finite sampling: generation versus selection

Condition on the full incoming solver state and the chosen ρ. Assume each pool contains G conditionally independent, identically distributed route samples, with $G \geq 1$. Rank distinct supported routes by score and a fixed deterministic route-level tie rule.

Step 1: selection from one pool. Let supported routes be ordered $\tau_1 \prec \cdots \prec \tau_N$, with masses p_i and cumulative masses $F_i = \sum_{j \leq i} p_j$, $F_0 = 0$. For G draws, the event "no sample precedes τ_i" has probability $(1 - F_{i-1})^G$; the subevent "every sample follows τ_i" has probability $(1 - F_i)^G$. Subtraction gives the exact selection probability

$$W(\tau_i; G) = (1 - F_{i-1})^G - (1 - F_i)^G. \tag{19}$$

Duplicates are permitted; the formula ranks distinct routes, not sample positions.

Step 2: the complete exploration–repair transition. Let $W_x(a)$ be Eq. (19) for exploration, and $W_z(b \mid a)$ its conditional repair counterpart. Set $Q_z(a) = \sum_{b \in \mathcal{G}_\Delta(g)} W_z(b \mid a)$. If the selected exploration a is already a target, the incumbent retains the gain. Otherwise the gain reaches Δ exactly when the repair winner is a target. Conditioning on a therefore gives

$$\Pr\{g^+ \in \mathcal{G}_\Delta(g) \mid \text{incoming state}, \rho\} = \sum_a W_x(a) \left[\mathbb{1}\{a \in \mathcal{G}_\Delta(g)\} + \mathbb{1}\{a \notin \mathcal{G}_\Delta(g)\} Q_z(a)\right]. \tag{20}$$

An infeasible exploration route matters through the conditional repair success $Q_z(a)$, not its cost saving alone. This identity assumes winner-only feasible retention.

Step 3: why more proposals need not improve the selected output. Suppose a conditional repair distribution assigns probability $q > 0$ to the lowest-scored non-target route, probability $p > 0$ to the next-ranked target, and the remaining mass to higher-ranked non-targets, with $p + q \leq 1$. The target wins exactly when the lowest-scored non-target is absent and the target is present. Hence

$$\Pr\{\text{repair winner is a target}\} = (1-q)^G - (1-p-q)^G, \tag{21}$$

which tends to zero, although the chance of generating a target, $1 - (1-p)^G$, tends to one. The construction in Appendix A.4 realizes the score ordering at $\rho = 2$: $x \prec f \prec g$. Assigning masses $(0.1, 0.2, 0.7)$ gives success probability $0.9^G - 0.7^G$. These are constructed probabilities. Retaining the cheapest feasible route from all proposals would remove this particular fixed-reference selection loss.

A sufficient finite-budget bound under score separation. Suppose a set of promising exploration references has proposal mass p_x and is ranked ahead of every other exploration route. For every such reference, suppose target repairs have conditional mass at least p_z and are ranked ahead of all non-target repairs. The first pool contains a promising reference with probability $1-(1-p_x)^G$, in which case it is selected. Conditional on that event, repair succeeds with probability at least $1-(1-p_z)^G$. Thus

$$\Pr\{g^+ \in \mathcal{G}_\Delta(g)\} \geq [1-(1-p_x)^G][1-(1-p_z)^G].$$

The bound links finite-budget success to proposal coverage and score separation. Together with Eq. (21), it separates the benefit of more generation from the effect of the selection rule.

A.6 Why small lateness does not imply inexpensive repair

We account for repair resources explicitly. The following TSPTW construction demonstrates that even arbitrarily small lateness can require substantial repair cost, despite bounded Euclidean geometry.

Proposition 1 (Vanishing lateness, nonvanishing repair premium)**.** There is a family of unit-square Euclidean TSPTW instances and infeasible routes x_ε with total lateness $V(x_\varepsilon)=\varepsilon\to 0$, such that every feasible route costs at least $2(\sqrt{2}-1)$ more than x_ε.

Proof. **Step 1: define the instance.** Put the depot at $(0,0)$ and customers $A=(1,0)$, $B=(1,1)$, and $C=(0,1)$. Departure is at time zero, service times are zero, and windows are

$$A:[0,3/2], \qquad B:[0,10], \qquad C:[0,3-\varepsilon], \qquad 0<\varepsilon<2-\sqrt{2}.$$

There is no binding return-depot deadline.

Step 2: evaluate the infeasible route. The route $x_\varepsilon=(0,A,B,C,0)$ costs 4 and visits customers at times $1,2,3$. Only C is late, by exactly ε.

Step 3: exhaust feasible orders. Every feasible route must visit A first: visiting either other customer first yields arrival at A of at least $1+\sqrt{2}>3/2$. Of the two orders starting with A, the order through B then C is infeasible. The remaining order $f=(0,A,C,B,0)$ reaches C at $1+\sqrt{2}<3-\varepsilon$ and B before 10, so it is feasible. Its cost is $2+2\sqrt{2}$. Therefore the unique feasible order has premium $2(\sqrt{2}-1)>0$, independent of ε. □

No uniform repair-cost bound $\omega(V)$ with $\omega(t)\to 0$ can hold over this family: taking $\varepsilon\to 0$ contradicts the fixed premium. The deadline varies while the geometry stays fixed. Along this family, $V\to 0$, while $\Phi(x_\varepsilon)=1+\varepsilon$. It shows why current lateness is not a calibrated measure of repair cost.

A.7 Target consistency under exact consensus

The finite-pool analysis does not need global optimality. We next characterize the target of the ideal exact-consensus formulation. Define $w_C=\alpha_C+\beta_C$, $w_\Phi=\alpha_V+\beta_V$, and assume $\mathcal{F}\neq\emptyset$. Let

$$C^\star_{\mathcal{F}}=\min_{f\in\mathcal{F}} C(f), \qquad \Gamma_{\mathrm{EP}}=\max_{\tau\in\mathcal{T}\setminus\mathcal{F}} \frac{[C^\star_{\mathcal{F}}-C(\tau)]_+}{\Phi(\tau)},$$

with $\Gamma_{\mathrm{EP}}=0$ when all routes are feasible.

Proposition 2 (Target consistency of consensus splitting)**.** Under exact consensus,

$$\min_{\substack{x,z\in\mathcal{T}\\ \mathbf{y}(x)=\mathbf{y}(z)}} \tilde{C}(x)+\tilde{V}(z)=\min_{\tau\in\mathcal{T}}\{w_C C(\tau)+w_\Phi\Phi(\tau)\}.$$

If $w_\Phi>w_C\Gamma_{\mathrm{EP}}$, every optimizer of the right-hand side is feasible and attains the optimal feasible cost.

Proof. **Step 1: reduce a consensus pair.** Equality of $\mathbf{y}$ gives equal cost and all customer violations, hence $\Phi(x)=\Phi(z)$. The pair objective equals $Q(x)=w_C C(x)+w_\Phi\Phi(x)$, and is at least $\min_\tau Q(\tau)$.

Step 2: achieve the reverse inequality. Any route τ induces the consensus pair (τ, τ) with objective $Q(\tau)$. A minimizing route exists by finiteness, proving equality of the minima.

Step 3: exclude infeasible optimizers under the additional condition. For infeasible τ, $\Phi(\tau) > 0$ and

$$Q(\tau) - w_C C^\star_{\mathcal{F}} \geq -w_C[C^\star_{\mathcal{F}} - C(\tau)]_+ + w_\Phi \Phi(\tau) \geq (w_\Phi - w_C \Gamma_{\mathrm{EP}})\Phi(\tau) > 0.$$

Every optimal feasible route attains $w_C C^\star_{\mathcal{F}}$, so no infeasible route can minimize Q. On feasible routes, $Q = w_C C$, which proves feasible-cost optimality. □

This is an ideal-target result.

