Towards Zero-Shot Neural Routing via Constraint Consequence Representation

Anonymous authors
Paper under double-blind review

Abstract

Extending classical solvers for Vehicle Routing Problems (VRPs) to new constraints often requires tailored designs of solver components. While recent neural routing generalists reuse decision rules across variants, their learned representations typically remain tied to predefined constraint types. To enable zero-shot generalization to new constraints, we propose Constraint Consequence REpresentation (CoRE). Motivated by resource extension in classical labeling algorithms, CoRE expresses every constraint through its candidate-level consequences in one shared space, in which a learned heuristic is reused on new constraints without fine-tuning. We integrate this representation into a framework in which one learned scorer guides both construction and refinement. Experiments on 110 VRP variants demonstrate that our model, trained solely on small-scale instances, generalizes effectively to unseen constraint combinations, new constraints, varying constraint hardness, and large-scale instances. CoRE achieves a mean reference gap of 3.68% across 110 routing variants, with a substantial improvement over state-of-the-art neural methods and over prior constraint representations within the same framework. We further show that consequences keep preferred candidates feasible, leading to improved quality, and that our representation serves as a generic module that enhances autoregressive construction and extends to node-oriented problems.

1 Introduction

Exact methods and heuristics solve many vehicle routing problem (VRP) variants effectively, including those with capacity, time-window, route-limit, and service-order constraints (Vidal et al., 2013). Yet extending a solver can still require changes to feasibility checks, route generation, or search operators, even when the new constraints resemble existing ones. These changes to constraint handling motivate reusing pretrained decision rules as the constraints change.

Although existing generalist routing models share parameters across variants and condition their decisions on numerical attributes and route state (Liu et al., 2024; Zhou et al., 2024; Berto et al., 2025; Li et al., 2025; Zhou et al., 2026), their inputs commonly retain predefined feature positions or constraint-specific learned maps. Such designs support transfer within the represented vocabulary, but an additional state variable may still require a new learned input mapping. For example, their zero-shot evaluations relax or recombine trained constraints or vary instance attributes such as the number of depots, while a new constraint, with its own state and feasibility check, is accommodated by adding and fine-tuning input parameters (Berto et al., 2025). This raises a question: can heterogeneous constraints share a common representation while retaining their distinct effects on candidate decisions?

One way to represent these effects comes from classical resource-constrained labeling: labels retain resource states and path history, which resource extension functions update as a path is extended (Irnich and Desaulniers, 2005; Irnich, 2008). These updated states are then used to check subsequent extensions and compare partial paths through dominance tests (Pessoa et al., 2020).

Motivated by this use of post-move information, we study how a pretrained model can handle new constraints without fine-tuning, a perspective yet to be explored to the best of our knowledge, and propose Constraint Consequence REpresentation (CoRE). For each candidate move, constraint-specific transitions and feasibility checks simulate the active constraints, producing normalized consequence vectors that shared neural maps combine with routing features and pooled constraint

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.

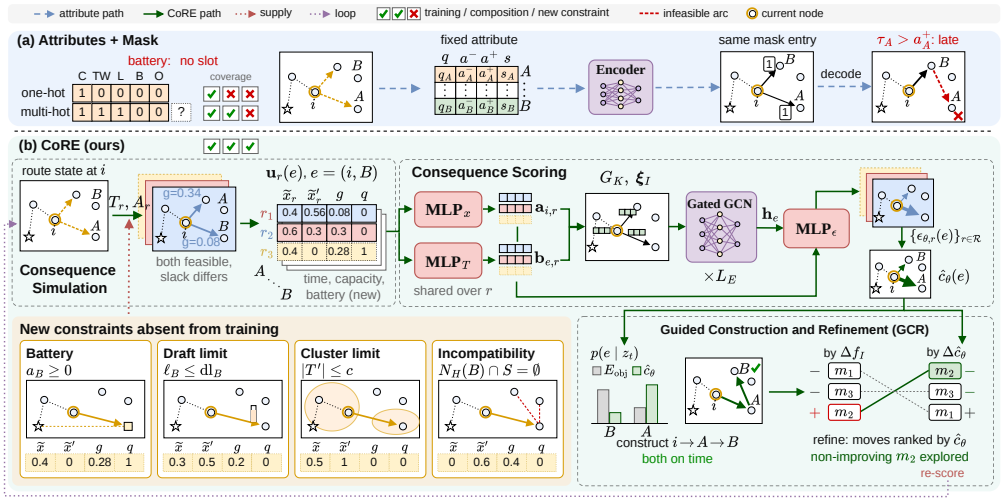


Figure 1: Constraint representations and the CoRE pipeline. (a) The node attributes, task bits, and feasibility mask of generalist routing models (Berto et al., 2025; Zhou et al., 2026) give feasible candidates A and B the same mask entry and have no input slot for a new constraint. (b) CoRE simulates each constraint on each candidate and scores the resulting consequences with maps shared across constraints.

context (Figure 1), while the simulator determines feasibility from the full current states. A new constraint provides these transitions and checks as it would to any constrained solver, and CoRE projects their outputs into the shared consequence space, where the pretrained scorer reads them with unchanged learned parameters. Because every constraint enters the same consequence space, the learned heuristic itself transfers to new compositions and constraints. Each grade refines the pass-or-fail outcome of a check into a margin to failure, and even the binary outcome alone yields a usable vector. We examine this claim through held-out combinations and new constraints in Tables 1 and 2, and trace it to the grade and event entries in Section 5.4.

Extensive experiments on 110 VRP variants show that one frozen model, trained on 100-customer instances, transfers to held-out constraint compositions, new constraints, and instances with up to 1,000 customers, attaining a 4.32% mean gap across the 89 held-out variants and improving on state-of-the-art neural generalists and on prior constraint representations within the same framework. Beyond this model, the same consequences enhance a separately trained POMO solver, lowering the gap of POMO* (Bi et al., 2026) by 23.6×, and extend to node-oriented problems such as independent set, vertex cover, and cut. We further analyze how consequences guide both models, showing that they keep preferred candidates feasible, where attribute inputs spend the slack that later customers need.

Our contributions are: (1) We identify that existing generalists read constraints through type-specific attributes and slack-blind masks, which ties a learned heuristic to its trained constraints; (2) We propose CoRE, which expresses every constraint through the consequences of its own transition and feasibility check in one shared space, pioneering the solution of new routing constraints by a pretrained model without fine-tuning; (3) We show that one frozen model transfers to unseen compositions, new constraints, and 10× larger instances, and that the same representation improves POMO and extends to node-oriented problems.

2 Related Work

Constraint representations in neural routing provide conditioning inputs for autoregressive policies such as the Attention Model (Kool et al., 2019) and POMO (Kwon et al., 2020), while generalists including MTPOMO, MVMoE, RouteFinder, and CaDA share parameters across variants through numerical attributes and task context (Liu et al., 2024; Zhou et al., 2024; Berto et al., 2025; Li et al., 2025; Zhou et al., 2026). URS (Zhou et al., 2026) extends this coverage by unifying predefined numerical attributes and service relations within one representation.

To update conditioning as construction proceeds, step-wise attention re-embeds nodes (Xin et al., 2021b), ReLD rebuilds decoder context (Huang et al., 2025), and CCL updates constraint relevance and node embeddings (Gui et al., 2026).

Expert-based methods instead combine learned policies: MoSES mixes specialized basis policies through learned weights (Pan et al., 2025), whereas CoEKS combines constraint-specific expert modules within its encoder for unseen combinations (Yu et al., 2026).

Against these approaches, we study whether candidate-level consequences provide reusable conditioning as the requirements change.

Learned guidance and constraint handling are integrated within routing methods such as Learning-to-delegate (Li et al., 2021), NeuroLKH (Xin et al., 2021a), L2Seg (Ouyang et al., 2026), and neural-guided ant colony optimization (Ye et al., 2023; Kim et al., 2025; Tran et al., 2026). Within this line, CaR (Bi et al., 2026) jointly trains construction and short-horizon refinement through a shared encoder, separate decoders, and Lagrangian construction penalties, while non-autoregressive and decomposition methods also examine transfer across graph sizes (Pirnay and Grimm, 2024; Luo et al., 2025; Drakulic et al., 2023; Luo et al., 2023; Liu et al., 2026). Whereas these approaches concern how learning interacts with construction, refinement, or decomposition, CoRE concerns the constraint information supplied to those procedures.

Neural representations beyond routing generate solutions for graph optimization, as in DIFUSCO (Sun and Yang, 2023), COExpander (Ma et al., 2025a), and NExCO (Wang et al., 2026), and encode solver states for mixed-integer linear programming, as in the variable–constraint graph of Gasse et al. (2019) with features from the current linear-programming relaxation. However, each model is trained for one problem class, so its learned heuristic remains specific to the constraints of that class. ANYCSP (Tönshoff et al., 2023) labels each constraint–value edge of a constraint satisfaction problem by whether choosing that value would satisfy the constraint, so one architecture accepts any instance of this class, and its models transfer across classes depending on the training distribution. Whereas these labels record whether a candidate satisfies a static relation, the consequence vector records how a candidate moves the state of each constraint along a route, its margin to failure, and whether it resets that state, and is designed for transfer to new constraints.

3 Preliminaries

Let I be a routing instance with minimization objective f_I, and let s contain the current route configuration and service history. Candidate actions are $e \in \Gamma_I(s)$. Each active constraint $r \in \mathcal{R}$ has internal state x_r; given the full state $z = (s, (x_r)_{r\in\mathcal{R}})$, written z_t at decision step t, it has an operational transition $T_r(z, e)$ and a feasibility check $A_r(z, e)$. An action is locally feasible when every applicable constraint accepts it:

$$A_{\mathcal{R}}(z, e) = \bigwedge_{r\in\mathcal{R}} A_r(z, e). \tag{1}$$

Committing an action updates the full decision state. Passing the current checks does not guarantee that the resulting partial solution can eventually be completed feasibly. We call the procedure that applies every T_r and A_r to a candidate the *simulator*; it defines constraint enforcement independently of the learned scorer, and CoRE projects its transitions into the consequence space.

4 Constraint Consequence Guidance

4.1 Challenges of Existing Constraint Representations

Existing generalists share one constraint representation and differ in how they process it: a shared encoder in MTPOMO, MVMoE, RouteFinder, CaDA, and URS (Liu et al., 2024; Zhou et al., 2024; Berto et al., 2025; Li et al., 2025; Zhou et al., 2026), node embeddings and constraint relevance recomputed as the route develops in CCL (Gui et al., 2026), and recombined constraint-specific experts in MoSES and CoEKS (Pan et al., 2025; Yu et al., 2026). This representation has two parts: 1) numerical attributes and 2) feasibility masks. Attributes describe each constraint by its parameters, which transfer across variants composed of trained constraints. However, they occupy input positions tied to predefined constraint types. For example, a new constraint with its own state, such as a

battery level, requires a new input map or a new expert trained on tasks containing it. On the other hand, masks enforce any constraint without new inputs. Yet, they discard how much slack a feasible candidate retains. For example, when two customers both pass the time-window check, the mask gives them the same entry whether one lies close to its deadline or both retain ample slack (Figure 1). Our approach instead supplies every constraint through the same candidate-level consequence space, described next.

4.2 Constraint Consequence Representation

To address these limitations, we note that the simulator of Eq. (1) already evaluates every active constraint on every candidate, while the policy receives only the resulting feasibility mask, even for candidates that pass a check at different distances from its bound. Inspired by resource extension in labeling algorithms (Irnich, 2008), we formally define the consequence vector, which places every constraint in one shared space. As shown in Figure 1, CoRE simulates each active constraint on each candidate, and then normalizes the resulting transition into one consequence vector per constraint–candidate pair. We thus expect a scorer fitted to this space to read a new constraint as another vector of the same form. The vector of every benchmark constraint is provided in Appendix A.3.

Consequence vector. Denote the state of constraint r before candidate e as x_r and its successor as $x'_{r,e} = T_r(z_t, e)$, where N_r normalizes both by the constraint's own bound, $\widetilde{x}_r = N_r(x_r)$ and $\widetilde{x}'_{r,e} = N_r(x'_{r,e})$. A grade $g_{r,e}$ is the normalized signed slack of the check A_r at this transition or, for a check without a scalar bound, the corresponding margin to failure (Appendix A.4). We denote the consequence vector of constraint r at candidate e as

$$\mathbf{u}_r(e) = (\widetilde{x}_r,\ \widetilde{x}'_{r,e},\ g_{r,e},\ q_{r,e}), \tag{2}$$

where $q_{r,e}$ marks a reset event such as a depot return or a battery replenishment. The two state entries follow from the transition alone, whereas the grade and event read that transition through the check and reset of the constraint. A consequence vector uses these four entries for every constraint (e.g., a time window and a route duration limit, with elapsed time rising from 20 to 30, every bound equal to 50, and a return leg of 5, both give $(0.4, 0.6, 0.3, 0)$) to represent constraints with different transitions.

New constraints. The consequence vector is broadly applicable to constraints that supply the transition T_r and check A_r that constrained search already executes in Eq. (1). Since the grade is the slack of the same comparison that N_r normalizes by its bound, $g_{r,e} \geq 0$ exactly when A_r accepts e, so the outcome of A_r alone already determines a vector. In Appendix A.4, we derive the vectors of four constraints absent from training, covering numerical bounds, a set-valued route state, and a relation without a scalar bound. In Appendix B, we further relate these vectors to the resource-label dominance of classical routing and characterize when they suffice for one decision rule to select an optimal action.

4.3 Scoring the Consequence Vectors

The non-autoregressive and autoregressive integrations of CoRE read constraints through their consequence vectors, sharing the encoding and masked policy described here and differing in how they aggregate over candidates (Section 4.4 and Appendix C).

Shared encoding. Given the consequence vectors $\{\mathbf{u}_r(e)\}_{r\in\mathcal{R}}$ of a candidate e, we compute the retained embedding $\boldsymbol{\kappa}_{\theta,r}(e) = \phi_\theta(\mathbf{u}_r(e))$, where ϕ_θ is a multilayer perceptron shared across constraints. Next, we sum-pool these embeddings over the active constraints, as in set encoders (Zaheer et al., 2017), and combine them with the routing features $\boldsymbol{\xi}_I(e, z_t)$, which describe the geometry, objective costs, and current solution without any constraint information:

$$\boldsymbol{\mu}_\theta(e) = \rho_\theta\Big(\tfrac{1}{\sqrt{|\mathcal{R}|}} \sum_{r\in\mathcal{R}} \boldsymbol{\kappa}_{\theta,r}(e),\ \tfrac{|\mathcal{R}|}{1+|\mathcal{R}|}\Big), \qquad \mathbf{h}_{\theta,r}(e) = \Phi_\theta(\boldsymbol{\kappa}_{\theta,r}(e), \boldsymbol{\xi}_I(e, z_t), \boldsymbol{\mu}_\theta(e)). \tag{3}$$

This step encodes the other active constraints into each candidate–constraint embedding through $\boldsymbol{\mu}_\theta$. Finally, each integration reads its candidate score from a sum over the active constraints, where a new constraint adds one summand and leaves every parameter shape unchanged, so a frozen model accepts it without new inputs.

Policy. Both integrations add a consequence bias to the content logit of a routing policy (Kool et al., 2019; Kwon et al., 2020). In CoRE, where the checks of Eq. (1) filter every candidate, the decision at

step t over a nonempty feasible candidate set is:

$$p_{\psi,\theta}(e \mid z_t) = \mathrm{softmax}_e[\ell_\psi(e \mid z_t, I) + b_\theta(e) + M_{\mathcal{R}}(z_t, e)], \tag{4}$$

where ℓ_ψ is the content logit, b_θ is the consequence bias, and $M_{\mathcal{R}}(z_t, e)$ is zero when $A_{\mathcal{R}}(z_t, e) = 1$ and $-\infty$ otherwise. The next action e_t is sampled from $p_{\psi,\theta}(\cdot \mid z_t)$.

4.4 Integration into Search and Construction

Guided Construction and Refinement (GCR). We adopt the edge-gated encoder of Joshi et al. (2022) over the K nearest candidates of each node and change its inputs and output. Its node and arc inputs add the pooled consequences, and instead of one edge score, which mixes objective and constraint cost, it scores each arc by its objective cost plus a learned constraint cost,

$$\widehat{c}_\theta(e) = E_{\mathrm{obj}}(e) + \lambda_\theta(z_t) \sum_{r \in \mathcal{R}} \epsilon_{\theta,r}(e), \tag{5}$$

where the learned λ_θ scales the constraint cost by the current states of all active constraints. Construction samples each action from Eq. (4) with logit $-\beta\widehat{c}_\theta(e)$, and refinement ranks the moves of standard operators (Vidal et al., 2013) by the learned cost of their removed and inserted arcs rather than by the objective alone, so the scorer that guides construction also steers refinement. Since the checks of Eq. (1) filter every candidate, the retained solution stays feasible, and it replaces the incumbent only when it lowers f_I (Appendix C.2).

Autoregressive construction. Unlike GCR, CoRE-POMO* builds each solution by autoregressive construction alone, with POMO (Kwon et al., 2020) as the backbone. Retaining POMO* of Bi et al. (2026), we add only a consequence context, pooled over candidates and constraints, to its decoder and a next-customer bias $b_\theta(e)$ to Eq. (4), with the visited-node and depot mask of POMO* in place of $M_{\mathcal{R}}$, since its attribute encoder receives the cost of slack only through penalties. It then recomputes the consequence vectors of every candidate after each commitment, as specified in Appendix C.3.

4.5 Training Objective

We train the scorer through the refinement it guides, building on the neural-guided local search of DeepACO (Ye et al., 2023) and on DyNACO (Tran et al., 2026), which localizes refinement around the decisions affected by its guidance. Each refinement rollout of GCR is rewarded by its relative objective improvement over the incumbent, and, beyond these designs, the learned costs guide the refinement itself in both training and inference (Appendices E.5 and E.6). We utilize PPO (Schulman et al., 2017) with the shared baseline of POMO (Kwon et al., 2020) to increase the probability of the perturbation choices in these improving rollouts. The neighborhood moves that follow are credited through the rollout outcome, leaving the objective independent of the refinement operators.