B Training and Inference Details

B.1 Policy-gradient losses

We train the two route policies and the penalty controller using REINFORCE (Williams, 1992).

With $R^k_x(\tau) = -\mathcal{J}^x_k(\tau)$ from Eq. (3), the cost-oriented kernel is optimized with

$$\mathcal{L}^k_x = -\mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}_x}(\cdot|\mathbf{s}^x_k)} \left[\left(R^k_x(\tau) - b^k_x \right) \log \pi_{\boldsymbol{\theta}_x}(\tau \mid \mathbf{s}^x_k) \right], \tag{22}$$

and the repair-oriented kernel uses $R^k_z(\tau) = -\mathcal{J}^z_k(\tau)$ analogously:

$$\mathcal{L}^k_z = -\mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}_z}(\cdot|\mathbf{s}^z_k)} \left[\left(R^k_z(\tau) - b^k_z \right) \log \pi_{\boldsymbol{\theta}_z}(\tau \mid \mathbf{s}^z_k) \right]. \tag{23}$$

The penalty actor is optimized with a policy-gradient objective:

$$\mathcal{L}^k_{\rho,\mathrm{actor}} = -\mathbb{E}_{\rho \sim p_{\boldsymbol{\psi}}(\cdot|\mathbf{s}^\rho_k)} \left[\left(R^k_\rho(\rho) - b^k_\rho \right) \log p_{\boldsymbol{\psi}}(\rho \mid \mathbf{s}^\rho_k) \right]. \tag{24}$$

The full controller loss also includes a critic regression term:

$$\mathcal{L}^k_\rho = \mathcal{L}^k_{\rho,\mathrm{actor}} + \lambda_{\rho,\mathrm{value}} \mathbb{E}\left[\left(\widehat{V}_{\rho,k} - R^k_\rho \right)^2 \right]. \tag{25}$$

Here, $\mathcal{L}^k_\rho$ is the full controller loss, $\mathcal{L}^k_{\rho,\mathrm{actor}}$ is the actor loss, $\lambda_{\rho,\mathrm{value}} \geq 0$ weights the critic regression term, $\widehat{V}_{\rho,k}$ is the critic prediction, and R^k_ρ is the immediate controller reward at iteration k. The expectation is estimated by the minibatch mean. The reward target is detached in the critic loss, and the reward-minus-baseline advantage is detached in the actor loss. The critic-loss weight $\lambda_{\rho,\mathrm{value}}$ is distinct from the overall controller-loss weight λ_ρ.

At split iteration k, b^k_x and b^k_z denote the baselines for the exploration and repair policies, respectively. The controller baseline is $b^k_\rho = \widehat{V}_{\rho,k}$, where $\widehat{V}_{\rho,k}$ is the critic prediction. The penalty parameterization is given in Appendix D.

B.2 Rollout procedure

Let $\mathrm{Inc}(g, \tau)$ denote the lexicographic incumbent update. We use g_k, $g_{k+1/2}$ and g_{k+1} for the incumbent before exploration, after exploration, and after repair, respectively. The penalty-controller reward is computed from g_k and g_{k+1}.

Algorithm 2 ALTER training

Require: Training distribution $\mathcal{D}$, batch size B, candidate count G, split steps K
Require: Loss weights $\lambda_x, \lambda_z, \lambda_\rho$, penalty range $[\rho_{\min}, \rho_{\max}]$, initial previous penalty ρ_{-1}
Ensure: Trained encoder, role-conditioned route policies, and penalty controller

for training step $m = 1, 2, \ldots$ **do**
  Sample a batch of instances $\{I_i\}_{i=1}^{B} \sim \mathcal{D}$
  Initialize $z_{0,i}$ as a random depot-start route; set $\rho_{-1,i} \leftarrow \rho_{-1}$
  Set $x_{0,i} \leftarrow z_{0,i}$, $\mathbf{u}_{0,i} \leftarrow \mathbf{0}$, and $g_{0,i} \leftarrow z_{0,i}$
  Initialize accumulated rollout loss $\mathcal{L}_{\text{roll}} \leftarrow 0$
  ZEROGRAD
  for $k = 0, \ldots, K-1$ **do**
    Encode the current repair reference state $(I_i, z_{k,i}, \mathbf{u}_{k,i})$
    Sample penalty $\rho_{k,i} \in [\rho_{\min}, \rho_{\max}]$ from the penalty controller
    Generate G cost candidates $\mathcal{X}_{k,i} \sim \pi_{\boldsymbol{\theta}_x}(\cdot \mid I_i, z_{k,i}, \mathbf{u}_{k,i}, \rho_{k,i})$
    Select $x_{k+1,i} \leftarrow \arg\min_{\tau \in \mathcal{X}_{k,i}} \mathcal{J}_k^x(\tau)$
    Set $g_{k+1/2,i} \leftarrow \mathrm{Inc}(g_{k,i}, x_{k+1,i})$
    Compute exploration-policy loss $\mathcal{L}_x^k$ from the split objective
    Generate G repair candidates $\mathcal{Z}_{k,i} \sim \pi_{\boldsymbol{\theta}_z}(\cdot \mid I_i, x_{k+1,i}, \mathbf{u}_{k,i}, \rho_{k,i})$
    Select $z_{k+1,i} \leftarrow \arg\min_{\tau \in \mathcal{Z}_{k,i}} \mathcal{J}_k^z(\tau)$
    Set $g_{k+1,i} \leftarrow \mathrm{Inc}(g_{k+1/2,i}, z_{k+1,i})$
    Compute repair-policy loss $\mathcal{L}_z^k$ from the split objective
    Compute penalty-controller loss $\mathcal{L}_\rho^k$ from $g_{k,i}$ and $g_{k+1,i}$
    Update disagreement memory:

$$\mathbf{u}_{k+1,i} \leftarrow \mathbf{u}_{k,i} + \mathbf{y}(x_{k+1,i}) - \mathbf{y}(z_{k+1,i})$$

    Accumulate

$$\mathcal{L}_{\text{roll}} \leftarrow \mathcal{L}_{\text{roll}} + \frac{1}{K}\left(\lambda_x \mathcal{L}_x^k + \lambda_z \mathcal{L}_z^k + \lambda_\rho \mathcal{L}_\rho^k\right)$$

  end for
  Backpropagate $\mathcal{L}_{\text{roll}}$
  Update parameters
end for

Algorithm 3 ALTER inference

Require: Test instances $\{I_i\}_{i=1}^{B}$, candidate count G, split steps K
Require: Trained encoder, exploration policy $\pi_{\boldsymbol{\theta}_x}$, repair policy $\pi_{\boldsymbol{\theta}_z}$, and penalty controller $p_{\boldsymbol{\psi}}$
Require: Initial previous penalty ρ_{-1}
Ensure: Retained routes $\{g_{K,i}\}_{i=1}^{B}$

Initialize $z_{0,i}$ as a random depot-start route; set $\rho_{-1,i} \leftarrow \rho_{-1}$
Set $x_{0,i} \leftarrow z_{0,i}$, $\mathbf{u}_{0,i} \leftarrow \mathbf{0}$, and $g_{0,i} \leftarrow z_{0,i}$
for $k = 0, \ldots, K-1$ **do**
  Encode the current repair reference state $(I_i, z_{k,i}, \mathbf{u}_{k,i})$
  Predict penalty

$$\rho_{k,i} \leftarrow \mathbb{E}_{p_{\boldsymbol{\psi}}(\cdot \mid \mathbf{s}_{k,i}^{\rho})}[\rho], \qquad \rho_{k,i} \in [\rho_{\min}, \rho_{\max}]$$

  Generate G cost candidates $\mathcal{X}_{k,i} \sim \pi_{\boldsymbol{\theta}_x}(\cdot \mid I_i, z_{k,i}, \mathbf{u}_{k,i}, \rho_{k,i})$
  Select $x_{k+1,i} \leftarrow \arg\min_{\tau \in \mathcal{X}_{k,i}} \mathcal{J}_k^x(\tau)$
  Set $g_{k+1/2,i} \leftarrow \mathrm{Inc}(g_{k,i}, x_{k+1,i})$
  Generate G repair candidates $\mathcal{Z}_{k,i} \sim \pi_{\boldsymbol{\theta}_z}(\cdot \mid I_i, x_{k+1,i}, \mathbf{u}_{k,i}, \rho_{k,i})$
  Select $z_{k+1,i} \leftarrow \arg\min_{\tau \in \mathcal{Z}_{k,i}} \mathcal{J}_k^z(\tau)$
  Set $g_{k+1,i} \leftarrow \mathrm{Inc}(g_{k+1/2,i}, z_{k+1,i})$
  Update disagreement memory:

$$\mathbf{u}_{k+1,i} \leftarrow \mathbf{u}_{k,i} + \mathbf{y}(x_{k+1,i}) - \mathbf{y}(z_{k+1,i})$$

end for
return $\{g_{K,i}\}_{i=1}^{B}$

B.3 Disagreement memory under changing penalties

The recurrence in Eq. (4) accumulates disagreement in fixed representation coordinates. To distinguish its two effects in candidate scoring, set $\mathbf{a}(\tau) = \mathbf{y}(\tau) - \mathbf{y}(z_k)$ for exploration. Then

$$\frac{\rho_k}{2}\|\mathbf{a}(\tau) + \mathbf{u}_k\|_2^2 = \frac{\rho_k}{2}\|\mathbf{a}(\tau)\|_2^2 + \langle \boldsymbol{\lambda}_k^{\text{eff}}, \mathbf{a}(\tau)\rangle + \frac{\rho_k}{2}\|\mathbf{u}_k\|_2^2, \qquad \boldsymbol{\lambda}_k^{\text{eff}} = \rho_k \mathbf{u}_k.$$

The last term is constant across a fixed candidate pool. Thus memory contributes a directional linear term, in addition to the quadratic agreement term. The same expansion holds for repair with $\mathbf{a}(\tau) = \mathbf{y}(x_{k+1}) - \mathbf{y}(\tau)$. The effective coefficient $\boldsymbol{\lambda}_k^{\text{eff}}$ is an interpretation of the existing score.

B.4 Ideal split formulation and update targets

The finite-pool scores in Section 3.1 are motivated by the ideal split problem

$$\min_{x,z\in\mathcal{T}} \quad \tilde{C}(x) + \tilde{V}(z) \qquad \text{s.t.} \qquad \mathbf{y}(x) = \mathbf{y}(z). \tag{26}$$

The route variables x and z retain different cost–violation preferences while agreeing at the representation level. Appendix A.7 states the additional condition under which the ideal target is feasible-cost optimal.

C Task Definitions and Violation Measures

TSPTW. The traveling salesman problem with time windows assigns each node i a time window $[e_i, \ell_i]$. The route must visit every customer once and return to the depot while minimizing travel distance. A customer visit is infeasible when the actual visit time exceeds its latest allowed time ℓ_i. Time-window routing has a long algorithmic literature (Solomon, 1987; Desrosiers et al., 1995), including exact TSPTW methods (Dumas et al., 1995) and feasibility-aware local search (Savelsbergh, 1985).

TSPDL. The draft-limit routing family was introduced and studied by Rakke et al. (2012); Battarra et al. (2014); the benchmark convention is specified below. The traveling salesman problem with draft limits assigns each customer i a demand d_i and a draft limit m_i. The route must visit every customer once and return to the depot while minimizing travel distance. A customer visit is infeasible when the accumulated load after visiting that customer exceeds its draft limit m_i.

CVRPBLTW. CVRPBLTW considers multiple vehicle routes under capacity, backhaul, route-length, and time-window constraints. Earlier routing formulations also combine backhauls and time windows (Thangiah et al., 1996; Duhamel et al., 1997). Each vehicle route follows the same τ notation above, and the total solution cost is the sum of route distances. We follow the CaR environment Bi et al. (2026), except that once all linehaul customers have been served, an empty vehicle may depart from the depot to serve remaining backhaul customers.

Common notation. Let

$$\mathcal{N} = \{0, 1, \ldots, n\}$$

be the node set, where node 0 is the depot and nodes $1, \ldots, n$ are customers. A vehicle route is written as

$$\tau = (0, \tau_1, \ldots, \tau_{|\tau|}, 0),$$

where $|\tau|$ is the number of customers served by the route and τ_t is the customer visited at position t. For TSPTW and TSPDL, $|\tau| = n$ and $\tau_1, \ldots, \tau_n$ form a permutation of all customers. For CVRPBLTW, a solution consists of multiple vehicle routes

$$\boldsymbol{\tau} = \{\tau^{(1)}, \ldots, \tau^{(M)}\},$$

where M is the number of vehicle routes and the routes collectively serve all customers exactly once.

Let

$$\mathbf{q}_i = (x_i, y_i)$$

be the coordinate of node i, where x_i and y_i are its two coordinate components. The Euclidean distance from node i to node j is

$$\Delta_{ij} = \|\mathbf{q}_i - \mathbf{q}_j\|_2.$$

Here, Δ_{ij} is the edge length between nodes i and j.

The distance of a route is

$$C(\tau) = \sum_{t=0}^{|\tau|} \Delta_{\tau_t,\tau_{t+1}}, \qquad \tau_0 = \tau_{|\tau|+1} = 0. \tag{27}$$

Here, $C(\tau)$ is the total travel distance of route τ, t indexes route positions, and $\Delta_{\tau_t,\tau_{t+1}}$ is the distance between consecutive nodes. For a multi-vehicle solution,

$$C(\boldsymbol{\tau}) = \sum_{m=1}^{M} C(\tau^{(m)}),$$

where $C(\boldsymbol{\tau})$ is the total solution cost, $\tau^{(m)}$ is the route of vehicle m, and M is the number of vehicle routes.

For a customer i contained in route τ, let

$$\mathrm{pos}_\tau(i)$$

denote its visit position. The task-specific customer violation is denoted by $v_i(\tau)$, and the depot violation is padded as

$$v_0(\tau) = 0.$$

TSPTW timing and violations. For TSPTW, let $A_t(\tau)$ be the arrival time at position t, and let $T_t(\tau)$ be the visit time after waiting. Starting from $T_0(\tau) = 0$, the rollout is

$$A_t(\tau) = T_{t-1}(\tau) + \Delta_{\tau_{t-1},\tau_t}, \qquad t = 1, \ldots, n, \tag{28}$$

$$T_t(\tau) = \max\{A_t(\tau), e_{\tau_t}\}. \tag{29}$$

Here, $A_t(\tau)$ is the arrival time, $T_t(\tau)$ is the actual visit time, $\Delta_{\tau_{t-1},\tau_t}$ is the travel time from the previous node, and e_{τ_t} is the earliest allowed visit time.

The time-window violation of customer i is

$$v_i(\tau) = \max\left\{0,\ T_{\mathrm{pos}_\tau(i)}(\tau) - \ell_i\right\}, \qquad i = 1, \ldots, n. \tag{30}$$

Here, $v_i(\tau)$ is the lateness of customer i, $T_{\mathrm{pos}_\tau(i)}(\tau)$ is its visit time, and ℓ_i is its latest allowed visit time.

TSPDL loads and violations. For TSPDL, the depot has

$$d_0 = 0.$$

Let $L_t(\tau)$ be the accumulated load after the node at position t is visited. The load recursion is

$$L_0(\tau) = 0, \qquad L_t(\tau) = L_{t-1}(\tau) + d_{\tau_t}, \qquad t = 1, \ldots, n. \tag{31}$$

Here, $L_t(\tau)$ is the accumulated load after position t, $L_{t-1}(\tau)$ is the previous load, and d_{τ_t} is the demand of the current customer.