Grade reconstruction. The goal of the policy objective is to credit each candidate's grade through the feasibility and cost of the completed solution. While this objective does not directly fit the grades, we observe that (1) the simulator computes $g_{r,e}$ for every constraint–candidate pair at every decision, and (2) a shared readout $\widehat{g}_{\theta,r}(e) = H_\theta(\mathbf{h}_{\theta,r}(e))$ can regress it with a mean-squared loss $\mathcal{L}_{\mathrm{rec}}$, much as PIP (Bi et al., 2024) supervises a decoder to predict a binary infeasibility mask, here with the grade of every constraint as a graded target. Adding $\lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}}$ to the policy objective lowers infeasibility under tight compositions but raises the held-out gap (Section 5.4); CoRE uses $\lambda_{\mathrm{rec}} = 0$.

5 Experiments

We first evaluate how one pretrained model transfers across constraint compositions, instance sizes, new constraints, and constraint hardness. Next, we apply the representation to separately trained POMO and node-oriented policies. Finally, we analyze which consequence entries and training signals account for these results. Experimental details and additional results are presented in Appendices D and E.

5.1 Experimental Setup

Datasets. We train CoRE on 22 VRP variants with at most two active constraints and evaluate it on the full benchmark of URS (Zhou et al., 2026), which composes capacity (C), backhauls (B, BP),

Table 1: Routing benchmark. Gaps (%) to the reference solver and total inference time; reference: 60 s PyVRP (†: OR-Tools). Bold: best learned gap.

Training variants

	Method	CVRP				CVRPL				CVRPBP				CVRPTW			
		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$	
		Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓
	PyVRP	0.00%	2.0h	0.00%	16m	0.00%	2.0h	0.00%	16m	0.00%	2.0h	0.00%	16m	0.00%	2.0h	0.00%	16m
	URS	1.14%	38s	9.27%	83s	1.06%	40s	9.05%	97s	13.73%	42s	29.45%	103s	4.43%	44s	11.36%	125s
	CoEKS	2.04%	49s	42.04%	154s	2.10%	50s	41.04%	155s	4.32%	48s	55.27%	154s	21.95%	57s	58.98%	190s
	MoSES	0.93%	62s	70.47%	177s	0.89%	62s	71.17%	176s	2.63%	61s	61.85%	178s	22.19%	73s	59.05%	199s
	RF-T	1.24%	37s	26.75%	146s	1.11%	37s	26.60%	149s	2.87%	35s	33.15%	149s	26.69%	43s	73.57%	183s
	RF-POMO	1.49%	36s	24.42%	145s	1.49%	37s	23.77%	146s	3.53%	36s	35.33%	156s	19.82%	42s	58.49%	178s
	RF-MoE	1.35%	51s	27.42%	158s	1.24%	54s	31.99%	167s	3.08%	49s	91.07%	176s	19.64%	61s	53.26%	197s
	MTPOMO	1.65%	36s	62.78%	146s	1.61%	36s	62.39%	149s	3.70%	36s	71.80%	145s	20.70%	43s	59.44%	178s
	MVMoE	1.45%	50s	32.27%	174s	1.37%	56s	32.11%	182s	3.46%	49s	46.62%	183s	25.65%	61s	58.07%	202s
	CCL	**0.86%**	98s	12.50%	90s	**0.80%**	98s	14.20%	90s	2.19%	93s	16.48%	90s	20.00%	107s	112.22%	163s
	CoRE	1.93%	23s	**6.10%**	102s	1.57%	30s	**5.86%**	136s	**2.08%**	26s	**9.61%**	98s	**3.80%**	41s	**6.03%**	245s

Held-out closed-route variants

	Method	CVRPBPL				CVRPLTW				CVRPBPTW				CVRPBPLTW			
		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$	
		Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓
	PyVRP	0.00%	2.1h	0.00%	16m	0.00%	2.1h	0.00%	16m	0.00%	2.1h	0.00%	16m	0.00%	2.1h	0.00%	16m
	URS	13.84%	46s	30.48%	117s	4.90%	49s	11.38%	139s	12.68%	53s	28.73%	150s	12.74%	55s	28.46%	165s
	CoEKS	4.19%	51s	59.49%	157s	22.02%	61s	59.63%	191s	22.45%	62s	56.46%	188s	22.20%	61s	55.47%	188s
	MoSES	2.75%	66s	73.11%	176s	22.86%	78s	63.09%	202s	21.42%	81s	55.61%	200s	22.10%	78s	59.27%	203s
	RF-T	3.10%	38s	31.69%	151s	26.22%	46s	71.59%	180s	25.26%	47s	66.73%	182s	25.06%	46s	65.93%	180s
	RF-POMO	3.60%	38s	34.65%	154s	19.62%	46s	57.41%	179s	19.22%	46s	52.33%	178s	18.55%	46s	51.90%	179s
	RF-MoE	3.34%	52s	95.79%	180s	19.63%	64s	50.63%	196s	18.38%	65s	49.91%	197s	18.45%	63s	47.96%	196s
	MTPOMO	4.24%	38s	73.25%	149s	20.31%	46s	57.50%	178s	19.50%	46s	54.69%	178s	18.94%	45s	52.82%	177s
	MVMoE	3.75%	52s	44.10%	183s	25.62%	64s	57.38%	201s	24.11%	65s	52.67%	199s	23.85%	65s	51.40%	198s
	CCL	2.22%	99s	19.48%	91s	20.75%	115s	113.43%	162s	20.79%	118s	120.38%	160s	20.51%	116s	108.20%	160s
	CoRE	**2.09%**	34s	**11.09%**	130s	**3.81%**	47s	**6.31%**	274s	**6.14%**	50s	**10.39%**	260s	**6.40%**	51s	**9.96%**	287s

Held-out asymmetric variants

	Method	APCTSP				AMDCVRPBPTW				AOCVRPLTW				AMDOCVRPLTW			
		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$		$N = 100$		$N = 1000$	
		Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓
	PyVRP	0.00%†	2.1h	0.00%†	16m	0.00%	2.1h	0.00%	16m	0.00%	2.1h	0.00%	16m	0.00%	2.1h	0.00%	16m
	URS	45.47%	19s	45.84%	70s	27.74%	112s	20.29%	450s	21.67%	35s	21.94%	138s	31.89%	97s	25.83%	392s
	CoRE	**1.13%**	35s	**3.80%**	103s	**8.85%**	35s	**11.99%**	194s	**2.53%**	25s	**8.09%**	114s	**2.45%**	23s	**5.97%**	132s

route limits (L), time windows (TW), pickup–delivery, and prize collection with open routes, multiple depots, and asymmetric costs, for a total of 110 variants with up to four active constraints. See Appendix A for the formulations and Appendix D.1 for the split. Every constraint in the benchmark appears in training; of the 89 held-out variants, 49 combine constraints into sets observed in training and 40 into new sets. Each variant's reference is the best solution PyVRP (Wouda et al., 2024) finds in 60 s per instance, or OR-Tools (Furnon and Perron, 2024) where PyVRP does not model the composition.

Implementation details. We train CoRE at $N = 100$ using clipped PPO with the policy objective alone. We select the checkpoint with the lowest mean objective on fixed validation instances from the training variants and retain it for all benchmark evaluation; Appendix D.2 specifies the architecture, optimizer, and inference settings. All experiments are conducted on servers equipped with one NVIDIA A100 GPU and AMD EPYC 7J13 CPUs at 2.6 GHz.

Baselines. We include released attribute-conditioned generalists, namely MTPOMO (Liu et al., 2024), MVMoE (Zhou et al., 2024), RouteFinder with its Transformer, POMO, and MoE backbones (RF-T, RF-POMO, RF-MoE) (Berto et al., 2025), and URS (Zhou et al., 2026); expert-combination models, namely MoSES (Pan et al., 2025) and CoEKS (Yu et al., 2026); and the dynamically conditioned CCL (Gui et al., 2026). We rerun every released model on our test instances, and Appendix D.3 gives their settings. Since non-autoregressive methods release checkpoints for CVRP or CVRPTW alone, Appendix E.5 compares them with CoRE on these problems, where CoRE remains competitive. To isolate the learned scorer, we also run GCR with two fixed candidate costs: Distance GCR scores candidates by normalized travel distance, and Uniform GCR assigns every candidate zero cost. We further run CoRE-UDR and CoRE-CaDA, which replace the consequence vectors with the unified data representation (UDR) of URS and the constraint prompt of CaDA (Li et al., 2025), respectively. Both variants use the same simulator, refinement operators, and inference settings as CoRE (Appendix D.3).

Evaluation and metrics. At $N = 100$, every neural baseline evaluates 25,600 rollouts per instance and retains the best, which places its total inference time in the range of CoRE. We report the mean reference gap $100(\bar{f} - \bar{f}^{\text{ref}})/|\bar{f}^{\text{ref}}|$ and total inference time for all routing comparisons, reversing the sign for prize maximization so that negative gaps improve on the reference.

5.2 Zero-Shot Generalization Performance

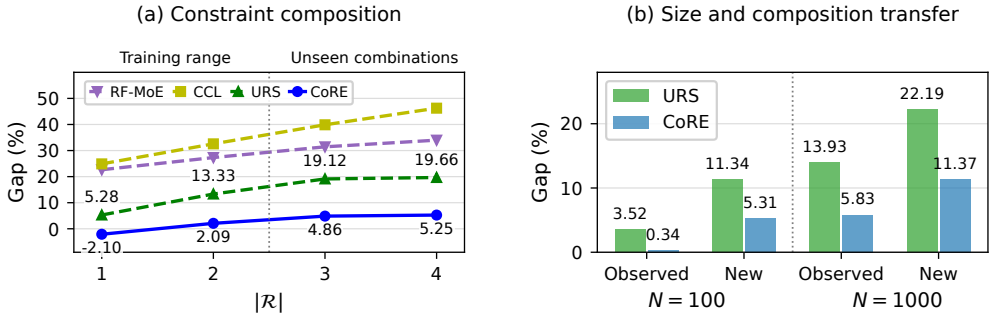


Figure 2: Composition and size transfer. Gaps averaged over (a) 48 common variants by constraint count and (b) the same 24 compositions at both sizes.

Generalization Across Different Constraint Compositions. We evaluate the selected CoRE checkpoint on all 110 variants, where it attains a 3.68% mean gap (Appendix E.1). As shown in Table 1, it outperforms every released baseline on six of the eight closed-route compositions, while most baselines, led by CCL, attain lower gaps on CVRP and CVRPL. With time windows active, it also improves on URS, the strongest baseline. Beyond the training range, with three and four active constraints, its gap is nearly four times lower than URS (Figure 2(a)). This pattern is consistent with the additive example of Appendix B.1: remaining load states the slack of capacity, whereas a mask reduces the slack of a time window to pass or fail. Against variants that differ only in their constraint representation, CoRE also attains the lowest gap (Section 5.4). Thus, CoRE retains the lowest gap as compositions grow beyond training. Appendix E.1 provides the composition breakdown.

Generalization to Large Scale. We next apply the $N = 100$ checkpoint to instances with $N = 1000$ without fine-tuning (Table 1 and Figure 2(b)). It retains the lowest gap on all eight closed-route compositions, 1.5× to 3.1× lower than that of URS. Across the 24 paired compositions, its gap stays at about half that of URS or below on both observed and new constraint sets.[1] These results suggest that CoRE retains its advantage when size and composition shifts compound. Appendix E.1 provides the additional quality and runtime results.

Generalization to New Constraints. Beyond recomposing trained constraints, we test four constraints absent from training. Battery replenishment (EVRP) and destination-dependent draft limits (VRPDL) carry a quantity against a bound, while cluster limits (VRPCL) and route incompatibility (VRPIC) track the clusters a route enters and a conflict relation among customers (Appendices A.4 and D.4).

Table 2: New constraints. Gaps to OR-Tools (600 s) of GCR models that differ only in their constraint representation, and total times, on 128 matched instances.

	OR-Tools (600 s)		Gap ↓			
Problem	**Obj. ↓**	**Time ↓**	**Distance**	**CoRE-UDR**	**CoRE**	**Time ↓**
EVRP	15.988	21.3h	0.26%	–0.38%	**–0.55%**	0.6m
VRPDL	17.421	21.3h	2.13%	0.97%	**0.41%**	0.5m
VRPCL	10.455	21.3h	0.49%	0.79%	**0.22%**	0.9m
VRPIC	17.316	21.3h	1.42%	1.40%	**0.06%**	1.0m

With every learned parameter fixed, CoRE outperforms OR-Tools at 600 s on EVRP in both quality and time (Table 2). On the other three constraints, it comes within 0.5% of OR-Tools, with gaps 2× to over 20× lower than those of Distance GCR. Among variants that differ only in their constraint representation, its mean gap is an order of magnitude below that of CoRE-UDR, demonstrating that transfer to new constraints rests on the shared consequence space. Appendix E.2 compares four consequence definitions for VRPIC.

[1] At $N = 1000$, we run CCL with test-time refinement probability 0.1, the setting that Gui et al. (2026) use on real-world instances with $N = 600$, since its default of 0.5 for $N = 100$ raises its gaps above 200% at this size.

Generalization Across Different Constraint Hardness. We further test feasible completion when several constraints are tight together. Since refinement accepts only feasible moves, feasibility is then decided at construction. Generalizing the hardness levels of PIP (Bi et al., 2024) to several numerical constraints, we jointly tighten capacity, route duration, time windows, and the fleet limit around planted feasible solutions in *Short-route packing* and *Long-route packing* (Appendix D.4). CoRE solves every Medium instance (Appendix E.3, Table 11). At Hard, it is infeasible 2.1× and 10× less often than URS on the two packings, while RF-MoE and CCL fail on nearly every instance. It also attains lower objectives than URS where both are feasible, and grade reconstruction further lowers the Hard infeasible rates. These results support consequence guidance for feasible completion under tight compositions.

Table 3: Compositional hardness (Hard). Objectives over feasible instances and infeasible rates on 128 instances; K: fleet limit.

	Short ($K = 14$)		**Long** ($K = 9$)	
Method	**Obj.** ↓	**Infsb%** ↓	**Obj.** ↓	**Infsb%** ↓
CoRE	15.273	45.31%	11.705	8.59%
+ $\mathcal{L}_{\text{rec}}$	15.222	**10.16%**	11.642	**0.78%**
URS	19.491	94.53%	15.224	85.94%
RF-MoE	—	100.00%	23.324	99.22%
CCL	—	100.00%	—	100.00%

5.3 Applicability to Other Paradigms and Problems

Autoregressive construction. We next verify whether the representation improves autoregressive construction without refinement. Table 4 compares POMO models trained on CVRPBLTW with the Lagrangian objective of Bi et al. (2026) against POMO* and CaR on 1,000 instances. At 50 customers, CoRE-POMO* lowers the gap of POMO* by 23.6×, and at 100 customers it doubles the improvement of POMO* over the reference. It also obtains a lower gap than CaR with remove-and-reinsertion refinement at both sizes, by construction alone and within a comparable total time. Appendix E.4 gives the complete comparison.

Node-oriented problems. To examine whether consequences extend beyond routing, we train two node policies separately. One shared model covers maximum independent set (MIS), maximum clique, minimum vertex cover (MVC), and maximum cut. One vertex-selection model covers exclusion (E), budgets (B), and group limits (Q), alone and as EB and EQ. As shown in Table 6, the shared model lowers the large-graph gap of the best specialist by at least 1.4× on MIS and MVC, while specialists lead on clique and cut. On the held-out BQ and EBQ, the vertex-selection model lowers the gap of multi-heuristic search by at least 1.3×, and CoRE-A attains similar gaps, since both inputs determine the same slack for these linear constraints (Appendix B.1). These results suggest that one representation serves new objectives and new compositions beyond routing. Appendix F gives more analyses.

Table 6: Node-oriented problems. Gaps (%) on the large graphs (ER-[700–800], RB-[800–1200], BA-[800–1200]) of Appendix F, specialists: (Sun and Yang, 2023; Li et al., 2024; Sanokowski et al., 2024; Wang and Li, 2023; Zhang et al., 2023); (b) uses a separately trained vertex-selection model on held-out compositions.

	(a) Shared policy	**ER-MIS**	**RB-MIS**	**RB-MVC**	**BA-MCut**
Specialist	GFlowNets	8.53%	13.14%	—	—
	DIFUSCO	14.08%	—	—	—
	Fast-T2T	9.51%	—	—	—
	DiffUCO	—	10.43%	—	**−1.69%**
	Meta-EGN	—	—	4.40%	—
	CoRE	**6.11%**	**6.06%**	**0.21%**	0.80%
	(b) Vertex selection	BQ_{32}	EBQ_{32}	BQ_{64}	EBQ_{64}
	Multi-heuristic	0.723%	0.719%	0.807%	1.467%
	CoRE-A	**0.487%**	0.467%	**0.559%**	0.929%
	CoRE	0.547%	**0.418%**	0.588%	**0.900%**

5.4 Analysis and Discussion

Ablation Study. Figure 3 and Table 5 compare CoRE with variants that replace its consequence vectors or its learned cost: CoRE-UDR and CoRE-CaDA read the attributes of URS and the constraint prompt of CaDA, CoRE-State reads the state entries alone, and Uniform and Distance GCR replace the learned cost with zero and travel distance (Appendix D.3). Results show that, within the same search, the learned cost lowers the mean gap of Distance GCR from 8.60% to 3.68% at $N = 100$ and from 11.14% to 7.50% at $N = 1000$, on training and held-out variants alike, and that CoRE surpasses both attribute representations on every population. On new constraint sets, CoRE consistently improves on CoRE-State and the attribute representations. Removing consequence guidance from CoRE-POMO* also raises its gap in every setting, by up to 20×. This confirms that the learned

Table 4: POMO* construction and CaR refinement. Gaps and total time on 1,000 CVRPBLTW instances, with size-matched CoRE-POMO* models; ⋆: Bi et al. (2026).