The draft-limit violation of customer i is

$$v_i(\tau) = \max\left\{0,\ L_{\mathrm{pos}_\tau(i)}(\tau) - m_i\right\}, \qquad i = 1, \ldots, n. \tag{32}$$

Here, $v_i(\tau)$ is the excess load at customer i, $L_{\mathrm{pos}_\tau(i)}(\tau)$ is the accumulated load after visiting customer i, and m_i is its draft limit.

CVRPBLTW rollout and violation. CVRPBLTW combines capacity, backhaul, route-length, and time-window constraints. We follow the CaR setting Bi et al. (2026), allowing an empty vehicle to serve remaining backhaul customers once all linehaul customers have been served.

Customer i has time window $[e_i, \ell_i]$ and signed demand d_i, positive for linehaul and negative for backhaul. Let s_i denote the service duration at node i, with $s_0 = 0$ at the depot. Service durations are expressed in the same time units as the time-window bounds. Let Q denote vehicle capacity, $\bar{d}_i = d_i/Q$ the capacity-normalized demand, and $L_{\max}$ the route-length bound. For vehicle m, $\kappa_m \in \{0, 1\}$ indicates whether all linehaul customers have been served before its departure; $\kappa_m = 1$ corresponds to an empty departure.

Starting from $T_0^{(m)} = L_0^{(m)} = D_0^{(m)} = 0$, the resource states at visit $t = 1, \dots, n_m$ satisfy

$$\begin{aligned} T_t^{(m)} &= \max\Big\{T_{t-1}^{(m)} + s_{\tau_{t-1}^{(m)}} + \Delta_{\tau_{t-1}^{(m)},\tau_t^{(m)}}, e_{\tau_t^{(m)}}\Big\}, \\ L_t^{(m)} &= L_{t-1}^{(m)} + \bar{d}_{\tau_t^{(m)}}, \\ D_t^{(m)} &= D_{t-1}^{(m)} + \Delta_{\tau_{t-1}^{(m)},\tau_t^{(m)}}. \end{aligned} \tag{33}$$

Here, $T_t^{(m)}$ is the service-start time after waiting at customer $\tau_t^{(m)}$, before service is performed; $L_t^{(m)}$ is cumulative signed demand; and $D_t^{(m)}$ is distance traveled by vehicle m since departure. The $s_{\tau_{t-1}^{(m)}}$ accounts for service at the previous node before traveling to the current customer. Its capacity-normalized on-board load is $B_t^{(m)} = 1 - \kappa_m - L_t^{(m)}$.

For customer $i = \tau_t^{(m)}$, the time-window, delivery-capacity, backhaul, and route-length violation magnitudes are defined by

$$\begin{aligned} p_i^{\mathrm{tw}}(\tau) &= [T_t^{(m)} - \ell_i]_+, \\ p_i^{\mathrm{del}}(\tau) &= [L_t^{(m)} - 1]_+, \\ p_i^{\mathrm{bh}}(\tau) &= [-L_t^{(m)} - \kappa_m]_+, \\ p_i^{\mathrm{len}}(\tau) &= [D_t^{(m)} - L_{\max}]_+. \end{aligned} \tag{34}$$

Let $\mathcal{K} = \{\mathrm{tw}, \mathrm{del}, \mathrm{bh}, \mathrm{len}\}$ denote the set of constraint types. The total violation magnitude and infeasibility indicator for customer i are

$$v_i(\tau) = \sum_{\gamma \in \mathcal{K}} p_i^{\gamma}(\tau), \qquad c_i(\tau) = \mathbb{1}\{v_i(\tau) > 0\}. \tag{35}$$

Consensus representation and descriptors. For all tasks, the split solver compares solutions through a cost–violation representation. For a solution σ, where $\sigma = \tau$ for TSPTW and TSPDL and $\sigma = \boldsymbol{\tau}$ for CVRPBLTW, we define

$$\mathbf{y}(\sigma) = [C(\sigma), v_1(\sigma), \dots, v_n(\sigma)]^\top. \tag{36}$$

Here, $C(\sigma)$ is total travel cost and $v_i(\sigma)$ is the task-specific violation attached to customer i. In a multi-vehicle solution the customer retains the same coordinate regardless of its assigned vehicle, whereas cost is summed over vehicle routes. Under the convention in Section 3.3 and Appendix A,

$$\begin{aligned} V(\sigma) &= \sum_{i=1}^{n} v_i(\sigma), \qquad N_{\mathrm{inf}}(\sigma) = \sum_{i=1}^{n} \mathbb{1}\{v_i(\sigma) > 0\}, \\ \Phi(\sigma) &= V(\sigma) + N_{\mathrm{inf}}(\sigma). \end{aligned} \tag{37}$$

D Model Architecture

The model consists of a static node encoder, a reference-route encoder, a shared autoregressive decoder, and a penalty controller. The static node encoder builds on the attention-based encoder design used in POMO (Kwon et al., 2020). We retain its stacked multi-head self-attention and position-wise feed-forward structure, while adapting the input features. The exploration and repair

policies, $\pi_{\boldsymbol{\theta}_x}$ and $\pi_{\boldsymbol{\theta}_z}$, share the node encoder, reference-route encoder, and autoregressive decoder. The two roles use different reference routes, role embeddings, and objectives. Throughout, n denotes the number of customers, node 0 is the depot, and d is the embedding dimension. Brackets denote concatenation, and W and b with subscripts denote learned projection matrices and biases.

Static node encoding. For node $i \in \{0, \dots, n\}$, the static representation is

$$\mathbf{f}_i = [\mathbf{q}_i, \mathbf{c}_i, \mathbb{1}\{i = 0\}], \qquad H = \mathrm{Enc}_{\mathrm{node}}\left((W_{\mathrm{in}}\mathbf{f}_i + b_{\mathrm{in}})_{i=0}^{n}\right), \tag{38}$$

where $\mathbf{q}_i$ contains the coordinates, $\mathbf{c}_i$ contains task-specific attributes, and $\mathbb{1}\{i = 0\}$ identifies the depot. The output $H = (\mathbf{h}_i)_{i=0}^{n}$ contains d-dimensional embeddings that depend only on the instance.

Both encoders stack non-causal Transformer blocks. We write their common attention-block form as

$$\widetilde{X} = \mathrm{LN}_1\left(X + \mathrm{MHA}(X, Y, Y)\right), \qquad \mathcal{B}(X, Y) = \mathrm{LN}_2\left(\widetilde{X} + F_{\mathrm{ff}}(\widetilde{X})\right). \tag{39}$$

Here, MHA is multi-head scaled dot-product attention, with query sequence X and key/value sequence Y; $\mathrm{LN}_1, \mathrm{LN}_2$ are layer normalizations; and F_{ff} is a position-wise, two-layer feed-forward network with a ReLU activation. Setting $Y = X$ gives an encoder block. Different layers and modules have independent parameters.

Reference-route encoding. At split iteration k, the exploration policy uses reference z_k, whereas the repair policy uses the newly selected route x_{k+1}. Write the disagreement memory as $\mathbf{u}_k = [u_k^C, u_{k,1}^V, \dots, u_{k,n}^V]$, where u_k^C is the cost component and $u_{k,i}^V$ is the violation component of customer i; set $u_{k,0}^V = 0$ for the depot. For a complete reference solution τ, let $\mathrm{seq}(\tau) = (\zeta_0, \dots, \zeta_{N_\tau - 1})$ be the node sequence supplied to the reference encoder. Here, N_τ is the number of valid input tokens and ζ_r is the node ID at sequence position r. Padding tokens are excluded from N_τ.

$$\begin{aligned} \mathbf{d}_{k,r}(\tau) &= [P_r(\tau), v_{\zeta_r}(\tau), b_r(\tau), u_{k,\zeta_r}^V, u_k^C, V(\tau)], \\ \mathbf{s}_{k,r}^{(0)}(\tau) &= \mathbf{h}_{\zeta_r} + \mathrm{CPE}(r; N_\tau) + W_D \mathbf{d}_{k,r}(\tau), \\ S_k(\tau) &= \mathrm{Enc}_{\mathrm{route}}\left((\mathbf{s}_{k,r}^{(0)}(\tau))_{r=0}^{N_\tau - 1}\right). \end{aligned} \tag{40}$$

Here, r indexes route positions; P_r is the resource-state profile; $v_{\zeta_r}(\tau)$ is the violation assigned to the customer at sequence position r; b_r records prefix infeasibility within the current vehicle route; and V is the aggregate violation magnitude, excluding violation counts. Task-dependent input scaling is assumed. For even d, the cyclic positional encoding is

$$\mathrm{CPE}(r; N_\tau) = \left(\sin\left(\frac{2\pi j r}{N_\tau}\right), \cos\left(\frac{2\pi j r}{N_\tau}\right)\right)_{j=1}^{d/2},$$

where j indexes harmonics. The reference encoder has the same block form as the node encoder but separate parameters. Its outputs for the two roles are $S_k^{(x)} = S_k(z_k)$ and $S_k^{(z)} = S_k(x_{k+1})$. Neither the role nor the current penalty enters this encoding.

Role-conditioned decoding. For role $h \in \{x, z\}$, denoting exploration and repair respectively, the decoder forms a reference-conditioned context:

$$\begin{aligned} \bar{\rho}_k &= \frac{\rho_k - \rho_{\min}}{\rho_{\max} - \rho_{\min}}, \\ \mathbf{c}_k^{(h)} &= \mathcal{B}_{\mathrm{ref}}\left(F_c([E_\rho(\bar{\rho}_k), \mathbf{e}_h, F_u(\mathbf{u}_k)]), S_k^{(h)}\right). \end{aligned} \tag{41}$$

Here, $\rho_k \in [\rho_{\min}, \rho_{\max}]$ is the consensus penalty, with $\rho_{\min} < \rho_{\max}$; $\bar{\rho}_k$ is its normalized value; E_ρ is a learned affine embedding; and $\mathbf{e}_h$ is a learned role embedding. The two-layer ReLU networks F_u and F_c embed the disagreement-memory vector and combine the conditioning signals, respectively. The cross-attention block $\mathcal{B}_{\mathrm{ref}}$ follows Eq. (39), and its output $\mathbf{c}_k^{(h)}$ is fixed during construction of a candidate route. For transfer to larger instances, the customer-wise memory vector is adaptively average-pooled to the number of customers used during training before being supplied to F_u; the scalar cost component is unchanged. This pooling is applied only to the input of F_u, with all model parameters fixed.

At decoding step t, the decoder computes

$$\begin{aligned}\mathbf{k}_j &= W_K \mathbf{h}_j, \\ \widetilde{\mathbf{q}}_t^{(h)} &= W_Q[\mathbf{c}_k^{(h)}, \mathbf{h}_{p_t}, \chi_t], \\ \mathbf{q}_t^{(h)} &= \mathcal{B}_{\mathrm{cand}}\left(\widetilde{\mathbf{q}}_t^{(h)}, (\mathbf{k}_j)_{j\in\mathcal{A}_t}\right).\end{aligned} \tag{42}$$

Here, p_t is the current node, χ_t is the evolving resource state, $\mathcal{A}_t$ is the available-node set, and j indexes candidates. The vectors $\mathbf{k}_j$, $\widetilde{\mathbf{q}}_t^{(h)}$, and $\mathbf{q}_t^{(h)}$ are the candidate key, initial query, and refined query, respectively. The candidate cross-attention block $\mathcal{B}_{\mathrm{cand}}$ has separate parameters from $\mathcal{B}_{\mathrm{ref}}$. Thus, reference-dependent information enters through the query, while candidate keys remain instance-dependent.

The next-node distribution is

$$\begin{aligned}\eta_{t,j}^{(h)} &= \lambda_{\mathrm{clip}} \tanh\left(\frac{(\mathbf{q}_t^{(h)})^\top \mathbf{k}_j}{\sqrt{d}}\right), \\ \pi_{\boldsymbol{\theta}_h}(p_{t+1} = j \mid \mathbf{s}_k^h, p_{\le t}) &= \mathrm{softmax}_j\left(\frac{\eta_{t,j}^{(h)} + M_{t,j}}{T_{\mathrm{samp}}}\right).\end{aligned} \tag{43}$$

Here, $\eta_{t,j}^{(h)}$ is the clipped compatibility score, $\lambda_{\mathrm{clip}} > 0$ is the clipping constant, $T_{\mathrm{samp}} > 0$ is the sampling temperature, $\mathbf{s}_k^h$ is the role-conditioned split state, and $p_{\le t}$ is the generated prefix. The mask is $M_{t,j} = 0$ for $j \in \mathcal{A}_t$ and $-\infty$ otherwise. Availability enforces customer uniqueness and depot/termination rules, but does not mask operational constraint violations. After each sampled visit, the environment updates χ_t and $\mathcal{A}_t$; the route likelihood is the product of the resulting conditional probabilities.

Penalty controller. Before selecting ρ_k, the controller uses the encoded repair reference $S_k^{(x)} = (\mathbf{s}_{k,r}^{(x)})_{r=0}^{N_{z_k}-1}$. Learned attention pooling produces

$$\omega_{k,r} = \mathrm{softmax}_r\left(\frac{\mathbf{w}_{\mathrm{p}}^\top W_{\mathrm{pK}} \mathbf{s}_{k,r}^{(x)}}{\sqrt{d}}\right), \qquad \mathbf{h}_k^{\mathrm{pool}} = \mathrm{LN}\left(W_{\mathrm{pO}} \sum_{r=0}^{N_{z_k}-1} \omega_{k,r} W_{\mathrm{pV}} \mathbf{s}_{k,r}^{(x)} + b_{\mathrm{pO}}\right). \tag{44}$$

Here, $\mathbf{w}_{\mathrm{p}}$ is a learned pooling query, $\omega_{k,r}$ is the attention weight of reference position r, and $\mathbf{h}_k^{\mathrm{pool}}$ is the pooled route representation. The controller then computes

$$\begin{aligned}\mathbf{s}_k^\rho &= [\mathbf{h}_k^{\mathrm{pool}}, E_{\mathrm{prev}}(\bar{\rho}_{k-1}), \mathrm{PE}(k)], \\ \mathbf{h}_{\rho,k} &= \mathrm{ReLU}\left(W_\rho \mathbf{s}_k^\rho + b_\rho\right), \\ \begin{bmatrix}\widetilde{\alpha}_k \\ \widetilde{\beta}_k\end{bmatrix} &= W_{\rho,\mathrm{act}} \mathbf{h}_{\rho,k} + b_{\rho,\mathrm{act}}, \\ \widehat{V}_{\rho,k} &= W_{\rho,\mathrm{val}} \mathbf{h}_{\rho,k} + b_{\rho,\mathrm{val}}.\end{aligned} \tag{45}$$

Here, k is the zero-based split-iteration index; $\mathbf{s}_k^\rho$ is the controller observation; $\mathbf{h}_k^{\mathrm{pool}}$ is the pooled encoding of the current repair reference; $\bar{\rho}_{k-1}$ is the normalized previous penalty; E_{prev} is its learned affine embedding; and $\mathrm{PE}(k)$ is the absolute iteration encoding. Brackets denote concatenation. The shared hidden state $\mathbf{h}_{\rho,k}$ is obtained using the learned matrix W_ρ, bias b_ρ, and element-wise ReLU activation. The actor and critic use separate learned output matrices $W_{\rho,\mathrm{act}}$ and $W_{\rho,\mathrm{val}}$, with biases $b_{\rho,\mathrm{act}}$ and $b_{\rho,\mathrm{val}}$, respectively. The scalars $\widetilde{\alpha}_k$ and $\widetilde{\beta}_k$ are the raw actor outputs before the positive transformation used to obtain the Beta shape parameters. The critic output $\widehat{V}_{\rho,k}$ is trained to predict the immediate controller reward and supplies the baseline b_ρ^k in the controller policy-gradient loss. The reward-minus-baseline advantage is detached when computing the actor loss. The absolute iteration encoding is $\mathrm{PE}(k) = (\sin(k/10000^{2j/d}), \cos(k/10000^{2j/d}))_{j=0}^{d/2-1}$, where j indexes frequency pairs and k starts from zero.