Method	$N = 50$		$N = 100$	
	Gap ↓	Time ↓	Gap ↓	Time ↓
OR-Tools (long)⋆	0.000%	102m	0.000%	210m
POMO*⋆	2.310%	2s	−1.645%	4s
MVMoE⋆	8.775%	3s	7.332%	10s
CaR (k-opt, $T_R = 20$)	2.116%	25.2s	−1.720%	73.7s
CaR (R&R, $T_R = 20$)	0.471%	29.7s	−2.448%	77.4s
CoRE-POMO*	**0.098%**	14.6s	**−3.324%**	104.6s

Table 5: Consequence guidance across solvers. Reference gaps (%).

GCR: Gap ↓ Constraints	Uniform GCR	Distance GCR	CoRE
No windows	0.48%	1.19%	**−1.45%**
Time windows	12.97%	9.63%	**6.27%**

POMO* on CVRPBLTW

N_{train}	N_{test}	POMO* Gap ↓	CoRE-POMO* Gap ↓
50	50	1.986%	**0.098%**
50	100	6.558%	**2.083%**
100	100	−1.645%	**−3.324%**

consequence cost is indispensable for both solvers, and that the grade and event entries carry transfer to new compositions. Appendix E.6 provides additional component ablations.

Effect of Grade Reconstruction. Grade reconstruction adds a dense slack target that favors feasible completion, while the policy objective fits the scorer to routing outcomes. However, due to this dense supervision, the scorer fits the training compositions more closely and transfers less to new ones. Figure 3 and Table 3 show that it lowers the training gap and both Hard infeasible rates, but raises the gap on new constraint sets. We therefore use the policy-only CoRE for transfer and retain $\mathcal{L}_{rec}$ where feasibility is the binding concern.

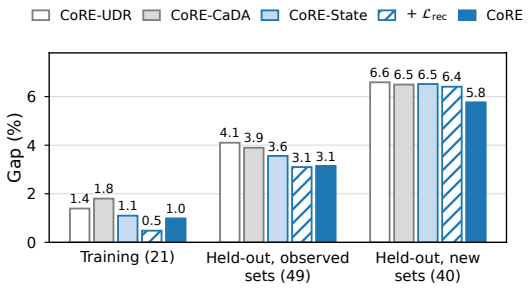


Figure 3: Conditioning and supervision. Gap (%) ↓

Why Do Consequences Help? Table 7 shows how often the candidates each model prefers remain feasible, for the POMO models on their training composition and for the non-autoregressive models on Short-route packing at Hard (Appendix E.7; see Appendix E.2 for a case study in which finer margins improve on the binary outcome for a relational constraint). CoRE-UDR prefers low-cost candidates but spends the slack that later customers need due to inputs that do not describe that slack, while POMO* receives the cost of that slack only through penalties after a violation. CoRE-POMO* and CoRE read that slack from the consequences of every active constraint, keeping more of their preferred candidates feasible and losing less to infeasible choices. Moreover, Appendix E.7 further shows that consequences keep new compositions within the range of the training inputs, with 3.5% of the inputs of CoRE outside that range against 80% for CoRE-UDR, for lower gaps on new constraint sets.

Table 7: Feasibility of preferred candidates. POMO on CVRPBLTW; GCR on Short-route packing at Hard. Loss: length lost to infeasible rollouts (POMO), customers unserved at the fleet limit (GCR).

Model	Preferred feasible ↑	Infeasibility loss ↓
POMO*	59.0%	1.31%
CoRE-POMO*	**83.4%**	**0.45%**
CoRE-UDR	2.3%	12.4
CoRE	**55.5%**	**2.1**

6 Conclusion

We introduced CoRE, a representation that places every constraint in a shared consequence space, making a learned routing heuristic reusable across constraints. By combining simulated per-constraint consequences with routing context, CoRE enables one pretrained model to handle held-out compositions, new constraints, and larger instances, pioneering the transfer to new routing constraints without fine-tuning. The representation also extends to separately trained autoregressive and node policies. One potential limitation is that each new constraint still requires its consequence definition, which follows directly from its transition and feasibility check but is not yet generated automatically for arbitrary problems. Future work includes extracting consequences from existing feasibility code, for example with large language models, unifying routing and node-oriented problems in one model, and integrating consequences into further autoregressive generalists.

AI USE STATEMENT

We disclose that LLMs were used solely to assist with grammar correction, phrasing, and stylistic editing. All the scientific hypotheses, experimental design, data analysis, and technical interpretations were conceived and conducted entirely by the authors. Every AI-suggested revision in the manuscript was carefully reviewed, and we take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of our results. Appendix A specifies the constraint transitions and consequence vectors, Appendix C both integrations, Appendix D.2 the training run, checkpoint selection, and implementations, and Appendices D.3 and D.4 the baselines and instance generators. We will release the source code and pretrained models with the paper.

REFERENCES

Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels Wouda, Leon Lan, Junyoung Park, Kevin Tierney, and Jinkyoo Park. RouteFinder: Towards foundation models for vehicle routing problems. *Transactions on Machine Learning Research*, 2025.

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.

Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, et al. The SCIP optimization suite 9.0. ZIB Report 24-02-29, Zuse Institute Berlin, 2024.

Xavier Bresson and Thomas Laurent. An experimental study of neural networks for variable graphs. In *International Conference on Learning Representations, Workshop Track*, 2018.

Jinho Choo, Yeong-Dae Kwon, Jihoon Kim, Jeongwoo Jae, André Hottung, Kevin Tierney, and Youngjune Gwon. Simulation-guided beam search for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2022.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 77416–77429, 2023. doi: 10.52202/075280-3385.

Shengyu Feng, Tarun Suresh, and Yiming Yang. Unsupervised diffusion solver for combinatorial optimization via combinatorial adjoint matching. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=y6LGSCnNOz.

Vincent Furnon and Laurent Perron. OR-Tools routing library, version 9.9, 2024. URL https://developers.google.com/optimization/routing/.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Shuangchun Gui, Suyu Liu, Xuehe Wang, and Zhiguang Cao. Chain-of-context learning: Dynamic constraint understanding for multi-task VRPs. In *International Conference on Learning Representations*, 2026.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2024. URL https://www.gurobi.com.

André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations*, 2022.

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.

Stefan Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30:113–148, 2008. doi: 10.1007/s00291-007-0083-6.

Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

Chaitanya K. Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, page 70–98, 2022.

Minsu Kim, Sanghyeok Choi, Hyeonah Kim, Jiwoo Son, Jinkyoo Park, and Yoshua Bengio. Ant colony sampling with GFlowNets for combinatorial optimization. In *International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pages 469–477, 2025.

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

Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 2022.

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

Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F. Werneck. Finding near-optimal independent sets at scale. *Journal of Heuristics*, 23(4):207–229, 2017. doi: 10.1007/s10732-017-9337-x.

Han Li, Fei Liu, Zhi Zheng, Yu Zhang, and Zhenkun Wang. CaDA: Cross-problem routing solver with constraint-aware dual-attention. In *International Conference on Machine Learning*, pages 35438–35456, 2025.

Sirui Li, Zhongxia Yan, and Cathy Wu. Learning to delegate for large-scale vehicle routing. In *Advances in Neural Information Processing Systems*, volume 34, pages 26198–26211, 2021.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast T2T: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 30179–30206, 2024. doi: 10.52202/079017-0950.

Fei Liu, Xi Lin, Zhenkun Wang, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. Multi-task learning for routing problem with cross-problem zero-shot generalization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1898–1908, 2024.

Suyu Liu, Zhiguang Cao, Nan Yin, and Yew-Soon Ong. Scale-Net: A hierarchical U-Net framework for cross-scale generalization in multi-task vehicle routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 14295–14303, 2026.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Advances in Neural Information Processing Systems*, volume 36, pages 8845–8864, 2023.

Fu Luo, Xi Lin, Mengyuan Zhong, Fei Liu, Zhenkun Wang, Jianyong Sun, and Qingfu Zhang. Learning to insert for constructive neural vehicle routing solver. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. COExpander: Adaptive solution expansion for combinatorial optimization. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 42130–42164, 2025a.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. ML4CO-bench-101: Benchmark machine learning for classic combinatorial problems on graphs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025b.

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*, volume 36, pages 49555–49578, 2023. doi: 10.52202/075280-2157.

Yimeng Min, Yiwei Bai, and Carla P. Gomes. Unsupervised learning for solving the travelling salesman problem. In *Advances in Neural Information Processing Systems*, volume 36, pages 47264–47278, 2023. doi: 10.52202/075280-2047.

Wenbin Ouyang, Sirui Li, Yining Ma, and Cathy Wu. Learning to segment for vehicle routing problems. In *International Conference on Learning Representations*, 2026.

Xuanhao Pan, Chenguang Wang, Chaolong Ying, Ye Xue, and Tianshu Yu. Beyond the heatmap: A rigorous evaluation of component impact in MCTS-based TSP solvers. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yuxin Pan, Zhiguang Cao, Chengyang Gu, Liu Liu, Peilin Zhao, Yize Chen, and Fangzhen Lin. Multi-task vehicle routing solver via mixture of specialized experts under state-decomposable MDP. In *Advances in Neural Information Processing Systems*, 2025.

Laurent Perron and Frédéric Didier. CP-SAT, OR-Tools version 9.9, 2024. URL https://developers.google.com/optimization/cp/cp_solver/.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183:483–523, 2020. doi: 10.1007/s10107-020-01523-z.

Jonathan Pirnay and Dominik G. Grimm. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *Transactions on Machine Learning Research*, 2024.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 35, pages 25531–25546, 2022. doi: 10.52202/068431-1851.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *International Conference on Machine Learning*, 2024.

Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500–520, 2014.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. arXiv:1707.06347. doi: 10.48550/arxiv.1707.06347.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 3706–3731, 2023. doi: 10.52202/075280-0164.

Jan Tönshoff, Berke Kisin, Jakob Lindner, and Martin Grohe. One model, any CSP: Graph neural networks as fast global search heuristics for constraint satisfaction. In *International Joint Conference on Artificial Intelligence*, pages 4280–4288, 2023. doi: 10.24963/ijcai.2023/476.

Dat Thanh Tran, Van Khu Vu, and Yining Ma. Beyond static priors: Dynamic neural guidance for large-scale ant colony optimization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026. doi: 10.1145/3770855.3817893.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013. doi: 10.1016/j.ejor.2013.02.053.

Haoyu Wang and Pan Li. Unsupervised learning for combinatorial optimization needs meta learning. In *International Conference on Learning Representations*, 2023.

Yu Wang, Yang Li, Jiale Ma, Junchi Yan, and Yi Chang. NExCO: Native solution expansion for diffusion-based combinatorial optimization. In *International Conference on Learning Representations*, 2026.

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

Yifan Xia, Xianliang Yang, Zichuan Liu, Zhihao Liu, Lei Song, and Jiang Bian. Position: Rethinking post-hoc search-based neural approaches for solving large-scale traveling salesman problems. In *Forty-first International Conference on Machine Learning*, 2024.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. NeuroLKH: Combining deep learning model with Lin–Kernighan–Helsgaun heuristic for solving the traveling salesman problem. In *Advances in Neural Information Processing Systems*, volume 34, pages 7472–7483, 2021a.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Step-wise deep learning models for solving routing problems. *IEEE Transactions on Industrial Informatics*, 17(7):4861–4871, 2021b. doi: 10.1109/tii.2020.3031409.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 43706–43728, 2023. doi: 10.52202/075280-1895.

Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 20284–20292. AAAI Press, 2024. doi: 10.1609/aaai.v38i18.30009.

Zikang Yu, Jinbiao Chen, and Jiahai Wang. Combination-of-experts with knowledge sharing for cross-task vehicle routing problems. In *International Conference on Learning Representations*, 2026.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ Salakhutdinov, and Alexander Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial optimization problems with GFlowNets. In *Advances in Neural Information Processing Systems*, volume 36, pages 11952–11969, 2023.

Ni Zhang and Zhiguang Cao. Hybrid-balance gflownet for solving vehicle routing problems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Zhi Zheng, Changliang Zhou, Xialiang Tong, Mingxuan Yuan, and Zhenkun Wang. UDC: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 37, pages 6081–6125, 2024. doi: 10.52202/079017-0197.

Changliang Zhou, Canhong Yu, Shunyu Yao, Xi Lin, Zhenkun Wang, Yu Zhou, and Qingfu Zhang. URS: A unified neural routing solver for cross-problem zero-shot generalization. In *International Conference on Machine Learning*, 2026.

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 *International Conference on Machine Learning*, pages 61804–61824, 2024.

Appendices

A Routing Problems and Consequence Calculations
- **A.1** Solution Structure
- **A.2** Consequence Projection
- **A.3** Benchmark Problems
- **A.4** New Constraints

B Information in Consequence Representations
- **B.1** Optimization Rationale for Consequence Representations
- **B.2** When One Decision Rule Suffices

C Consequence-Guided Solvers
- **C.1** Non-Autoregressive Model
- **C.2** Non-Autoregressive Training and Search
- **C.3** Autoregressive Construction

D Experimental and Implementation Details
- **D.1** Benchmark and References
- **D.2** Parameters and Training
- **D.3** Baselines
- **D.4** Instance Generation

E Additional Experiments and Analysis
- **E.1** Additional Benchmark Results
- **E.2** Case Study: Consequence Definitions for Route Incompatibility
- **E.3** Results under Compositional Hardness
- **E.4** Results on Autoregressive Construction
- **E.5** Comparison with Specialists and Post-Hoc Search
- **E.6** Component Ablations
- **E.7** Mechanism Analysis

F Consequence Guidance for Node Problems
- **F.1** Node Tasks and Models
- **F.2** Experimental Results

A Routing Problems and Consequence Calculations

This appendix defines the routing variants and the consequence projection that our simulator computes for each active constraint. We present the solution structures (Appendix A.1) and the projection shared by all constraints (Appendix A.2), followed by the benchmark constraints (Appendix A.3) and the new constraints (Appendix A.4).

Notation. Let $G = (V, E)$ be a complete graph on $V = \mathcal{D} \uplus C$, where $\mathcal{D}$ is the depot set and $C = \{x_1, \dots, x_n\}$ is the customer set. Each arc has a travel distance or time d_{uv} from x_u to x_v. A homogeneous fleet of capacity-Q vehicles is stationed at the depots. The objective is to find a solution $\mathcal{S} = \{R_1, \dots, R_K\}$ of routes $R_k = (v_0^k, v_1^k, \dots, v_{n_k}^k, v_{n_k+1}^k)$ that minimizes the total travel cost, subject to: (i) each route starts and ends at a depot, $v_0^k = v_{n_k+1}^k \in \mathcal{D}$, with $v_i^k \in C$ for $1 \le i \le n_k$; (ii) each customer is visited exactly once; and (iii) the feasibility conditions of the active constraint set $\mathcal{R}$ hold:

$$f(\mathcal{S}) = \sum_{k=1}^{K} \sum_{i=0}^{n_k} d_{v_i^k v_{i+1}^k}, \qquad C = \biguplus_{k=1}^{K} \{v_1^k, \dots, v_{n_k}^k\}. \tag{6}$$

Unless stated otherwise, variants use this objective. For $r \in \mathcal{R}$, let $x_r^{(i)}$ denote its state after v_i on the route under consideration.

A.1 Solution Structure

Following the benchmark of Zhou et al. (2026), three attributes modify the solution structure in Eq. (6). Open routes (O) drop the closing depot, giving $R_k = (v_0^k, \dots, v_{n_k}^k)$; the objective then sums to $i = n_k - 1$, and the projected return terms below are omitted. Multiple depots (MD) use $|\mathcal{D}| = 3$, following Berto et al. (2025); $v_0^k = v_{n_k+1}^k$ assigns each route to one depot, where its route-scoped states are released. Asymmetric costs (A) use $d_{uv} \neq d_{vu}$ on a directed metric closure. All other problems except TSP use $\mathcal{D} = \{x_0\}$, and the tour problems use $K = 1$.