The Beta policy is

$$\begin{aligned} \alpha_k &= \mathrm{softplus}(\widetilde{\alpha}_k) + \epsilon, \quad \beta_k = \mathrm{softplus}(\widetilde{\beta}_k) + \epsilon, \\ a_k &\sim \mathrm{Beta}(\alpha_k, \beta_k), \qquad \rho_k = \rho_{\min} + (\rho_{\max} - \rho_{\min}) a_k. \end{aligned} \tag{46}$$

Here, softplus is a positive activation, $\epsilon > 0$ is a small positive floor, α_k, β_k are the Beta shape parameters, and a_k is a unit-interval action. At inference, the action is replaced by its mean $a_k = \alpha_k/(\alpha_k + \beta_k)$. The previous penalty ρ_{-1} is initialized to a prescribed value. The resulting ρ_k conditions both route updates at iteration k.

Task-specific inputs. The static attributes $\mathbf{c}_i$ of node i, resource states P_r at position r along the reference route, and decoder resource states χ_t at step t are summarized below.

Task	$\mathbf{c}_i$	P_r	χ_t
TSPTW	(e_i, ℓ_i)	A_r	T_t
TSPDL	(d_i, m_i)	L_r	L_t
CVRPBLTW	$(e_i, \ell_i, \bar{d}_i)$	$(T_r^{(m)}, B_r^{(m)}, D_r^{(m)})$	$(T_t^{(m)}, B_t^{(m)}, D_t^{(m)})$

(47)

Here, e_i, ℓ_i are time-window bounds; d_i is demand; m_i is the draft limit; and $\bar{d}_i$ is capacity-normalized demand, signed for CVRPBLTW. The resource variables are arrival time A, visit time after waiting T (the service-start time before service is performed for CVRPBLTW), cumulative demand L, on-board load B, and traveled route distance D. The superscript m identifies the vehicle. Time-window bounds and temporal inputs are normalized by the depot horizon. For CVRPBLTW, the distance input is normalized by the route-length bound $L_{\max}$.

Network configuration. We use an embedding dimension of $d = 128$, with six static-node encoder layers and two reference-route encoder layers. All Transformer blocks use eight attention heads, a feed-forward hidden width of 512, ReLU activations, and post-residual LayerNorm. The shared decoder contains one reference-context cross-attention block and one candidate cross-attention block. The disagreement-memory network F_u and conditioning network F_c each contain two linear layers with hidden and output widths of 128. The penalty controller uses one shared 128-unit ReLU hidden layer followed by separate linear actor and critic heads with two and one outputs, respectively. We use a logit-clipping constant of 10 and a Beta-parameter floor of 10^{-4}.

E Baselines and Evaluation Protocol

This section describes the implementation details of the baselines used in our experiments. Unless otherwise specified, we use the official implementations and the best officially released checkpoints when available; otherwise, we train the models using the official code and default configurations. We use the best feasible solution among the generated candidates for objective and gap computation.

E.1 Heuristic Methods

Heuristic methods include non-neural solvers used as strong reference solvers and general-purpose constraint-handling baselines.

LKH-3. LKH-3 Helsgaun (2017) is used as the reference solver for computing the optimality gap. For both TSPTW and TSPDL, we run LKH-3 with one run and 10,000 trials, without imposing a time limit. We enable the SPECIAL option and parallelize the experiments over 16 CPU cores. The best feasible LKH-3 objective values are stored and used as the reference values for gap computation.

OR-Tools. OR-Tools Furnon & Perron (2026) is used as a general-purpose routing solver baseline. We employ local cheapest insertion as the first solution strategy and guided local search as the local search metaheuristic. The time limit is set to 20 seconds for $n = 50$ and 40 seconds for $n = 100$, and the experiments are parallelized over 16 CPU cores.

E.2 Constructive Baselines

POMO. POMO Kwon et al. (2020) is a neural constructive solver that exploits multiple rollouts to improve exploration in routing problems. In hard-constrained problems, such as TSPTW and TSPDL, however, the original POMO formulation does not directly account for constraint violations during training. Thus, we use POMO*, the constrained variant used in PIP Bi et al. (2024), where constraint violations are handled through a fixed-dual Lagrangian training scheme. Conceptually, this variant evaluates a route by a penalized objective of the form $\mathcal{L}(\tau) = C(\tau) + \lambda V(\tau)$, where $C(\tau)$ is the route cost, $V(\tau)$ is the total constraint violation, and λ is a fixed penalty coefficient.

MVMoE. For CVRPBLTW, we evaluate MVMoE/4E and MVMoE/4E-L Zhou et al. (2024) using the official implementation and released checkpoints with the authors' inference settings.

ReLD-MoEL. We evaluate ReLD-MoEL Huang et al. (2025) on CVRPBLTW using the official implementation and released checkpoint with the authors' inference settings.

E.3 Masking-Based Feasibility Control

PIP. PIP Bi et al. (2024) is a feasibility-aware constructive baseline based on preventative infeasibility masking. At inference time, PIP directly generates preventative infeasibility masks. PIP-D is a decoder-based variant of PIP that learns to predict preventative infeasibility masks. During training, the PIP decoder is trained as a mask predictor with the PI mask enabled. During validation and inference, the decoder option is disabled, while the learned mask predictor is used to guide decoding. In the main tables, we report PIP-D results obtained from the pretrained checkpoint.

LMask. LMask Li et al. (2026) is evaluated using its official implementation and released checkpoint, following the authors' inference configuration.

E.4 Search and Repair

CaR. *CaR-POMO* Bi et al. (2026) is a construction-and-repair baseline built on the POMO constructive policy. For TSPTW-100 Hard, we retrain the model using the PIP Bi et al. (2024) TSPTW-100 Hard configuration, as the released model Bi et al. (2026) was trained under a different environment. The number of refinement steps is fixed to 5 during training and 20 during validation and inference.

E.5 Test-time augmentation

For learning-based constructive baselines, we apply 8-fold dihedral augmentation at validation and test time. For each routing instance with two-dimensional node coordinates $\mathbf{p}_i = (x_i, y_i)$ defined on the unit square $[0, 1]^2$, we generate eight geometric variants corresponding to the symmetries of the square:

$$\begin{aligned}\mathcal{A}(x, y) = \{&(x, y),\ (1 - x, y),\ (x, 1 - y),\ (1 - x, 1 - y),\\ &(y, x),\ (1 - y, x),\ (y, 1 - x),\ (1 - y, 1 - x)\}.\end{aligned}$$

These variants include the original coordinates, axis-wise reflections, coordinate exchange, and the reflected variants after coordinate exchange.

Only the node coordinates are transformed. All node-wise constraint features are copied to the same node indices without modification, since the above transformations are isometries that preserve the Euclidean distance structure of the route. Thus, the node identities and constraint values remain unchanged while only the geometric representation is augmented. In TSPTW, the time-window start and time-window end are repeated without changing their values. In TSPDL, the demand and draft-limit values are repeated in the same manner.