A.2 Consequence Projection

Each consequence vector records normalized current and successor states, a constraint grade, and a transition event, defined below. An accumulating constraint r specifies an increment $\Delta_r(u, v)$, waiting map $J_{r,v}$, bound $[\ell_{r,v}, U_{r,v}]$ with possibly infinite endpoints, post-check increment $\pi_r(v)$, reset set D_r with values $c_r(v)$, and positive scale $\sigma_r(I)$. Given a candidate arc $e = (u, v)$ at full state $z = (s, (x_r)_{r\in\mathcal{R}})$, we compute provisional arrival $a_{r,v}$ and successor state x_r' as

$$a_{r,v} = J_{r,v}(x_r + \Delta_r(u, v)), \qquad x_r' = \begin{cases} c_r(v), & v \in D_r, \\ a_{r,v} + \pi_r(v), & \text{otherwise.} \end{cases} \tag{7}$$

Let $b_r = \min\{a_{r,v} - \ell_{r,v},\ U_{r,v} - a_{r,v}\}$ over the finite endpoints, let h_r be the same quantity evaluated after the projected return $v \to v_0$ when r carries a return guard, and let either equal $\sigma_r(I)$ when its check is inactive at this transition. With $d_r(z, e) = \mathbf{1}[v \in \mathcal{D}]$ marking a depot release, the unnormalized constraint grade is

$$\gamma_r(z, e) = \begin{cases} b_r, & d_r(z, e) = 1, \\ \min\{b_r, h_r\}, & \text{otherwise,} \end{cases} \qquad A_r(z, e) = d_r(z, e) \ \vee\ \mathbf{1}[\gamma_r(z, e) \ge 0]. \tag{8}$$

At its carrying node v, a state entering the consequence vector is normalized by

$$N_r(x) = \operatorname{clip}\left(\begin{cases} (x - \ell_{r,v})/(U_{r,v} - \ell_{r,v}), & \ell_{r,v}, U_{r,v} \text{ finite},\ \ell_{r,v} < U_{r,v}, \\ 1 - (x - \ell_{r,v})/\sigma_r(I), & \ell_{r,v} \text{ finite},\ U_{r,v} = +\infty, \\ x/\sigma_r(I), & \ell_{r,v} = -\infty,\ U_{r,v} \text{ finite}, \\ |x|/\sigma_r(I), & \ell_{r,v} = -\infty,\ U_{r,v} = +\infty, \end{cases} 0, 1 \right), \tag{9}$$

For a zero-width bound, use $|x - \ell_{r,v}|/\sigma_r(I)$. This gives $\widetilde{x}_r = N_r(x_r)$ and $\widetilde{x}'_{r,e} = N_r(x'_r)$; if a constraint maintains several components, $\widetilde{x}_r$ uses the component on which its bound is stated. Eqs. (8) and (9) share one numerical boundary tolerance, so the check and grade agree in sign at a bound. The release term d_r admits depot moves without a bound check; on closed routes the return guard h_r already

checks every return, and on open routes the release keeps a route-ending depot move feasible. We normalize and clip the raw grade of an accumulating constraint as

$$g_{r,e} = \mathrm{clip}(\gamma_r(z,e)/\sigma_r(I), -\mathbf{1}[A_r(z,e)=0], 1). \tag{10}$$

A service-order constraint r is given by a relation $\sim_r$ over C and a counter c_r of outstanding obligations or, for a class order, a phase indicator, with a positive scale m_r that each constraint below specifies: the number of pairs for a matching and the number of backhaul customers for the class order. Its state entries are $\widetilde{x}_r = \mathrm{clip}(c_r/m_r, 0, 1)$ and $\widetilde{x}'_{r,e} = \mathrm{clip}(c'_r/m_r, 0, 1)$, and its grade is

$$g_{r,e} = \begin{cases} 1 - \mathrm{clip}(c'_r/m_r, 0, 1), & A_r(z,e) = 1, \\ -1, & A_r(z,e) = 0, \end{cases} \tag{11}$$

so its sign reproduces the relation check while its magnitude grades the residual obligations. Both families construct the same vector

$$\mathbf{u}_r(e) = (\widetilde{x}_r,\ \widetilde{x}'_{r,e},\ g_{r,e},\ q_{r,e}), \qquad q_{r,e} = \mathbf{1}[v \in D_r], \tag{12}$$

where service-order constraints set $q_{r,e} = 0$. Both integrations compute this vector from the same transitions, checks, and reset sets D_r, and Appendix C specifies how each computes it during search, and the problem definitions below supply the parameters of both projections.

A.3 Benchmark Problems

The formulations below follow the routing variants evaluated by Zhou et al. (2026); we state the transitions and feasibility checks used by our simulator.

Capacity (CVRP). For each customer $x_i \in C$, let $q_i > 0$ denote its demand, and let the total demand of each route not exceed Q. Along a route $R = (v_0, \ldots, v_{n_R+1})$ carrying n_R customers, the load still carried satisfies

$$x^{(0)} = Q, \qquad x^{(i)} = x^{(i-1)} - q_{v_i}, \qquad 0 \le x^{(i)} \le Q \quad (1 \le i \le n_R), \tag{13}$$

which for nonnegative demands is $\sum_{i=1}^{n_R} q_{v_i} \le Q$. Capacity takes $\Delta_r(u,v) = -q_v$, $J_{r,v} = \mathrm{id}$, $\pi_r = 0$, $[\ell_{r,v}, U_{r,v}] = [0, Q]$, $D_r = \mathcal{D}$ with $c_r(v) = Q$, $\sigma_r(I) = Q$, and no return guard, so that

$$a_{r,v} = x_r - q_v, \qquad \gamma_r(z,e) = \min\{a_{r,v},\ Q - a_{r,v}\}, \qquad \widetilde{x}_r = \mathrm{clip}(x_r/Q, 0, 1). \tag{14}$$

Backhauls (VRPB). Following the signed-demand convention of Liu et al. (2024), denote the linehaul and backhaul customer sets by C^+ and C^-, respectively, so that $C = C^+ \uplus C^-$. Demand is positive for $x_i \in C^+$ and negative for $x_i \in C^-$; the objective remains $\min_{\mathcal{S}} f(\mathcal{S})$. Eq. (13) holds with signed q_{v_i} and, writing C_R^+ for the linehaul customers still unserved when R opens, the opening load

$$x^{(0)} = \begin{cases} 0, & v_1 \in C^- \text{ under BP}, \\ Q \cdot \mathbf{1}[C_R^+ \ne \emptyset], & \text{otherwise}. \end{cases} \tag{15}$$

The mixed variant (B) admits either class at any position in a route. For the ordered variant (BP), let $\kappa(v) = \mathbf{1}[v \in C^-]$; its additional condition is

$$\kappa(v_1) \le \kappa(v_2) \le \cdots \le \kappa(v_{n_R}), \tag{16}$$

so every linehaul customer of a route precedes every backhaul customer of that route. Capacity projects as in Eq. (13) with signed q_v and the reset value of Eq. (15). The ordering constraint of Eq. (16) is a service-order constraint with class order $\sim_r$, counter $c_r = \kappa(u)$, $m_r = |C^-|$, and $A_r(z,e) = \mathbf{1}[\kappa(v) \ge c_r]$, graded by Eq. (11). Here c_r is a phase indicator rather than a progress count: it records whether the route has entered its backhaul phase, so the state entries take the values 0 and $1/|C^-|$ and feasible grades are 1 or $1 - 1/|C^-|$, while the check itself depends only on the phase.

Route duration limits (L). Following the duration-limit variant of Liu et al. (2024), let L be the maximum accumulated travel of a route. Along $R = (v_0, \ldots, v_{n_R+1})$, this condition is

$$x^{(0)} = 0, \qquad x^{(i)} = x^{(i-1)} + d_{v_{i-1}v_i} \le L \quad (1 \le i \le n_R), \qquad x^{(n_R)} + d_{v_{n_R}v_0} \le L, \tag{17}$$

where the closing term is present for closed routes only. The route limit takes $\Delta_r(u,v) = d_{uv}$, $J_{r,v} = \mathrm{id}$, $\pi_r = 0$, $[\ell_{r,v}, U_{r,v}] = (-\infty, L]$, $D_r = \mathcal{D}$ with $c_r(v) = 0$, and $\sigma_r(I) = L$, with the return guard $h_r = L - (a_{r,v} + d_{vv_0})$ of Eq. (8).

Time windows (TW). Following the benchmark time-window setup of Zhou et al. (2024), let $[a_i^-, a_i^+]$ be the window and s_i the service time of each customer x_i; each depot has horizon $[0, a_{v_0}^+]$. A vehicle leaves its depot at time zero and waits if it reaches a customer before a_i^-. With $s_{v_0} = 0$, its service start $\tau^{(i)}$ satisfies

$$\tau^{(0)} = 0, \qquad \tau^{(i)} = \max\{\tau^{(i-1)} + s_{v_{i-1}} + d_{v_{i-1}v_i},\ a_{v_i}^-\} \le a_{v_i}^+, \qquad \tau^{(n_R)} + s_{v_{n_R}} + d_{v_{n_R}v_0} \le a_{v_0}^+. \tag{18}$$

The window constraint takes $\Delta_r(u, v) = d_{uv}$, $J_{r,v}(a) = \max\{a, a_v^-\}$, $\pi_r(v) = s_v$, $[\ell_{r,v}, U_{r,v}] = (-\infty, a_v^+]$, $D_r = \mathcal{D}$ with $c_r(v) = 0$, and $\sigma_r(I)$ the instance horizon, with the return guard $h_r = a_{v_0}^+ - (a_{r,v} + s_v + d_{vv_0})$. The deadline is therefore tested at the service start $a_{r,v}$ and the service time enters the successor.

Pickup and delivery (PD). Following the paired-node construction used by Zhou et al. (2026), denote the $n/2$ pickup–delivery pairs by $\mathcal{P} = \{(p_j, \delta_j)\}$, where p_j is a pickup, δ_j is its delivery, and $q_{p_j} = -q_{\delta_j}$. For every pair $(p, \delta) \in \mathcal{P}$, one route R_k must contain both nodes at indices $i < j$, with $v_i^k = p$ and $v_j^k = \delta$. The pairing is a service-order constraint with matching $\sim_r = \mathcal{P}$, $m_r = |\mathcal{P}|$, and counter c_r equal to the number of pairs opened on the current route and not yet closed, so that

$$A_r(z, e) = \begin{cases} \mathbf{1}[c_r = 0], & v \in \mathcal{D}, \\ 1, & v = p_j \text{ for some } j, \\ \mathbf{1}[p_j \text{ served on the current route}], & v = \delta_j, \end{cases} \tag{19}$$

graded by Eq. (11). PDTSP activates this constraint alone on a single route; PDCVRP composes it with capacity over K routes.

Traveling salesman problem (TSP). $\mathcal{D} = \emptyset$ and $K = 1$, so a solution is a single cycle on C and the problem is $\min_S f(S)$, with $d_{uv} \neq d_{vu}$ in the asymmetric variant. Feasibility is the visit-once condition of Eq. (6) alone, enforced by the visited-node mask. $\mathcal{R} = \emptyset$, so no consequence vectors are computed and the scorer conditions on the objective features alone.

Orienteering (OP). Following the OP formulation in Zhou et al. (2026), let $p_i \ge 0$ denote the prize of each node. A single tour ($K = 1$) starts and ends at x_0 and may omit customers, so the visit-once condition becomes $\{v_1, \dots, v_{n_R}\} \subseteq C$. The objective is to maximize collected prize, subject to the tour limit:

$$\max_S \sum_{i=1}^{n_R} p_{v_i} \qquad \text{subject to} \qquad \sum_{i=0}^{n_R} d_{v_i v_{i+1}} \le L. \tag{20}$$

The tour limit projects exactly as the route limit of Eq. (17), with $\sigma_r(I) = L$ and the return guard included.

Prize collection (PCTSP). Following the prize-collecting setup in Zhou et al. (2026), let $p_i \ge 0$ denote each node's prize and $\beta_i \ge 0$ its penalty. A single tour starts and ends at x_0 and may omit customers. Denote its visited customers by $\mathcal{V}(S)$; the objective and quota are

$$\min_S\ f(S) + \sum_{v \in C \setminus \mathcal{V}(S)} \beta_v \qquad \text{subject to} \qquad \sum_{v \in \mathcal{V}(S)} p_v\ \ge\ p^{\min}. \tag{21}$$

The quota is a solution-scoped accumulation with $\Delta_r(u, v) = p_v$, $[\ell_{r,v}, U_{r,v}] = [p^{\min}, +\infty)$, $D_r = \emptyset$, and $\sigma_r(I) = p^{\min}$, whose check is inactive until completion:

$$\gamma_r(z, e) = \begin{cases} x'_r - p^{\min}, & e \text{ completes the tour}, \\ \sigma_r(I), & \text{otherwise}. \end{cases} \tag{22}$$

SPCTSP uses the same formulation on the benchmark's stochastic-prize instances.

Composed variants. Benchmark names combine the definitions above: prefixes A, MD, and O select the solution structures of Appendix A.1, while suffixes B, BP, L, and TW add constraints to the capacitated base. Pickup–delivery and tour problems form additional families. For example, OCVRPBLTW uses open routes and activates signed capacity, the route limit, and the time window. For each active constraint in a composed variant, we construct one vector of Eq. (12).

A.4 New Constraints

The four new constraints below are absent from training. Each is supplied to the simulator as its transition and feasibility check, as any constructive solver requires, and the simulator computes

the vector of Eq. (12) from them without adding learned parameters. Battery replenishment and draft limits use route-scoped numerical states and instantiate Eq. (8) directly: the state entries normalize the checked quantity by the constraint's own bound, the grade records the signed slack of the same comparison, and the event marks a reset. The cluster limit uses a set-valued state and route incompatibility a relation over unserved customers, so their vectors are defined below; for both, a depot candidate closes the route and opens the next one on an empty state, giving $\widetilde{x}'_{r,e} = 0$, $g_{r,e} = 1$, and $q_{r,e} = 1$, while $q_{r,e} = 0$ at a customer. Appendix D.4 specifies the test instances.

Battery (EVRP). Let B denote battery capacity and $\mathcal{B} \subseteq C$ the charger set. Arcs consume charge at the travel rate, while chargers and depots replenish it (Schneider et al., 2014). Along $R = (v_0, \dots, v_{n_R+1})$, the charge transition is

$$x^{(0)} = B, \qquad a^{(i)} = x^{(i-1)} - d_{v_{i-1}v_i} \;\geq\; 0, \qquad x^{(i)} = \begin{cases} B, & v_i \in \mathcal{D} \cup \mathcal{B}, \\ a^{(i)}, & \text{otherwise}, \end{cases} \tag{23}$$

so the charge is tested on arrival, before any replenishment. The constraint takes $\Delta_r(u, v) = -d_{uv}$, $[\ell_{r,v}, U_{r,v}] = [0, +\infty)$, $D_r = \mathcal{D} \cup \mathcal{B}$ with $c_r(v) = B$, $\sigma_r(I) = B$, and the construction return guard $h_r = a_{r,v} - d_{vv_0}$. Two candidates can replenish to the same successor $x'_r = B$ yet receive different grades $g_{r,e}$ because their arrival charges $a_{r,v}$ differ.

Draft limits (VRPDL). For each customer x_i, let dl_i be the destination draft limit on the signed load brought to it, in addition to capacity (Bi et al., 2026). Along a route, the draft state satisfies

$$x^{(i)} = x^{(i-1)} + q_{v_i} \;\leq\; \mathrm{dl}_{v_i}, \qquad x^{(0)} = 0, \tag{24}$$

the route-scoped signed extension of the TSPDL accumulation of Bi et al. (2026): positive demand raises the draft and negative demand lowers it, while capacity uses $x^{(i)} = x^{(i-1)} - q_{v_i}$ as in Eq. (13). The evaluated VRPDL instances have nonnegative demands, so the draft state stays nonnegative and its normalization in Eq. (9) involves no clipping at zero. The constraint takes $\Delta_r(u, v) = q_v$, $[\ell_{r,v}, U_{r,v}] = (-\infty, \mathrm{dl}_v]$, $D_r = \mathcal{D}$ with $c_r(v) = 0$, $\sigma_r(I) = 1$, and no return guard.

Cluster limit (VRPCL). Let $\mathcal{G} = \{G_1, \dots, G_8\}$ be the customer groups obtained by k-means on their coordinates. A route may enter at most $c = 2$ groups. Since no capacity bound is active, this limit alone determines how many routes a solution uses. For every route R,

$$\big|\{\, g : G_g \cap R \neq \emptyset \,\}\big| \;\leq\; c. \tag{25}$$

The state of an open route is the set $T \subseteq \{1, \dots, |\mathcal{G}|\}$ of groups it has entered, so the increment of a candidate depends on whether its own group already lies in T. With $\mathrm{cl}(v)$ the group containing v and $T' = T \cup \{\mathrm{cl}(v)\}$,

$$\widetilde{x}_r = \mathrm{clip}(|T|/c, 0, 1), \quad \widetilde{x}'_{r,e} = \mathrm{clip}(|T'|/c, 0, 1), \quad A_r(z, e) = \mathbf{1}[|T'| \leq c], \quad g_{r,e} = \mathrm{clip}\Big(\tfrac{c-|T'|}{c}, -1, 1\Big). \tag{26}$$

Since entering a new group consumes one available slot, two feasible customers can receive different grades when only one belongs to a group already counted by T. Because the limit checks the number of entered groups against c, this vector follows the numerical-bound rule of Eq. (8).

Route incompatibility (VRPIC). Let $H = (C, E_H)$ be the conflict graph joining each customer to its three nearest neighbors. Incompatible customers cannot share a route, and no capacity bound is active. For every route R and every pair $u, v \in R$,

$$\{u, v\} \notin E_H. \tag{27}$$

Let S be the customers an open route has served, U the unserved customers, and $F = \{w \in U : N_H(w) \cap S = \emptyset\}$ the compatible frontier of the open route. For candidate v, write $F_v = F \setminus (\{v\} \cup N_H(v))$ for the frontier surviving the choice of v. With $|F|$ the size of the current compatible frontier,

$$\widetilde{x}_r = 0, \qquad \widetilde{x}'_{r,e} = 1 - \frac{|F_v|}{\max(|F|,\, 1)}, \qquad A_r(z, e) = \mathbf{1}[N_H(v) \cap S = \emptyset], \tag{28}$$

$$g_{r,e} = \begin{cases} |F_v| / \max(|F|,\, 1), & A_r(z, e) = 1, \\ -\min\{1,\, |N_H(v) \cap S| / \deg_H(v)\}, & A_r(z, e) = 0. \end{cases} \tag{29}$$

The current entry sets the baseline for candidate-relative frontier consumption to zero, while the successor entry records the fraction of F removed by the candidate. For a feasible insertion, the grade is the fraction that survives, as exclusion counts it in Eq. (50), so two candidates passing the same binary check can receive different grades. For a rejected insertion, the grade is the negative fraction of v's neighbors already served on this route. Appendix E.2 compares it with three alternative definitions, including a vector read off the binary check alone.

B Information in Consequence Representations

We first relate the consequence entries to the residual feasible sets and resource-label dominance of classical routing (Appendix B.1), and then characterize when a decision rule observing the complete model input can select an optimal action (Appendix B.2).

B.1 Optimization Rationale for Consequence Representations

Candidate decisions and residual feasible sets. Given full state z and a candidate e at a sequential construction step, denote the feasible complete continuations by $\mathcal{F}_{\mathcal{R}}(z, e)$, where each continuation satisfies the active constraints. For an additive minimization objective, we compute the optimal action value as

$$Q^{\star}_{\mathcal{R}}(z, e) = c_I(z, e) + \inf_{\omega \in \mathcal{F}_{\mathcal{R}}(z,e)} C_I(\omega \mid z, e), \tag{30}$$

where c_I is immediate cost, C_I is continuation cost, and the infimum over an empty set is $+\infty$. If two fixed-cost state–candidate pairs have equal residual feasible sets, Eq. (30) assigns them equal values. The consequence vector summarizes each candidate's resulting state locally and records the grades of the applicable checks. The example below shows how these entries can distinguish residual restrictions.

Resource labels and dominance. Pricing subproblems in branch-and-price routing commonly use resource-constrained shortest paths (Irnich and Desaulniers, 2005). Given a partial path, a label stores its reduced cost, resource states, and relevant history; resource extension functions update these states after an arc extension (Irnich, 2008). The updated states determine which subsequent extensions remain feasible. Denote the feasible continuations of partial path P by $\mathcal{E}(P)$, and the reduced cost of its completion with ω by $\mathrm{rc}(P \circ \omega)$. For paths P_1, P_2 ending at the same node, the conditions

$$\begin{aligned} &\mathcal{E}(P_2) \subseteq \mathcal{E}(P_1), \\ &\mathrm{rc}(P_1 \circ \omega) \le \mathrm{rc}(P_2 \circ \omega) \quad \text{for every } \omega \in \mathcal{E}(P_2) \end{aligned} \tag{31}$$

permit discarding P_2 because each of its completions has a feasible replacement of no greater reduced cost. This is the same-completion dominance condition described by Pessoa et al. (2020). Classical labeling uses the post-extension states to establish such inclusions; CoRE supplies normalized post-extension states and grades to a learned scorer.

An additive-budget example. Consider a non-restoring resource with capacity $B > 0$ and remaining allowance $0 < b \le B$. For each remaining item i, let $d_i \ge 0$ be its fixed consumption and permit at most one selection. Denote the individually feasible items, for which $d_i \le b$, by J. The unclipped current and successor allowances are

$$\widetilde{x} = \frac{b}{B}, \qquad \widetilde{x}'_i = \frac{b - d_i}{B}, \qquad i \in J.$$

The item-indexed consequence table then gives

$$p_i = \frac{\widetilde{x} - \widetilde{x}'_i}{\widetilde{x}} = \frac{d_i}{b},$$

so a future subset $S \subseteq J$ satisfies the remaining budget exactly when

$$\sum_{i \in S} d_i \le b \quad \Longleftrightarrow \quad \sum_{i \in S} p_i \le 1. \tag{32}$$

After selecting item i with $p_i < 1$, we update the remaining coefficients to $p_j^{\mathrm{new}} = p_j/(1 - p_i)$; when $p_i = 1$, only zero-consumption items remain feasible. For $b = 6$, the pairs $(4, 4)$ and $(2, 2)$ have the same individual feasibility mask, whereas only the second pair can be selected together. Their consequence tables distinguish the residual budget restrictions left unresolved by that mask.

In this non-restoring linear case, $(b/B, d_i/B)$ and $(\widetilde{x}, \widetilde{x}'_i)$ are mutually recoverable by subtraction, so the consequence table and state-aware attributes determine the same budget restriction. Routing constraints can retain history or internal state beyond the summaries in Appendix A.2. We therefore evaluate decisions from the complete scorer observation, including routing and objective inputs outside the consequence vectors.

B.2 When One Decision Rule Suffices

From local consequences to decision information. States with the same complete model observation can assign different values to available actions. If their optimal-action sets share a member, one observation-based rule can select an optimal action across those states; if the intersection is empty, no such rule can. Exact action-value recovery additionally requires each available action to have the same value throughout the observation class.

Denote the complete observation of a decision rule at state z by $\mathsf{O}(z)$, and the common available action set at observation o by $\mathcal{D}(o)$. The observation includes the indexed consequence collection, graph and geometry, routing and objective features, the action mask and decoder indicators, and any retained or cached information used by the rule. States with the same observation have aligned candidate identifiers, the same available actions, and equal remaining horizon, although their unobserved constraint dynamics may differ. Since the rule uses only $o = \mathsf{O}(z)$, any relation supplied elsewhere to the model belongs in this observation. Let $Q^\star(z, e)$ be the optimal continuation value defined in Eq. (30).

Remark B.1 (Information needed for values and decisions). For an observation o, let $\Omega_o = \{z : \mathsf{O}(z) = o\}$, and assume each state under consideration has a finite optimal value and the common action set $\mathcal{D}(o)$ is finite. Exact action values can be recovered by a function $q(o, e)$ if and only if $Q^\star(z, e)$ is constant over $z \in \Omega_o$ for every available action e. An observation-based deterministic rule can choose an optimal action at every such state if and only if

$$\bigcap_{z \in \Omega_o} \arg\min_{e \in \mathcal{D}(o)} Q^\star(z, e) \neq \varnothing \quad \text{for every } o. \tag{33}$$

A randomized rule is optimal at every such state exactly when its support lies in this intersection. Actions with no feasible completion may have value $+\infty$.

Proof. We first consider exact values. Since a function of o takes the same value at every state in Ω_o, constancy of $Q^\star(z, e)$ is necessary; conversely, that constancy defines $q(o, e)$ by choosing any representative state. Next, an observation-based deterministic rule must choose the same action throughout Ω_o, so its choice must belong to every state's optimal-action set. When the intersection is nonempty, choosing a member with a fixed tie-breaking order constructs such a rule. Finally, every action has value at least the finite state optimum, so a randomized rule attains that optimum exactly when it assigns probability only to optimal actions at that state. Requiring this at every state gives the same intersection condition. □

Values can differ while the best decision is unchanged. Adding a state-dependent constant to every action value changes $Q^\star$ while preserving all minimizing actions. Thus, states can satisfy the decision condition of Eq. (33) even when exact values differ. The following construction shows that this condition cannot be weakened.

Proposition B.1 (Tightness of the decision condition)**.** *For every $M > 0$, two feasible two-decision problems can have identical initial complete model observations and compatible local consequence grades, but optimal action values* $(1, 1 + M)$ *and* $(1 + M, 1)$*. Every observation-based initial decision rule then has expected excess cost at least $M/2$ on one problem. Both optimal costs are* 1.

Proof. Consider two problems, denoted + and −, with initial actions a and b of zero immediate cost. At the second decision, the candidate actions are L and H, with costs 1 and $1 + M$ in both problems. In problem +, the constraint permits L after a and H after b, while in problem − it permits H after a and L after b. Both initial actions pass the local check. The full constraint state stores the first choice, but the relation distinguishing the two problems is absent from the initial observation.

Let the state summary be the fraction of the two commitments already made, and let the grade of a feasible transition be the successor fraction. Every initial candidate then has consequence vector

$$(\widetilde{x}, \widetilde{x}', g, q) = (0, \tfrac{1}{2}, \tfrac{1}{2}, 0),$$

computed only by incrementing the commitment count. All other observed graph, geometry, routing, objective, candidate, mask, and cache information is identical across the two problems. At the second decision, the simulator checks the stored first choice and the candidate relation, assigning grade 1

to feasible actions and -1 to infeasible actions. Thus, the local grades remain compatible with the checks even though the initial observation omits the relation that determines the later choice.

The two initial action-value vectors are therefore $(1, 1 + M)$ and $(1 + M, 1)$. If the common initial rule selects a with probability p, its excess costs are $M(1 - p)$ in problem + and Mp in problem $-$. Consequently,

$$\max\{M(1 - p), Mp\} \geq \frac{M}{2}.$$

Both problems have optimal cost 1, and $M > 0$ is arbitrary, which gives the claimed decision-loss bound. □

Implications for neural reuse. Compatible consequence calculations preserve the scorer's input roles and parameterization, and a relation that the initial observation omits, as in Proposition B.1, is restored by supplying it to the model. Table 7 shows that both the non-autoregressive model and POMO learn to keep their preferred candidates feasible from consequence inputs, which accounts for the lower gaps in Sections 5.2 and 5.3. CoRE uses projected consequences to rank candidates, while full-state checks and completed-solution validation determine feasibility. Appendix C specifies the non-autoregressive and autoregressive implementations.

C Consequence-Guided Solvers

Given candidate consequences and routing features, the non-autoregressive model computes arc costs for construction and refinement. Construction reads each cost directly; refinement evaluates the cost change from removed and inserted arcs. The autoregressive model adds pooled consequence context and candidate biases to a POMO decoder. Every model reads the vector of Eq. (12) for each candidate and constraint and encodes it with one network shared across constraints, while the aggregation over candidates and the entry point of the score follow the backbone. Appendix D.2 gives the numerical settings.

C.1 Non-Autoregressive Model

Given a candidate graph with consequence vectors and routing features, one encoder pass computes a cost for every candidate arc. First, shared consequence encoders produce node and arc inputs. Next, the graph network processes these inputs. Finally, the cost heads read the resulting embeddings.

Consequence encoding. At each neural evaluation, we replay the incumbent $\mathcal{S}_t$ up to the tail u of each candidate arc e to obtain the first three entries of Eq. (12). Since arcs whose tail lies off $\mathcal{S}_t$ have no prior state to replay, the scorer also reads a known-state indicator $v_{r,e} = \mathbf{1}[u \in \mathcal{S}_t]$, which separates a zeroed entry from a true zero state. Since the state entry is a per-node quantity of the incumbent, a shared MLP encodes it once per node i rather than once per arc, for every active constraint. For each arc e, another shared MLP encodes the successor summary, constraint grade, and event, together with the known-state indicator, as

$$\mathbf{a}_{i,r} = \phi^x_\theta(\widetilde{x}_{i,r}), \qquad \mathbf{b}_{e,r} = \phi^T_\theta(\widetilde{x}_{r,e}, g_{r,e}, v_{r,e}, q_{r,e}). \tag{34}$$

For arc $e = (i, j)$, $\mathbf{b}_{e,r}$ encodes the successor, grade, and event of $\mathbf{u}_r(e)$, the state entries, encoded once per node, enter the graph network with its node inputs, and the readout also retains $\mathbf{a}_{j,r}$, the incumbent state at the head.

Given the embeddings $\mathbf{a}_{i,r}$ and $\mathbf{b}_{e,r}$, we form fixed-width node and edge summaries by summing over active constraints and dividing by the square root of the active-set size. The encoders also receive that size as $|\mathcal{R}|/(1 + |\mathcal{R}|)$, while the cost heads retain the individual embeddings. Given the objective coefficients $\mathbf{o}_I$ and objective scale $\bar{\sigma}_I$ of the instance, we compute the context shared by every active constraint as

$$\mathbf{h}_\star = \Psi_\theta(\mathbf{o}_I, \bar{\sigma}_I). \tag{35}$$

Candidate graph and routing features. Let $G_K = (V, E_K)$ be the candidate graph, initially connecting each node to its K nearest candidates under normalized travel cost. If no feasible action is available, we expand this graph. Node features include position, depot status, incoming and outgoing travel summaries, incumbent route position, and endpoint distances. Arc features include travel and

objective costs and incumbent and reverse-incumbent indicators; simulated consequences supply the numerical constraint inputs.

Given the pooled consequences and routing inputs, a residual gated graph ConvNet (Bresson and Laurent, 2018; Joshi et al., 2022) of depth L_E and width d computes edge embeddings $\mathbf{h}_e$, whose mean $\mathbf{h}_G = |E_K|^{-1} \sum_{e \in E_K} \mathbf{h}_e$ gives the graph embedding:

$$\{\mathbf{h}_e\}_{e \in E_K} = \mathrm{GNN}_\theta(G_K, \mathrm{Pool}_r\, \mathbf{a}_{i,r}, \mathrm{Pool}_r\, \mathbf{b}_{e,r}), \tag{36}$$

Routing and objective inputs are implicit in this expression.

Cost and state weighting. For $e = (i, j)$, the interaction map combines its graph embedding, the shared context, and the retained node and transition embeddings:

$$\mathbf{c}_{e,r} = \tanh\Big(\mathrm{LN}(W_e \mathbf{h}_e) + W_h \mathbf{h}_\star + W_u[\mathbf{a}_{j,r}, \mathbf{b}_{e,r}, q_{r,e}]\Big). \tag{37}$$

For each candidate–constraint pair, two shared linear heads with tanh outputs predict the cost contribution and auxiliary grade estimate as

$$\epsilon_{\theta,r}(e) = \tanh(w_{\mathrm{cost}}^\top \mathbf{c}_{e,r}), \qquad \widehat{g}_{\theta,r}(e) = \tanh(w_{\mathrm{cons}}^\top \mathbf{c}_{e,r}). \tag{38}$$

A shared encoder maps the active indicator and the objective context of Eq. (35) to $\bar{\mathbf{t}}$, and a residual layer gives the constraint token $\mathbf{t} = \bar{\mathbf{t}} + \Gamma_\theta(\bar{\mathbf{t}})$. Given $\mathbf{s} = \tanh(\mathrm{LN}(W_g \mathbf{h}_G) + \mathbf{t})$, linear heads compute a base intensity $\lambda_\theta^{\mathrm{base}} = \mathrm{softplus}(w_m^\top \mathbf{s})\, \sigma(w_{\mathrm{bind}}^\top \mathbf{s})$, a bias $b = w_b^\top \mathbf{s}$, and a coupling weight $w = \langle q_\theta(\mathbf{s}), k_\theta(\mathbf{t}) \rangle / \sqrt{d}$. The weight of Eq. (5) then reads the joint current state of all active constraints:

$$\lambda_\theta(z_t) = \lambda_\theta^{\mathrm{base}}\, 2\sigma\Big(b + w \sum_{s \in \mathcal{R}} \widetilde{x}_s\Big). \tag{39}$$

The constraint cost weights the contributions by λ_θ and adds the objective cost:

$$E_{\mathrm{cons},\theta}(e) = \lambda_\theta(z_t) \sum_{r \in \mathcal{R}} \epsilon_{\theta,r}(e), \qquad \widehat{c}_\theta(e) = E_{\mathrm{obj}}(e) + E_{\mathrm{cons},\theta}(e). \tag{40}$$

where $E_{\mathrm{obj}}(e)$ is the normalized objective cost of arc e. In the common policy of Eq. (4), CoRE sets $\ell_\psi(e) = -\beta E_{\mathrm{obj}}(e)$ and $b_\theta(e) = -\beta E_{\mathrm{cons},\theta}(e)$. Between neural evaluations, we recompute λ_θ from current states and cache arc contributions, $\lambda_\theta^{\mathrm{base}}$, b, and w, while feasibility checks use the current full state. A new constraint thus supplies another consequence embedding and candidate-cost summand. In the notation of Section 4.3, ϕ_θ comprises ϕ_θ^x and ϕ_θ^T; $\boldsymbol{\kappa}_{\theta,r}(e)$ collects $\mathbf{b}_{e,r}$ with the pooled tail state $\mathbf{a}_{i,r}$ and the retained head state $\mathbf{a}_{j,r}$; ρ_θ maps the constraint sums and the size $|\mathcal{R}|/(1 + |\mathcal{R}|)$ to the node and edge summaries $\boldsymbol{\mu}_\theta$; Φ_θ comprises the graph network and the interaction map of Eq. (37), so that $\mathbf{h}_{\theta,r}(e)$ is $\mathbf{c}_{e,r}$; and H_θ is the grade head with w_{cons}.

C.2 Non-Autoregressive Training and Search

Given the costs of Appendix C.1, search constructs and refines solutions. We then train the scorer from rollout outcomes.

C.2.1 Structured Construction and Refinement

Initialization and refresh. Unless otherwise stated, evaluation uses the inference settings of Table 8. Given one neural evaluation, we run M initial construction rollouts and use the best completed solution as the incumbent. Next, we replay that solution to obtain the state for a second neural evaluation. Subsequent iterations use cached arc contributions and weights recomputed from current states. If the incumbent improves, we evaluate the network again before the next iteration; thus, T_{R} iterations require at most $T_{\mathrm{R}} + 1$ evaluations.

Construction samples actions from Eq. (4). Given an incumbent y, refinement releases at least the minimum number of perturbed edges to obtain a region and makes at most 64 candidate attempts. Next, it applies relocate, Or-opt, segment exchange, single-node swap, 2-opt*, 2-opt, and paired pickup–delivery relocation (Vidal et al., 2013) within that region. The same constraint checks filter candidates in both stages. Prefix and suffix summaries update incrementally evaluable quantities; other updates resimulate the affected routes.

Finally, completed candidates pass route validation before comparison with the incumbent under the original objective:

$$y_{\text{best}}^{(b+1)} \in \underset{y \in \left\{y_{\text{best}}^{(b)}, \tilde{y}_b\right\}}{\arg\min}\ f_I(y), \tag{41}$$

where $\tilde{y}_b$ is feasible candidate solution b. The retained objective is nonincreasing.

Valuing a move. For a solution y and full state z that fixes the route configuration, service history, and current constraint values at the move origin, define

$$\widehat{c}_\theta(y \mid z) = \lambda_\theta(z) \sum_{r \in \mathcal{R}} \sum_{(u,v) \in y} \epsilon_{\theta,r}((u,v)) + \sum_{(u,v) \in y} E_{\text{obj}}((u,v)), \tag{42}$$

For each move, only removed and inserted arcs contribute to the cost difference.

Objective-improving candidates are accepted directly; during bounded exploration, an objective-worsening candidate may also become the current state when its learned cost improves,

$$f_I(\overline{y}) > f_I(y), \qquad \widehat{c}_\theta(\overline{y}) < \widehat{c}_\theta(y), \tag{43}$$

subject to an exploration budget of 10 accepted exploratory moves per refinement invocation.