During inference, each original instance is expanded into eight geometric variants and decoded independently. After decoding, we map the resulting routes back to the original instance and select the best feasible solution among the candidates obtained from the augmentation dimension. The same 8-fold augmentation is applied to POMO, PIP, and PIP-D during validation and testing.

E.6 RUNTIME MEASUREMENT

The reported runtime is the total wall-clock inference time for solving the full test set of instances. All neural results reported in the main tables are measured on a single NVIDIA RTX 6000 Ada GPU. Classical solvers are evaluated using 16 CPU cores on a server equipped with two AMD EPYC 9354 32-Core processors, providing 128 logical CPUs in total. Data loading and logging overhead are excluded from the reported runtime.

F ADDITIONAL QUALITATIVE EXAMPLES

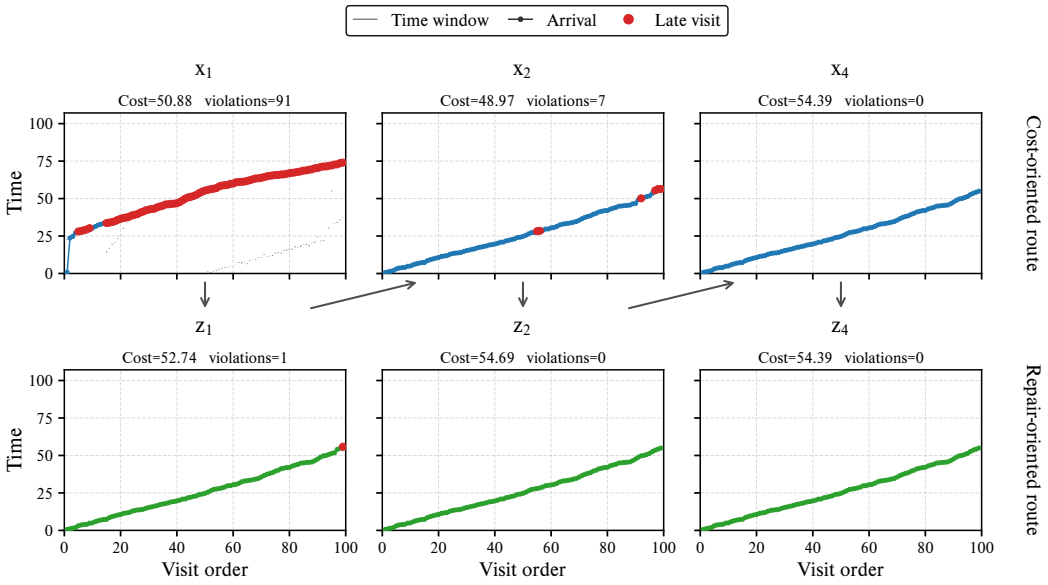


Figure 4: Qualitative split-search dynamics on a TSPTW-100 Hard instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red markers indicate late visits. Selected iterations are shown.

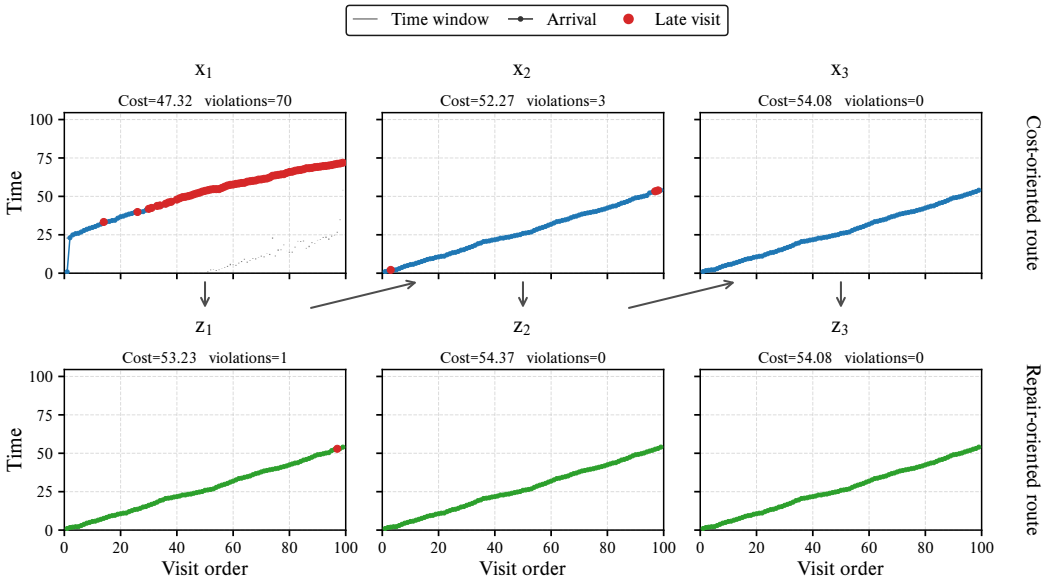


Figure 5: Qualitative split-search dynamics on a TSPTW-100 Hard instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red markers indicate late visits. Selected iterations are shown.

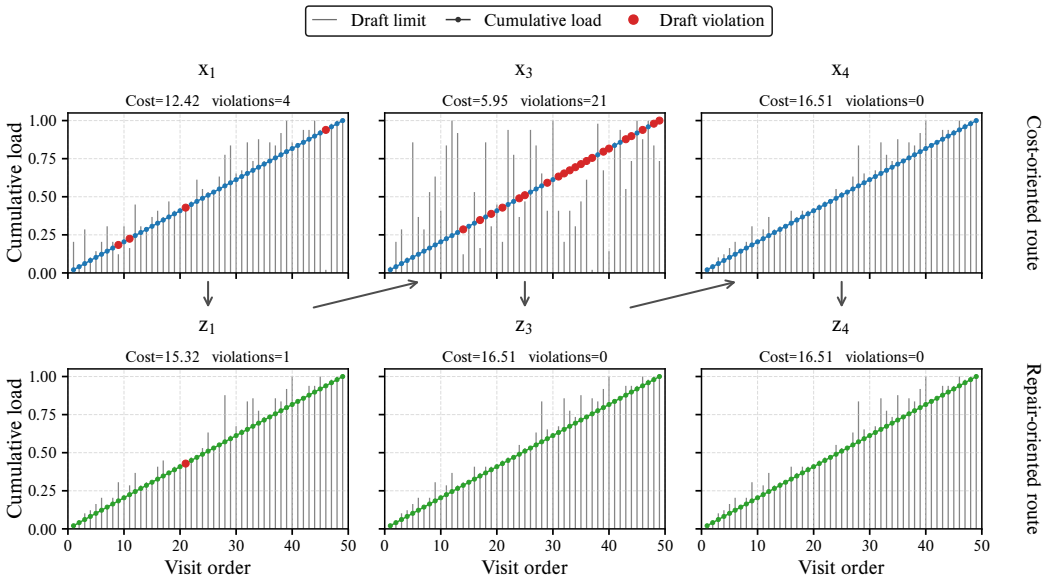


Figure 6: Qualitative split-search dynamics on a TSPDL-50 Hard instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red markers indicate draft-limit violations. Selected iterations are shown.