C.2.2 Training Objective

Policy learning. Rollout i starts from incumbent y_i^0 and produces terminal outcome $y_i^\star$, with reward

$$R_i = \text{clip}\left(\frac{f_I(y_i^0) - f_I(y_i^\star)}{\max(|f_I(y_i^0)|, \varepsilon)} - 10\, \mathbb{I}_i^{\text{inf}},\ -1,\ 1 \right), \tag{44}$$

with $\mathbb{I}_i^{\text{inf}}$ marking an infeasible terminal outcome. Advantages use the POMO shared baseline (Kwon et al., 2020): for the M rollouts of one problem, $\widehat{A}_i = (R_i - \bar{R})/(\sigma_A + \varepsilon)$, where $\bar{R}$ is their mean reward and σ_A is the batch advantage scale; every step t of rollout i uses $\widehat{A}_t = \widehat{A}_i$.

The policy loss is clipped PPO (Schulman et al., 2017) on the policy $p_\theta \equiv p_{\psi,\theta}$ of Eq. (4), with importance ratio $r_t(\theta) = p_\theta(e_t \mid z_t)/p_{\theta_{\text{old}}}(e_t \mid z_t)$,

$$\mathcal{L}_{\text{PPO}} = -\,\mathbb{E}_t[\min(r_t(\theta)\widehat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon_{\text{PPO}}, 1+\epsilon_{\text{PPO}})\widehat{A}_t)], \qquad \epsilon_{\text{PPO}} = 0.1. \tag{45}$$

Each step t is a stochastic perturbation choice, whose probability factorizes into successive softmax choices over the candidates remaining at each inspection until a feasible perturbation is accepted. The neighborhood moves that follow are selected by rule and receive no direct gradient, and their effect reaches training through the rollout reward. Construction, which initializes the incumbent, reads the same learned costs and lies outside the PPO batch.

Optional grade reconstruction. The head in Eq. (38) predicts the computed constraint grade for each constraint–candidate pair using mean squared error:

$$\mathcal{L}_{\text{rec}} = \frac{1}{\sum_{r,e} \mathbb{I}_{r,e}} \sum_e \sum_{r \in \mathcal{R}} \mathbb{I}_{r,e} \left[\widehat{g}_{\theta,r}(e) - g_{r,e}\right]^2, \tag{46}$$

where $\mathbb{I}_{r,e}$ restricts the sum to constraint–candidate pairs whose prefix state the incumbent determines. The grade and cost heads share the interaction map of Eq. (37), so the reconstruction gradient reaches the graph representation and retained consequence embeddings used by the cost head.

Complete objective. The complete training objective is

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} - \lambda_H \mathcal{H}(p_\theta). \tag{47}$$

Setting $\lambda_{\text{rec}} = 0$ gives policy optimization alone, which the reported CoRE model uses; the reconstruction model of Section 5.4 sets $\lambda_{\text{rec}} = 1$.

Relation to neural-guided local search. The refinement above adapts the neural-guided local search of DeepACO (Ye et al., 2023), in which a heatmap guides a short perturbation and a distance-based local search then restores and improves the solution. DeepACO computes its heatmap once per instance and trains it through the construction probabilities, while the outcome of this local search enters only the reward. Following DyNACO (Tran et al., 2026), GCR instead recomputes its costs from the current state and confines each refinement to the region released by a guided perturbation. Within that region, the learned costs rank the descent moves and decide exploratory acceptance as well as the perturbation, and the policy objective of Eq. (45) trains the perturbation choices directly (Appendix E.6).

C.3 Autoregressive Construction

We retain every components of POMO* implemented in CaR (Bi et al., 2026), then add pooled constraint context and next-customer biases to the decoder. The bias enters after content-logit clipping and before masking and softmax in Eq. (4). After each customer selection, we recompute both context and bias.

At each step, the simulator evaluates three constraints: capacity with signed demand, route duration, and time windows. Capacity exposes remaining load, its candidate update, and its grade against the two load bounds. Duration exposes elapsed length, the candidate length, and its grade against the route limit including the return leg, as in Eq. (17). The time-window grade is the minimum of the customer-deadline and depot-return grades. The scales are vehicle capacity, route limit, and depot time horizon. As in Eq. (12), depot visits are the reset events of every accumulating constraint. Each vector also carries active and local-feasibility indicators, and since construction always holds the prior state, no known-state indicator is needed.

The bias reads the full vector of every candidate and constraint. A shared encoder embeds each vector, and weights computed from the constraint tokens and current states scale these embeddings. We sum the weighted embeddings, divide by the square root of the active-constraint count, and pass this summary and the normalized count to a scalar readout, bounded by a scaled tanh. The decoder context instead pools the vectors over candidates: for each constraint, its token reads the current state and the mean and maximum of $\widetilde{x}'_{r,e} - \widetilde{x}_r$ over the candidates. We train encoder and decoder jointly.

D Experimental and Implementation Details

We specify the benchmark and references, the baselines and controlled variants, the parameters and training of both integrations, and the generated instances of the hardness and new-constraint experiments.

D.1 Benchmark and References

Datasets. Our experiments comprise distinct variants and instances for training, validation, and testing:

- **Training:** 22 variants, each with at most two active constraints: TSP, ATSP, OP, PCTSP, PDTSP, VRPTW, CVRP, ACVRP, OCVRP, CVRPB, CVRPBP, CVRPL, CVRPTW, PDCVRP, OCVRPL, OCVRPBP, OCVRPTW, OPDCVRP, MDOCVRP, AMDOCVRP, MDCVRPTW, and MDOCVRPTW. All except the uncapacitated VRPTW belong to the 110-variant benchmark.
- **Validation:** eight fixed instances per training variant, on which each model keeps the epoch with the lowest mean objective.
- **Testing:** all 110 benchmark variants, whose deepest compositions activate four constraints, with 120 or 128 instances per variant at $N = 100$ and 16 at $N = 1000$.

Appendix A defines the problems these names compose.

References. At both sizes, we run PyVRP (Wouda et al., 2024) on each variant it models natively and OR-Tools (Furnon and Perron, 2024) elsewhere, for 60 s per instance, and take the best solution found in that budget as the reference. PyVRP covers 96 of the 110 variants, including the compositions in Table 9, and OR-Tools the remaining 14. For OR-Tools, we run guided local search with up to one vehicle per customer and represent each constraint on its own decision variables. We re-score every reference solution and returned route with the common constraint simulator on the same objective scale.

Reported quantities. For minimization, the reference gap in Section 5.1 is $100(\bar{f} - \bar{f}^{\text{ref}})/|\bar{f}^{\text{ref}}|$, where the bars average objectives over the evaluated instances. Variant-level means receive equal weight in aggregate gaps, and grouped comparisons use the variants shared by all compared methods. We divide total batched evaluation time by instance count to obtain inference time. The autoregressive comparison follows CaR (Bi et al., 2026) and averages per-instance ratios,

$$\text{Gap}_{\text{AR}} = \frac{100}{m} \sum_{i=1}^{m} \frac{f_i - f_i^{\text{ref}}}{f_i^{\text{ref}}}, \qquad m = 1{,}000, \tag{48}$$

Table 8: Architecture, inference, and training settings. Values apply to the non-autoregressive model.

Group	Hyperparameter	Value
Architecture	Hidden width d / graph depth L_E	32 / 12
	Shared context width	32
	Candidate width K	64
Inference	Inverse temperature β	2
	Refinement iterations T_R / rollouts M / exploration budget	10 / 10 / 10
	Minimum perturbed edges / maximum Or-opt segment	8 / 3
Training	Optimizer / learning rate	Adam / 5×10^{-5}
	PPO clip / PPO epochs	0.1 / 4
	Instances × rollouts per PPO batch	up to 13×10
	Epochs × generated instances per epoch	10×500
	Entropy coefficient / gradient clip	0.001 / 1.0
	Reconstruction weight λ_{rec}	0 (1 with reconstruction)
	Auxiliary scale	0.1

using the supplied OR-Tools reference solutions. The all-variant figure takes the same ratio of means against $\bar{f}_*$, the best objective reported for that variant by any plotted method, in place of the 60 s reference. Objective reductions against URS report $100(\bar{f}_{\text{URS}} - \bar{f}_{\text{CoRE}})/\bar{f}_{\text{URS}}$, in which a positive value is an improvement, the opposite sign convention to a gap; paired reductions average $100(f_{\text{URS}} - f_{\text{CoRE}})/f_{\text{URS}}$ over the instances both methods solve feasibly. The sign of every gap is reversed for the prize-maximizing variants OP and AOP.

D.2 Parameters and Training

Non-autoregressive model. Table 8 gives the architecture, inference, and training settings of the non-autoregressive model; the auxiliary scale multiplies λ_{rec}.

Autoregressive models. Following the integration in Appendix C.3, we train consequence-conditioned POMO models on CVRPBLTW at 50 and 100 customers, each for 1,000 epochs with 20,000 generated instances per epoch and effective batch size 128. Following Bi et al. (2026), we use a training mask for visited-node and depot-transition restrictions and a Lagrangian cost for violations of both load bounds, route length, and time windows. This soft constraint handling retains the POMO* setting of CaR and applies to the POMO models alone, whereas the non-autoregressive model masks every candidate that fails a constraint check (Eq. (1)). For each candidate, including numerically violating ones, we compute consequence inputs and optimize encoder and decoder with the construction objective. We evaluate both models at the final epoch on the same 1,000 test instances with eightfold geometric augmentation and greedy customer selection. POMO uses construction alone, whereas the released CaR models apply either k-opt or R&R (Remove-and-Reinsertion) with $T_R \in \{5, 10, 20\}$ refinement steps. For each method, we report total inference time over the full evaluation, including all construction passes and any refinement.

Size transfer and size-matched comparison. The POMO* block of Table 5 compares POMO* with CoRE-POMO* at three train/test sizes. The $50 \to 50$ and $50 \to 100$ rows use the retrained $N = 50$ POMO* baseline and our $N = 50$ model, with no further training before the larger test. The $100 \to 100$ row instead uses the published POMO* result and our separately trained $N = 100$ model, both also reported in Tables 4 and 12. Thus the $50 \to 100$ row measures size transfer, whereas the $100 \to 100$ row is a size-matched comparison.

D.3 Baselines

Neural baselines. We evaluate the released neural baselines of Section 5.1. URS (Zhou et al., 2026) uses its released unified model, and the other neural baselines use their released inference settings with eightfold augmentation. At $N = 100$, each neural baseline evaluates 25,600 rollouts per instance and retains the best: 32 passes over eight augmentations and 100 POMO starts, with the first start greedy and the rest sampled; asymmetric URS instead uses two passes over 128 random

projections and 100 starts. At $N = 1000$, every baseline other than CCL uses two passes over eight augmentations, and CCL uses its released POMO starts and refinement over eight augmentations. The URS evaluation completes 92 of the 110 variants at this size. We rerun every baseline on the same test instances and hardware as CoRE.

Search controls. For comparative fairness, we apply the same candidate width, refinement operators, iteration and rollout counts, and constraint simulator of Table 8 to the three GCR variants. CoRE uses the learned cost of Eq. (40); Distance GCR uses normalized travel distance $\bar{d}_{ij} = d_{ij}/s_d$; Uniform GCR assigns zero cost to every candidate and samples uniformly among feasible candidates. The scale s_d is the root mean squared deviation of candidate distances from their source-node means, replaced by the mean absolute candidate distance or one when that deviation is zero. Table 5 averages reference gaps equally over 12 symmetric single-depot variants without time windows and 12 with time windows at $N = 100$. Over the full benchmark, CoRE lowers the mean gap of Distance GCR from 6.10% to 0.98% on the training variants and from 9.19% to 4.32% on the held-out variants at $N = 100$, and from −3.45% to −7.03% and from 14.88% to 11.22%, respectively, on the 103 variants with both results at $N = 1000$.

Controlled variants. For comparative fairness, we give CoRE, CoRE-State, CoRE-UDR, CoRE-CaDA, and the reconstruction model the same 22 training variants, 5,000 sampled instances, decoder, inference settings, and test instances. We also use the optimization settings of Table 8 and the same active parameter shapes, apart from the prompt layers of CoRE-CaDA. The input comparison changes the entries supplied to both pooled graph features and retained constraint–candidate embeddings; the supervision comparison holds those entries fixed. Reported means cover the 21 training and 89 held-out variants, and each gap uses the same reference as the main comparison.

Consequence inputs. CoRE receives the full vector of Eq. (2); CoRE-State receives its state entries $(\widetilde{x}_r, \widetilde{x}_{r,e})$ alone, both with the known-state indicator, so that both models read the simulated transition and differ only in the entries derived from the check and the reset. Both use the policy objective with $\lambda_{\text{rec}} = 0$. Routing and objective features are present in every model.

Grade reconstruction. The reconstruction model uses the inputs of CoRE and adds $\mathcal{L}_{\text{rec}}$ with $\lambda_{\text{rec}} = 1$. Table 3 evaluates CoRE and the reconstruction model on the same hardness instances.

CoRE-UDR. We adapt CoRE-UDR from the unified data representation of URS (Zhou et al., 2026), which extends the attribute inputs of MTPOMO and MVMoE (Liu et al., 2024; Zhou et al., 2024) to a 13-bit multi-hot task vector with six numerical node attributes and five static indicators. Its task vector conditions the node and arc projections and a task embedding, while the graph network, candidate head, simulator, and search match CoRE. The comparison thus separates the effect of consequences from that of prior attribute representations.

CoRE-CaDA. We adapt CoRE-CaDA from the constraint prompt of CaDA (Li et al., 2025), a multi-hot vector over constraint types that we extend from its four route attributes to the other constraints of the benchmark, and fuse it with the node attributes of CoRE-UDR through two linear layers. The graph network, candidate head, simulator, search, and training schedule match CoRE, and, to isolate the constraint representation, the shared graph network replaces the dual-attention encoder of CaDA.

D.4 Instance Generation

Compositional hardness. Following the hardness levels of PIP (Bi et al., 2024), also adopted by CaR (Bi et al., 2026), which tighten time windows around a planted feasible tour, we plant a feasible solution and tighten every numerical bound around it. Since a new route resets every route-scoped state, the fleet limit tightens with the same parameter, so that opening routes cannot restore the slack that the tighter bounds remove. Given signed capacity, linehaul-before-backhaul ordering, route duration, time windows, and a solution-wide fleet limit, we plant one such solution in each $N = 100$ instance. *Short-route packing* uses ten routes with eight linehaul and two backhaul customers per route; *Long-route packing* uses five routes with eighteen linehaul and two backhaul customers per route. For each matched instance, coordinates, demands, and planted routes remain fixed across hardness levels.

For each continuous bound b_j, let b_j^{tight} be the value constructed from the planted solution and b_j^{loose} its loose counterpart. The evaluated bound is

$$b_j(\alpha) = b_j^{\text{tight}} + \alpha\left(b_j^{\text{loose}} - b_j^{\text{tight}}\right), \qquad \alpha \in [0, 1]. \tag{49}$$

Capacity, route duration, and window width share the interpolation parameter α, while the fleet limit follows $K(\alpha) = K_{\min} + \lfloor \alpha(N - K_{\min}) \rfloor$. Here $K_{\min} = 10$ for Short-route packing and $K_{\min} = 5$ for Long-route packing. Medium uses $\alpha = 0.10$, giving fleet limits 19 and 14, respectively; tightening to Hard with $\alpha = 0.05$ reduces them to 14 and 9.

For CoRE, we use the default settings of Table 8; for URS, POMO-10 with one augmentation; and for RouteFinder-MoE and CCL, POMO-100 with eight augmentations. We check every sampled solution under the same signed-load and fleet constraints.

New constraints. Each setting of Section 5.2 adds one new constraint to generated instances and compares the same pretrained model with OR-Tools at 600 s per instance, which solves every instance. All solvers use the same single-depot instances per setting. EVRP and VRPDL add their constraints to capacitated instances. For EVRP, following the geometric sampling in Zhou et al. (2026) and demand sampling in Liu et al. (2024), we sample coordinates uniformly on $[0, 1]^2$ and demands from $\{1, \ldots, 9\}$, then normalize demand by capacity 50. VRPDL uses the generated geometry and demand data of Bi et al. (2026), with capacity and destination draft limits active.

Battery capacity is $1.05 \max_v(d_{0v} + d_{v0})$, and chargers comprise 2% of customers. Half of the customers, chosen uniformly at random, receive a restricted draft cap sampled uniformly from $[|q_v|, 1]$; the others have cap one. The battery allowance covers a depot–customer–depot trip; draft caps admit the destination demand.

Following the capacitated benchmark geometry of Zhou et al. (2026), VRPCL and VRPIC use its coordinates and single depot; Appendix A.4 defines our groups, conflict graph, and objective. The OR-Tools model bounds the number of distinct groups each vehicle enters and adds a disequality between the vehicle variables of incompatible customers; where its insertion heuristics construct no first solution, its search starts from an angular-sweep solution.

E Additional Experiments and Analysis

The results below use the evaluation populations and reporting conventions of Appendix D.1.

E.1 Additional Benchmark Results

Table 9 gives the reference gaps and per-instance time ranges of the eight symmetric closed-route variants of Table 1 at both sizes, retaining every method with recorded results for at least one setting. Figure 4 shows quality and runtime for every variant at $N = 100$, where each point is a method–variant mean against the best-reported objective $\bar{f}_*$ of Appendix D.1.

Training coverage and transfer populations. Figure 5 groups all 96 capacitated variants by joint constraint set. These variants cross route closure, cost symmetry, depot count, delivery or backhaul demand structure, route limits, and time windows. Mixed backhauls change signed capacity increments, while ordered backhauls also activate precedence. Four of the eight distinct constraint sets occur in training. The other four contain 40 held-out variants and combine backhaul precedence with route limits or time windows, or route limits with time windows. Within each set, routing structure and numerical parameters vary.

For Figure 2(a), we use the 48 symmetric capacitated variants shared by all four displayed methods, including single and multiple depots and open and closed routes. The groups with one, two, three, and four active constraints contain 8, 20, 16, and 4 variants, respectively. We give each variant equal weight in its group mean.

For Figure 2(b), we use the 24 symmetric, single-depot variants with CoRE and URS results at both sizes: 14 use an observed constraint set and 10 use a new set. The bars average the same variants at each size, evaluated on separate size-specific instance populations.

Table 9: Eight-composition routing comparison. Gaps (%) on eight closed-route compositions; † marks a CoRE training variant. Distance GCR runs the search of CoRE with travel-distance costs; bold: best learned gap.

$N = 100$**: reference gap (%)**

Method	C†	CL†	CTW†	CLTW	CBP†	CBPL	CBPTW	CBPLTW	Time range
URS	1.14%	1.06%	4.43%	4.90%	13.73%	13.84%	12.68%	12.74%	0.32–0.43s
CoEKS	2.04%	2.10%	21.95%	22.02%	4.32%	4.19%	22.45%	22.20%	0.40–0.48s
MoSES	0.93%	0.89%	22.19%	22.86%	2.63%	2.75%	21.42%	22.10%	0.51–0.63s
RF-T	1.24%	1.11%	26.69%	26.22%	2.87%	3.10%	25.26%	25.06%	0.29–0.36s
RF-POMO	1.49%	1.49%	19.82%	19.62%	3.53%	3.60%	19.22%	18.55%	0.30–0.36s
RF-MoE	1.35%	1.24%	19.64%	19.63%	3.08%	3.34%	18.38%	18.45%	0.41–0.51s
MTPOMO	1.65%	1.61%	20.70%	20.31%	3.70%	4.24%	19.50%	18.94%	0.30–0.36s
MVMoE	1.45%	1.37%	25.65%	25.62%	3.46%	3.75%	24.11%	23.85%	0.41–0.51s
CCL	**0.86%**	**0.80%**	20.00%	20.75%	2.19%	2.22%	20.79%	20.51%	0.77–0.92s
Distance GCR	4.12%	3.74%	6.00%	6.14%	4.92%	4.81%	9.62%	10.65%	0.43–0.88s
CoRE	1.93%	1.57%	**3.80%**	**3.81%**	**2.08%**	**2.09%**	**6.14%**	**6.40%**	0.20–0.40s

$N = 1000$**: reference gap (%)**

Method	C†	CL†	CTW†	CLTW	CBP†	CBPL	CBPTW	CBPLTW	Time range
URS	9.27%	9.05%	11.36%	11.38%	29.45%	30.48%	28.73%	28.46%	5.17–10.33s
CoEKS	42.04%	41.04%	58.98%	59.63%	55.27%	59.49%	56.46%	55.47%	9.61–11.92s
MoSES	70.47%	71.17%	59.05%	63.09%	61.85%	73.11%	55.61%	59.27%	11.01–12.70s
RF-T	26.75%	26.60%	73.57%	71.59%	33.15%	31.69%	66.73%	65.93%	9.15–11.44s
RF-POMO	24.42%	23.77%	58.49%	57.41%	35.33%	34.65%	52.33%	51.90%	9.08–11.20s
RF-MoE	27.42%	31.99%	53.26%	50.63%	91.07%	95.79%	49.91%	47.96%	9.90–12.31s
MTPOMO	62.78%	62.39%	59.44%	57.50%	71.80%	73.25%	54.69%	52.82%	9.06–11.15s
MVMoE	32.27%	32.11%	58.07%	57.38%	46.62%	44.10%	52.67%	51.40%	10.89–12.62s
CCL	12.50%	14.20%	112.22%	113.43%	16.48%	19.48%	120.38%	108.20%	5.63–10.21s
Distance GCR	8.17%	7.88%	7.18%	7.60%	13.51%	15.04%	11.96%	12.29%	3.38–11.60s
CoRE	**6.10%**	**5.86%**	**6.03%**	**6.31%**	**9.61%**	**11.09%**	**10.39%**	**9.96%**	6.12–17.91s

E.2 Case Study: Consequence Definitions for Route Incompatibility

We present a case study on 32 VRPIC instances, generated as in Appendix D.4, to analyze how the definition of the consequence vector shapes the gain of CoRE on route incompatibility. Since route incompatibility has a binary check without scalar slack, we evaluate the pretrained CoRE checkpoint under four definitions of its vector with the default inference settings. All four share the conflict graph, feasibility check, and depot event.

With F, F_v, and $N_H(v)$ as in Appendix A.4, the *binary-feasibility* definition is read off the check alone: both states are zero, and the grade is one for an accepted move and -1 for a rejected one. The other three give a rejected customer the negative grade of Appendix A.4. The *static-degree* definition sets the successor state to $\min\{\deg_H(v)/6, 1\}$ and the grade to its complement. The *global-count* definition measures the surviving frontier against the instance, with states $1 - |F|/N$ and $1 - |F_v|/N$ and grade $|F_v|/N$, and the *route-relative* definition of Appendix A.4 measures it against the current route.

Table 10: VRPIC consequence definitions. Mean objectives, gaps to OR-Tools, and total times on 32 instances.

Method	Obj ↓	Gap ↓	Time ↓
OR-Tools (60 s)	17.591	0.00%	32.1m
CoRE-UDR	17.533	−0.33%	17.2s
CoRE (binary feasibility)	17.357	−1.33%	30.2s
CoRE (static degree)	**17.292**	**−1.70%**	32.9s
CoRE (global count)	17.307	−1.61%	33.8s
CoRE (route-relative survival)	17.293	−1.69%	33.5s

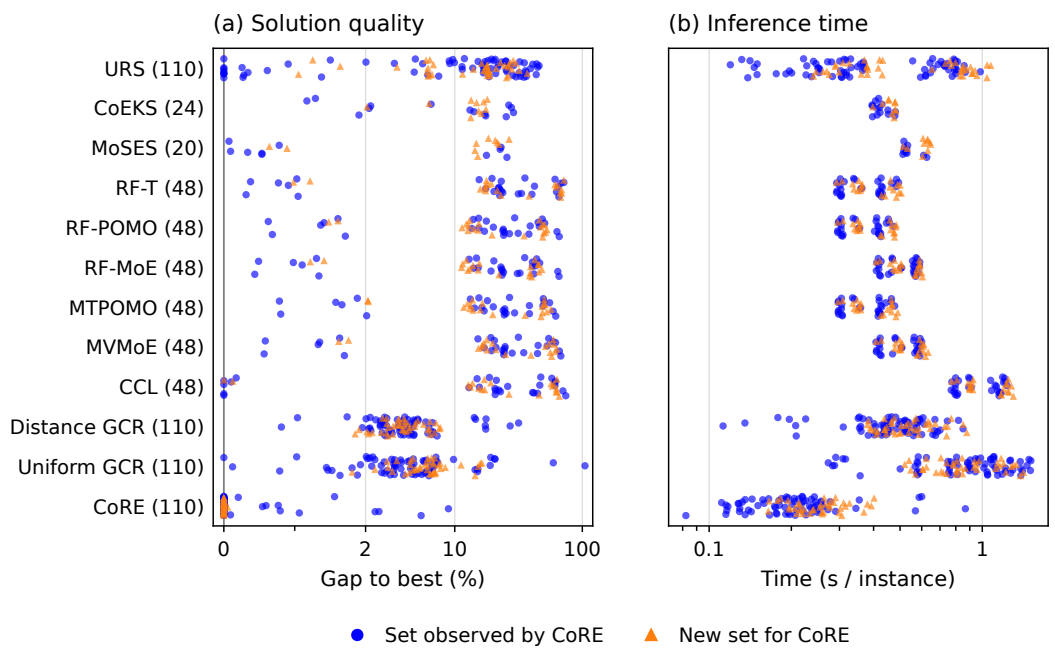


Figure 4: Quality and runtime across all reported variants at $N = 100$. Each point is one variant mean; counts show each method's coverage. Blue circles use constraint sets observed during CoRE training; orange triangles use new sets. Quality is the gap to the best recorded objective for each variant, on an axis that is linear up to 2% and logarithmic thereafter. Runtime uses a logarithmic axis. Lower values on either axis are better.

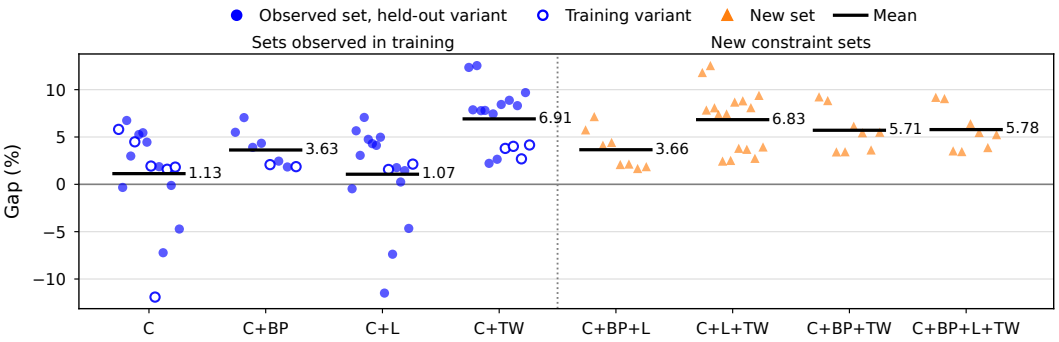


Figure 5: Composition transfer by constraint set. CoRE reference gaps at $N = 100$ for every capacitated variant, grouped by its joint constraint set.

In Table 10, the binary-feasibility definition, whose vector carries only the outcome of the feasibility check, already attains 4.0× the improvement of CoRE-UDR over OR-Tools. The static-degree definition adds candidate-specific information and the route-relative definition route-specific information, raising it to 5.2× and 5.1×, respectively. The global-count definition measures the frontier against the whole instance rather than the route and still reaches 4.9×. These results show that the consequence vector transfers even when read off the feasibility check alone, and that candidate- and route-specific definitions raise the gain further.

E.3 Results under Compositional Hardness

To examine whether the trends of Section 5.2 hold as the bounds loosen, Table 11 reports mean objectives and infeasible rates at both levels for the instances of Appendix D.4. Among instances solved feasibly by both CoRE and URS, CoRE reduces the mean objective by 28.44% and 24.31% under Short-route packing at Medium and Hard, respectively, and by 26.90% and 23.48% under

Long-route packing. These results suggest that consequence guidance retains its feasibility and objective advantage over URS at both levels.

Table 11: Cost and feasibility under compositional hardness at both levels. Results use 128 matched $N = 100$ instances per setting; K is the fleet limit. Objectives average each method's feasible instances; bold marks the lowest infeasible rate.

Method	Short-route packing				Long-route packing			
	Medium ($K = 19$)		**Hard** ($K = 14$)		**Medium** ($K = 14$)		**Hard** ($K = 9$)	
	Obj. ↓	**Infsb%** ↓	**Obj.** ↓	**Infsb%** ↓	**Obj.** ↓	**Infsb%** ↓	**Obj.** ↓	**Infsb%** ↓
CoRE	14.478	**0.00%**	15.273	45.31%	11.449	**0.00%**	11.705	8.59%
+ $\mathcal{L}_{\text{rec}}$	14.441	**0.00%**	15.222	**10.16%**	11.398	**0.00%**	11.642	**0.78%**
URS	20.267	7.81%	19.491	94.53%	15.762	8.59%	15.224	85.94%
RF-MoE	26.693	99.22%	—	100.00%	—	100.00%	23.324	99.22%
CCL	—	100.00%	—	100.00%	—	100.00%	—	100.00%

E.4 Results on Autoregressive Construction

To compare against the full set of construction and improvement baselines reported by Bi et al. (2026), Table 12 follows the evaluation procedure of Appendix D.2 and includes CaR at three levels of the refinement budget. Among methods that return a feasible solution on every instance, CoRE-POMO* attains the lowest gap at both sizes by construction alone, below CaR with every refinement operator and budget. NeuOpt-GIRE reaches lower gaps at $N = 50$ only over its feasible instances, leaving 27% to 34% of them infeasible.

E.5 Comparison with Specialists and Post-Hoc Search

Non-autoregressive and diffusion specialists. To the best of our knowledge, CoRE is the first non-autoregressive method that targets transfer across routing problems, so the generalists of Section 5.2 available for comparison are all autoregressive. Here, we further compare CoRE with non-autoregressive and diffusion specialists, which also score candidate graphs but learn a heuristic for one routing problem; their released checkpoints cover CVRP, with NeuroLKH (Xin et al., 2021a) also covering CVRPTW. We compare on these two problems at $N = 100$ and $N = 500$, using 16 paired instances per set and each method's released checkpoint and evaluator. Since several checkpoints were trained at other sizes and their default budgets are shorter than the search of CoRE, at $N = 500$ we raise each budget to its largest tested setting within the time of CoRE with $T_R = 100$ refinement iterations on the same A100 host. Every returned solution is checked for customer coverage, capacity, time windows, and exact Euclidean cost.

As shown in Table 13, CoRE with $T_R = 100$ attains the lowest gap on CVRP100, CVRP500, and CVRPTW100, and improves on PyVRP at CVRP500. On CVRPTW500, NeuroLKH, whose checkpoint is trained on CVRPTW, attains a lower gap. With $T_R = 10$, CoRE runs within the time range of the specialists at $N = 100$ and attains a lower CVRP100 gap than most specialists. These results suggest that the shared consequence space retains single-problem quality, as one set of parameters covering all 110 variants remains competitive with these specialists on both problems.

Relation to post-hoc search. (1) Heatmap-based solvers for the traveling salesman problem, such as DIMES, DIFUSCO, and UTSP (Qiu et al., 2022; Sun and Yang, 2023; Min et al., 2023), apply Monte Carlo tree search after training. The heatmap is fitted without search, and the search then improves its decoded tours under its own rules. For instance, Xia et al. (2024) match these solvers with a heatmap computed from distances alone, and Pan et al. (2026) with a tuned search that uses little heatmap information. The gains of this paradigm therefore largely stem from the post-hoc search. (2) CoRE, in contrast, trains its costs through the refinement they guide, and the same costs rank every refinement move at inference. Under the same inference search, a scorer trained on construction rollouts alone raises the gap on every population (Table 14), and so does Distance GCR, the counterpart of a distance-based heatmap and the control that Xia et al. (2024) prescribe (Table 5). (3) Moreover, under tight compositions, feasibility is decided at construction before refinement acts, so the lower infeasibility of CoRE in Section 5.2 arises from its learned costs alone.

Table 12: Construction and refinement under complex constraints. Results on CVRPBLTW with 1,000 instances per size. ⋆ denotes results from Bi et al. (2026); CaR and size-matched CoRE-POMO* models are evaluated in this study. Obj. and Gap use the best retained feasible solution; Infsb% measures instances without one. Gap averages per-instance ratios to the OR-Tools reference solutions of Bi et al. (2026) (Eq. (48)), so it need not match the ratio of mean objectives. Bold marks the best objective and gap among fully feasible methods. CoRE-POMO* uses construction alone.

	Method	#Params	Paradigm	$N = 50$				$N = 100$			
				Obj. ↓	Gap ↓	Infsb% ↓	Time ↓	Obj. ↓	Gap ↓	Infsb% ↓	Time ↓
CVRPBLTW	OR-Tools (short)⋆	/	I	14.890	1.402%	0.00%	10.4m	25.979	2.518%	0.00%	20.8m
	OR-Tools (long)⋆	/	I	14.677	⋄	0.00%	102m	25.342	⋄	0.00%	210m
	POMO⋆	1.25M	L2C-S	15.999	9.169%	0.00%	2s	27.046	7.004%	0.00%	4s
	POMO-MTL⋆	1.25M	L2C-M	15.980	9.035%	0.00%	2s	27.247	7.746%	0.00%	7s
	MVMoE⋆	3.68M	L2C-M	15.945	8.775%	0.00%	3s	27.142	7.332%	0.00%	10s
	ReLD-MoEL⋆	3.68M	L2C-M	15.925	8.623%	0.00%	3s	27.044	6.915%	0.00%	9s
	POMO+EAS+SGBS* (short)⋆	1.25M	L2C-S	15.386	4.831%	0.00%	25s	26.005	2.616%	0.00%	2.3m
	POMO+EAS+SGBS* (long)⋆	1.25M	L2C-S	15.156	3.263%	0.00%	10.3m	25.558	0.854%	0.00%	60m
	POMO* (relaxed)⋆	1.25M	L2C-S	14.873	2.310%	0.00%	2s	24.592	−1.645%	0.00%	4s
	NeuOpt-GIRE*‡ ($T = 1$k)⋆	0.69M	L2I-S	14.521	1.329%	33.80%	1.1m	24.597	3.390%	51.20%	2.5m
	NeuOpt-GIRE*‡ ($T = 2$k)⋆	0.69M	L2I-S	14.352	−0.031%	31.00%	2.2m	24.365	0.875%	44.50%	5.1m
	NeuOpt-GIRE*‡ ($T = 5$k)⋆	0.69M	L2I-S	14.201	−1.163%	27.30%	5.5m	24.038	−1.541%	39.10%	12.7m
	CaR (k-opt, $T_R = 5$)	1.64M	L2(C+I)-S	14.869	2.250%	0.00%	9.8s	24.598	−1.669%	0.00%	30.0s
	CaR (k-opt, $T_R = 10$)	1.64M	L2(C+I)-S	14.865	2.220%	0.00%	15.0s	24.590	−1.702%	0.00%	44.7s
	CaR (k-opt, $T_R = 20$)	1.64M	L2(C+I)-S	14.847	2.116%	0.00%	25.2s	24.586	−1.720%	0.00%	73.7s
	CaR (R&R, $T_R = 5$)	1.72M	L2(C+I)-S	14.726	1.325%	0.00%	10.9s	24.553	−1.834%	0.00%	30.7s
	CaR (R&R, $T_R = 10$)	1.72M	L2(C+I)-S	14.662	0.894%	0.00%	17.1s	24.479	−2.136%	0.00%	46.2s
	CaR (R&R, $T_R = 20$)	1.72M	L2(C+I)-S	14.601	0.471%	0.00%	29.7s	24.400	−2.448%	0.00%	77.4s
	CoRE-POMO*	1.26M	L2C-S	**14.543**	**0.098%**	0.00%	14.6s	**24.166**	**−3.324%**	0.00%	104.6s

⋆: values reproduced from Table 2 of Bi et al. (2026). I: improvement; L2C/L2I: learned construction/improvement; S/M: single-/multi-task. Following CaR's notation, * marks relaxed constraint handling and ‡ marks added solution-level features. ⋄ identifies the CaR reference solution. T_R counts CaR refinement steps. POMO-MTL is MTPOMO (Liu et al., 2024); EAS (Hottung et al., 2022), SGBS (Choo et al., 2022), and NeuOpt-GIRE (Ma et al., 2023) follow the configurations of Bi et al. (2026).