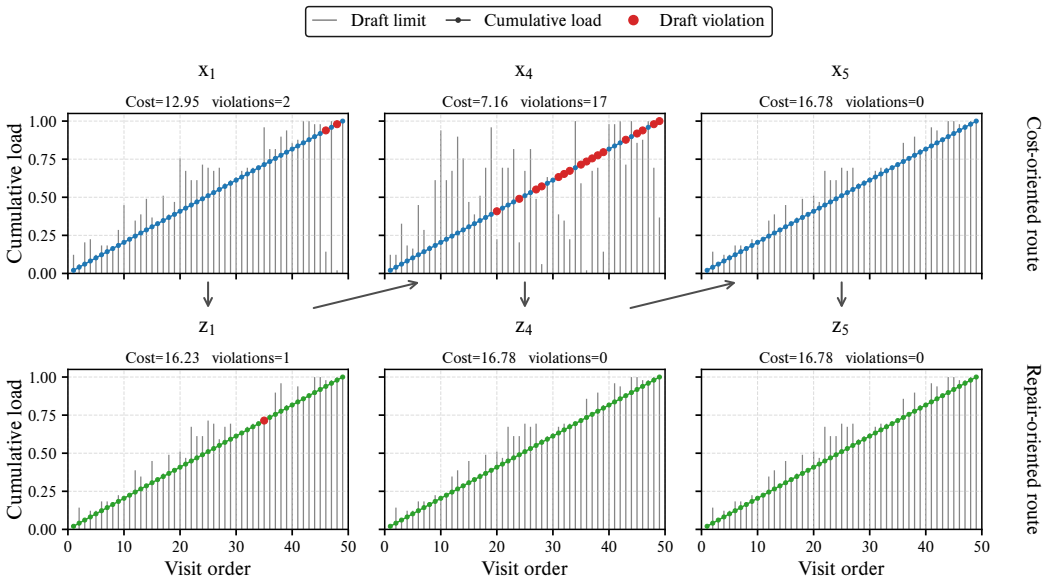


Figure 7: Qualitative split-search dynamics on a TSPDL-50 Hard instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red markers indicate draft-limit violations. Selected iterations are shown.

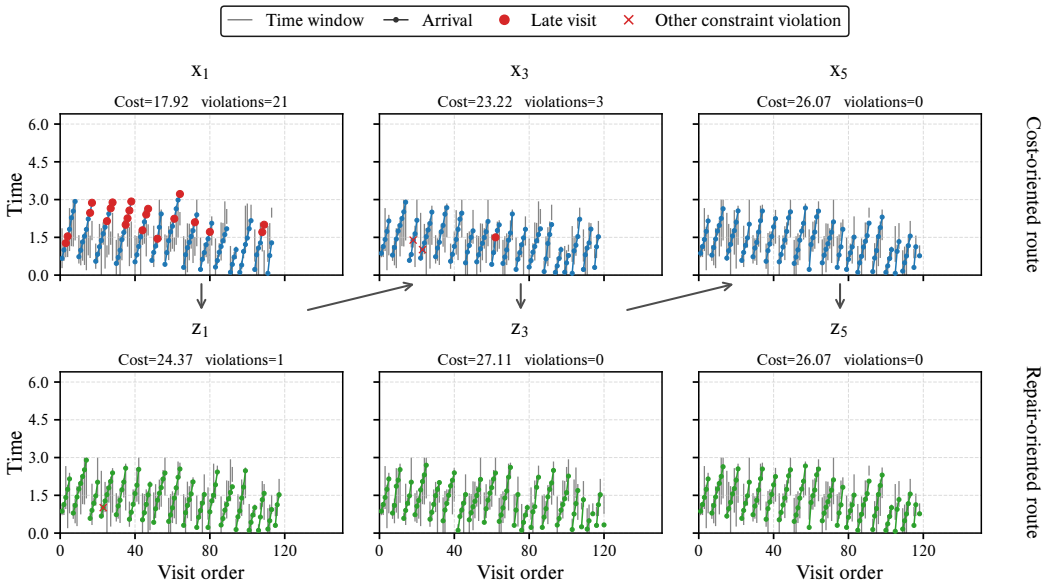


Figure 8: Qualitative split-search dynamics on a CVRPBLTW-100 instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red circles indicate late visits; red crosses indicate other constraint violations. Selected iterations are shown.

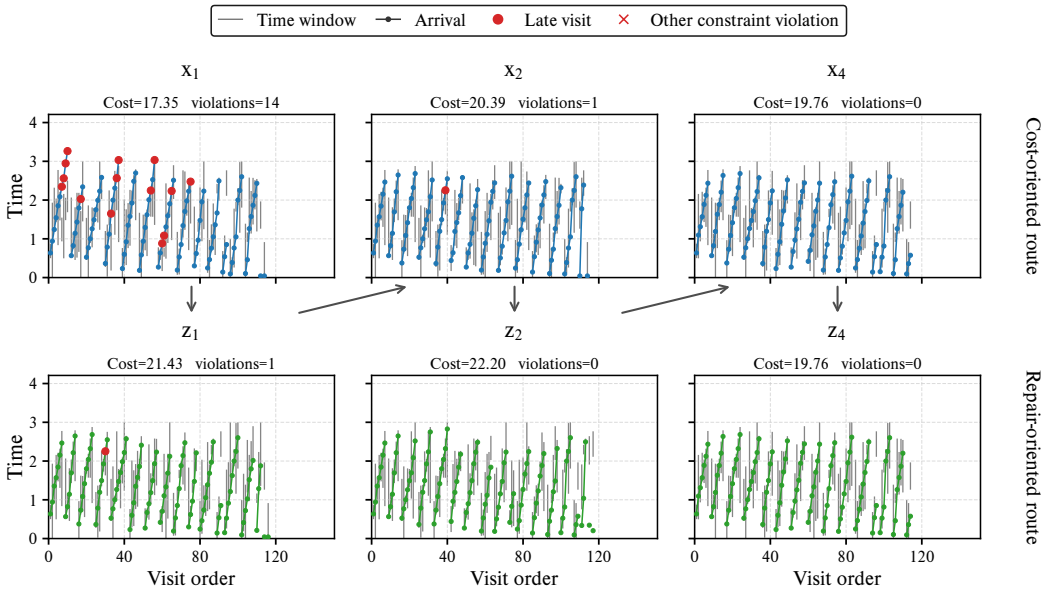


Figure 9: Qualitative split-search dynamics on a CVRPBLTW-100 instance. Top: cost-oriented routes x_k. Bottom: repair-oriented routes z_k. Red circles indicate late visits; red crosses indicate other constraint violations. Selected iterations are shown.

G Existing Assets and Licenses

Table 5 summarizes the external solvers, codebases, pretrained models, and benchmark resources used in this work.

Table 5: Existing assets used in this work.

Asset	Usage in this work	Source / version	License / terms	Notes
LKH-3	Reference heuristic solver for TSPTW and TSPDL and reference-cost computation	Helsgaun, version 3.0.7	Academic and non-commercial use	Used as a classical optimization baseline
OR-Tools	General-purpose routing baseline	Google OR-Tools, version v9.15.6755	Apache License 2.0	Used as a classical routing baseline
PIP / PIP-D	Feasibility-control baselines and synthetic benchmark protocol for TSPTW and TSPDL	`PIP-constraint`	MIT License	Official implementation and benchmark setup are used for evaluation
CaR	Construction-and-repair baseline and CVRPBLTW benchmark setup	`CaR-constraint`	MIT License	Official implementation and evaluation setting are used for comparison
LMask	Lazy-masking baseline for TSPTW and TSPDL	`LMask`	GNU GPL v3.0	Official implementation and released checkpoints are used for evaluation
MVMoE	MVMoE/4E and MVMoE/4E-L baselines for CVRPBLTW	`Routing-MVMoE`	MIT License	Official implementation and released checkpoints are used for evaluation
ReLD-MoEL	ReLD-MoEL baseline for CVRPBLTW	`reld-nco`	No explicit license stated in the repository	Official released implementation and checkpoint are used for evaluation
POMO*	Constrained constructive baseline for TSPTW and TSPDL	Implementation adapted from `PIP-constraint`; POMO backbone Kwon et al. (2020)	MIT License for the PIP-constraint implementation	Used as the constructive baseline following the constrained POMO setting