Table 13: Non-autoregressive and diffusion specialists. Mean gaps to PyVRP (60 s per instance) and total time on 16 paired instances per set. Specialists use their released CVRP or CVRPTW checkpoints; CoRE is the single model of Section 5.2. At $N = 500$, each specialist uses its largest tested budget within the time of CoRE with $T_R = 100$ refinement iterations. —: not evaluated or not recorded; bold: best learned gap.

(a) CVRP

Method	$N = 100$ Gap ↓	$N = 100$ Time ↓	$N = 500$ Gap ↓	$N = 500$ Time ↓
PyVRP (60 s) (Wouda et al., 2024)	0.00%	16m	0.00%	16m
Non-autoregressive				
NeuroLKH (Xin et al., 2021a)	1.67%	5.6s	−0.11%	268s
DeepACO (Ye et al., 2023)	3.31%	3.6s	2.94%	78s
HBG-GFACS (Zhang and Cao, 2025)	19.92%	3.3s	12.85%	249s
HBG-AGFN (Zhang and Cao, 2025)	11.20%	2.0s	8.47%	7.4s
DyNACO (Tran et al., 2026)	4.01%	3.0s	0.42%	260s
DPDP (Kool et al., 2022)	7.38%	3.2s	7.89%	94s
GLOP (Ye et al., 2024)	20.22%	4.4s	21.68%	125s
UDC (Zheng et al., 2024)	7.01%	191s	2.32%	258s
Diffusion				
NExCO (Wang et al., 2026)	1.44%	2.7s	0.42%	191s
CAM (Feng et al., 2026)	2.63%	3.5s	0.20%	241s
Generalist				
CoRE ($T_R = 10$)	2.10%	3.1s	1.48%	209s
CoRE ($T_R = 100$)	**1.03%**	28s	**−0.54%**	270s

(b) CVRPTW

Method	$N = 100$ Gap ↓	$N = 100$ Time ↓	$N = 500$ Gap ↓	$N = 500$ Time ↓
PyVRP (60 s)	0.00%	16m	0.00%	16m
NeuroLKH	3.90%	5.7s	**0.86%**	11m
CoRE ($T_R = 10$)	4.12%	5.5s	4.80%	11m
CoRE ($T_R = 100$)	**2.21%**	48s	2.12%	12m

CaR (Bi et al., 2026) and CoRE both train with refinement following after construction, but differ in how the learned signal reaches refinement. CaR trains a refinement decoder for its operators, while CoRE amortizes one scorer over construction and refinement and credits the neighborhood moves only through the rollout outcome, so the same costs rank the moves of any operator.

E.6 Component Ablations

Training through refinement. Heatmap methods such as DIMES (Qiu et al., 2022) train their scores on sampled constructions and refine the decoded solutions only at inference. To examine whether training through refinement contributes beyond the refinement applied at inference, we train the same model on the construction rollouts of GCR, as in this practice, instead of its refinement rollouts. For comparative fairness, both models share the training variants, instances, schedule, and settings of Table 8, as well as the search at inference, so that they differ only in the rollouts that supply the policy gradient.

Table 14: Training through refinement. Mean reference gaps (%) of CoRE trained on construction or refinement rollouts, with the same search at inference.

Training rollouts	Training ↓	Held-out ↓	Observed sets ↓	New sets ↓	All ↓
Construction	1.66%	5.07%	3.77%	6.66%	4.42%
Refinement (CoRE)	**0.98%**	**4.32%**	**3.15%**	**5.76%**	**3.68%**

Results show that training on refinement rollouts lowers the reference gap on every population in Table 14, including the new constraint sets. This confirms that refinement contributes through the scorer trained within it, rather than as a procedure applied after training.

Constraint weighting. The weight of Eq. (39) reads the joint state $\sum_{s\in\mathcal{R}} \widetilde{x}_s$ of all active constraints, and the constraint token passes through the residual layer Γ_θ. Table 15 compares CoRE + $\mathcal{L}_{\text{rec}}$ with two controlled variants. The per-constraint variant gives each constraint its own weight read from its own state, $\lambda_\theta^{\text{base}}\, 2\sigma(b + w\,\widetilde{x}_r)$, and the token variant removes Γ_θ. All three models share consequence inputs, training data and objective, model selection, inference settings, parameter layout, and test instances.

Table 15: Constraint weighting. Mean reference gaps (%) of CoRE + $\mathcal{L}_{\text{rec}}$ and its weighting ablations.

Configuration	Training ↓	Held-out ↓	All ↓
CoRE + $\mathcal{L}_{\text{rec}}$	**0.48%**	**4.59%**	**3.80%**
per-constraint weight	1.07%	4.63%	3.95%
w/o Γ_θ	1.32%	4.72%	4.07%

As shown in Table 15, per-constraint weights raise the mean reference gap relative to the full model by 0.59 percentage points on training variants and 0.04 on held-out variants. Removing Γ_θ raises these gaps by 0.84 and 0.13 percentage points, respectively. These comparisons support both designs for the training compositions, where their effects are largest. Since each contribution $\epsilon_{\theta,r}$ already reads the consequences of its own constraint, the weight contributes most by reading the joint state of all active constraints, as in the full model.

E.7 Mechanism Analysis

Feasibility of preferred candidates. To examine how often each model's preferred candidates remain feasible (Section 5.4), we evaluate the POMO models on the 1,000 CVRPBLTW test instances of Appendix D.2 with their soft-constrained construction, eightfold augmentation, and greedy selection, and the masked non-autoregressive models on the Hard instances of both packings (Appendix D.4) and 16 held-out CVRPBPLTW instances with the settings of Table 8. We set the pool of each decision to the candidates that each model's mask admits, rank the chosen candidate within it, and certify refinement moves in learned-cost order as in Appendix C.2. Table 16 reports five quantities. Objective preference is the mean percentile of the chosen candidate's travel cost within its pool, with one for the cheapest. Preferred feasible is the share of shortest rollouts per instance and augmentation that satisfy every constraint for POMO, and the share of initial construction rollouts that complete for the non-autoregressive models. Infeasibility loss is the relative length of the best feasible rollout over the best rollout. Feasible moves is the share of certified refinement moves that are feasible and improving. Finally, over the M initial construction rollouts per Hard instance, we count the customers left unserved when a construction exhausts the fleet limit.

Table 16: Decision statistics. POMO models on CVRPBLTW at $N = 50$; non-autoregressive models on Short-route packing at Hard and held-out CVRPBPLTW.

Integration	Model	Objective pref.	Preferred feasible ↑	Infeasibility loss ↓	Feasible moves ↑	Unserved at failure ↓
POMO	POMO*	0.772	59.0%	1.31%	–	–
	CoRE-POMO*	0.776	**83.4%**	**0.45%**	–	–
GCR	CoRE-UDR	0.999	2.3%	–	1.06%	12.4
	CoRE	0.976	**55.5%**	–	**1.39%**	**2.1**

These statistics extend Table 7: with consequence inputs, both solvers keep more of their preferred candidates feasible, and the non-autoregressive model also certifies more feasible moves.

Input range on held-out variants. To examine whether held-out inputs remain within the training range (Section 5.4), we build the scorer inputs that each model consumes on the benchmark instances at $N = 100$: one consequence vector $(\widetilde{x}_r, \widetilde{x}_{r,e}, g_{r,e}, q_{r,e})$ per candidate arc and active constraint for CoRE and its first two entries $(\widetilde{x}_r, \widetilde{x}_{r,e})$ for CoRE-State, and one node vector of task bits and node attributes for CoRE-UDR. Instances 0–5 of the 21 training variants form the reference set, instances 6–7 of the same variants calibrate the distance scale, and instances 0–1 of every held-out variant form the queries. We measure the Euclidean distance from each input to its nearest reference input and report the share of inputs farther than the 99th percentile of the calibration distances, averaged over variants in Table 17.

Table 17: Inputs outside the training range (%), by population.

Population	CoRE	CoRE-State	CoRE-UDR	CoRE-UDR w/o task bits
Training (21)	1.2	1.3	1.0	1.0
Held-out, observed sets (49)	3.7	2.9	61.3	18.4
Held-out, new sets (40)	3.5	2.2	80.5	47.0

CoRE-UDR leaves the training range on 47.0% of new-set inputs even without its task bits, so its out-of-range share in Section 5.4 does not stem from the task vector alone.

F Consequence Guidance for Node Problems

We apply the consequence representation to vertex selection and assignment with policies trained separately from routing. Appendix F.1 specifies the tasks and models, and Appendix F.2 compares one generalist across four objectives with task-specific specialists and evaluates a vertex-selection model on held-out constraint compositions.

F.1 Node Tasks and Models

Tasks. MIS admits a vertex when none of its neighbors is selected, and maximum clique applies the same check to the complement graph; MVC selects vertices until every edge is covered, and maximum cut assigns each vertex to one of two sides. For exclusion, let $F(S)$ be the unselected vertices with no conflict with the selected set S, and let $F_v = F(S) \setminus (\{v\} \cup N(v))$. A feasible insertion has the consequence vector

$$\mathbf{u}_E(v, S) = \left(1 - \frac{|F(S)|}{|V|},\ 1 - \frac{|F_v|}{|V|},\ \frac{|F_v|}{|V|},\ 0\right). \tag{50}$$

For MVC, the state entries and grade are covered-edge fractions, and for maximum cut the objective head reads the change in cut value. The composition experiment maximizes reward $W(S) = \sum_{v \in S} w_v$ under pairwise exclusion E, linear budgets B, and partition-group limits Q:

$$\begin{aligned} E &: \{u, v\} \not\subseteq S && \text{for every conflict } \{u, v\}, \\ B &: \textstyle\sum_{v \in S} c_{vr} \le b_r && \text{for every budget } r, \\ Q &: |S \cap V_g| \le k_g && \text{for every group } g, \end{aligned} \tag{51}$$

where exclusion uses Eq. (50), and budgets and group limits use normalized current and successor consumption with the residual allowance as grade.

Models. The generalist is a three-layer residual message-passing network of width 64 that scores every candidate–constraint pair with one shared head, plus a side-conditioned head for cut. It is trained on ER graphs across the four tasks by imitating feasible partial reference solutions and a greedy teacher with local improvement, and one model selected on validation graphs serves every test population. The vertex-selection model keeps a hidden state per candidate–constraint pair through its message-passing layers and is trained on E, B, Q, EB, and EQ. The matched CoRE-A model shares its architecture and supervision but replaces the successor and grade entries with normalized consumption attributes.

Evaluation. The node test graphs are the ML4CO-Bench-101 populations (Ma et al., 2025b) released with COExpander (Ma et al., 2025a): RB and BA graphs with 200–300 and 800–1200 vertices and ER graphs with 700–800 vertices. Gaps use the released KaMIS (Lamm et al., 2017) solutions for MIS and Gurobi (Gurobi Optimization, LLC, 2024) solutions otherwise, and published times keep their original hardware. For the composition experiment, SCIP (Bolusani et al., 2024) and CP-SAT (Perron and Didier, 2024) certify the optimum of 288 fresh graphs per size, and every method is compared at a matched number M of completed solutions against multi-heuristic search over eight fixed rankings.

F.2 Experimental Results

One generalist across node objectives. Tables 18 and 19 compare the generalist, applied without retraining to RB, BA, and larger ER graphs, with published task-specific specialists at their first reported configuration. After $T_{\mathrm{R}} = 100$, the generalist remains competitive with these specialists across the four objectives, while the specialists attain lower gaps on clique and cut.

Table 18: Generalist transfer across MIS datasets. Mean objective, reference gap (%), and inference time (seconds per graph); published neural baselines are MIS specialists.

	RB-[200–300]			RB-[800–1200]			ER-[700–800]		
Method	**Obj. ↑**	**Gap ↓**	**Time ↓**	**Obj. ↑**	**Gap ↓**	**Time ↓**	**Obj. ↑**	**Gap ↓**	**Time ↓**
KaMIS[§]	20.09	0.00%	45.81s	43.00	0.00%	56.97s	44.97	0.00%	60.75s
Gurobi[§]	20.09	0.00%	0.54s	42.19	1.83%	33.84s	38.78	13.75%	60.49s
GFlowNets[‡]	19.18	4.57%	0.06s	37.48	13.14%	0.52s	41.14	<u>8.53%</u>	0.35s
DIFUSCO ($S = 1, T_s = 50$)[§]	—	—	—	—	—	—	38.64	14.08%	2.80s
Fast-T2T ($S = 1, T_g = 5, T_s = 5$)[§]	19.50	<u>2.89%</u>	0.41s	—	—	—	40.69	9.51%	1.03s
DiffUCO: CE ($S = 1, F = 1$)[§]	19.20	4.37%	0.47s	38.49	<u>10.43%</u>	4.71s	—	—	—
Greedy	18.48	7.92%	0.01s	37.78	12.06%	0.02s	38.85	13.60%	0.01s
CoRE generalist ($T_{\mathrm{R}} = 100$)	19.72	**1.78%**	0.63s	40.36	**6.06%**	4.28s	42.22	**6.11%**	2.25s

[§] Table 1 of Ma et al. (2025a). KaMIS (Lamm et al., 2017), Gurobi (Gurobi Optimization, LLC, 2024), GFlowNets (Zhang et al., 2023), DIFUSCO (Sun and Yang, 2023), Fast-T2T (Li et al., 2024), DiffUCO (Sanokowski et al., 2024). ‡: Tables 1 and 4 of Zhang et al. (2023), using their original 500-graph populations, aggregate-objective gaps, and total times divided by 500. T_{R}: refinement iterations; S: samples; T_s: denoising steps; D_s: determination steps; T_g: refinement steps; F: diffusion-step multiplier. A dash denotes an unreported result; bold: best gap; underline: second-best gap, excluding KaMIS and Gurobi.

Zero-shot composition within vertex selection. We apply the vertex-selection model without retraining to BQ and EBQ, which are absent from training and validation, at $N = 32$ and at $N = 64$, which exceeds the training sizes. As shown in Table 20, it attains a lower gap than multi-heuristic search on every held-out composition and size at both $M = 32$ and $M = 128$. CoRE-A attains similar gaps, since for these non-restoring linear constraints both inputs determine the same successor utilization and slack (Appendix B.1).

Table 19: One generalist across node-selection and assignment tasks. Gaps (%) and times (seconds per graph); published learned baselines are task-specific specialists.

	Maximum clique				Minimum vertex cover				Maximum cut			
	RB-[200–300]		RB-[800–1200]		RB-[200–300]		RB-[800–1200]		BA-[200–300]		BA-[800–1200]	
Method	**Gap ↓**	**Time ↓**	**Gap ↓**	**Time ↓**	**Gap ↓**	**Time ↓**	**Gap ↓**	**Time ↓**	**Gap ↓**	**Time ↓**	**Gap ↓**	**Time ↓**
Gurobi[§]	0.00%	0.90s	0.00%	276.66s	0.00%	3.34s	0.00%	290.23s	0.00%	60.61s	0.00%	300.21s
Meta-EGN[§]	**8.30%**	0.27s	**15.49%**	0.54s	1.56%	0.30s	4.40%	1.03s	—	—	—	—
DiffUCO[§]	18.25%	0.56s	—	—	—	—	—	—	**0.15%**	0.20s	**−1.69%**	0.65s
Greedy	16.47%	0.01s	24.16%	0.02s	2.21%	0.08s	1.44%	0.24s	5.47%	0.05s	3.70%	0.22s
CoRE ($T_R = 100$)	9.16%	0.75s	22.81%	11.90s	**0.18%**	0.78s	**0.21%**	4.93s	1.51%	0.30s	0.80%	1.04s

[§] Reported in Table 2 of Ma et al. (2025a), at the first reported configuration: DiffUCO: CE ($S = 1, F = 1$). Gurobi (Gurobi Optimization, LLC, 2024), Meta-EGN (Wang and Li, 2023), DiffUCO (Sanokowski et al., 2024). T_R: refinement iterations; S: samples; T_s: denoising steps; D_s: determination steps; F: diffusion-step multiplier. A dash denotes an unreported result; bold: best gap; underline: second-best gap, excluding Gurobi.

Table 20: Zero-shot transfer within constrained vertex selection. Mean gap to certified optimum (%). Both BQ and EBQ are absent from training and model selection.

		$N = 32$		$N = 64$	
Method	M	$BQ \downarrow$	$EBQ \downarrow$	$BQ \downarrow$	$EBQ \downarrow$
Multi-heuristic ($\tau = 0.2$)	32	0.723%	0.719%	0.807%	1.467%
CoRE-A	32	0.487%	0.467%	0.559%	0.929%
CoRE	32	0.547%	0.418%	0.588%	0.900%
Multi-heuristic ($\tau = 0.2$)	128	0.254%	0.302%	0.372%	0.782%
CoRE-A	128	0.205%	0.156%	0.273%	0.373%
CoRE	128	0.231%	0.170%	0.237%	0.380%

M: completed solutions per graph; τ: sampling temperature.