Rethinking Neural Vehicle Routing as Multi-Agent Decision Making

Jiwoo Son [*1] **Chuanbo Hua** [*2] **Federico Berto** [3] **Bongseok Kim** [4] **Jinkyoo Park** [2 1]

Abstract

Vehicle Routing Problems (VRPs) are a fundamental class of combinatorial optimization problems with broad applications in logistics and mobility. A VRP inherently involves multiple vehicles cooperating to serve a shared set of customers. However, the dominant neural paradigm, including recent VRP foundation models, casts it as a single-agent sequential decision-making process in which a single policy constructs routes one at a time. We rethink VRPs from a multi-agent perspective and observe that this framing flattens a relational problem into a temporal one, hiding the cross-agent state, intent, and allocation structure that the problem naturally exposes. Our proposed approach jointly represents agents and actions, lets agents communicate during concurrent route construction, and adapts multi-start rollouts to the multi-agent setting. Our method matches or outperforms state-of-the-art neural baselines on cross-task VRPs and generalizes substantially better to unseen distributions and tasks. Our findings suggest that the multi-agent structure of VRPs is an underexplored axis for understanding and advancing neural combinatorial optimization.

1. Introduction

The Vehicle Routing Problem (VRP) is a cornerstone of combinatorial optimization, with direct economic stakes in logistics, last-mile delivery, ride-pooling, and on-demand mobility (Dantzig & Ramser, 1959; Toth & Vigo, 2014). Real-world instances combine a fleet of vehicles with rich side constraints such as capacities, time windows, route-length limits, and backhauls, and the number of practically relevant variants grows accordingly. Over the past decades, operations researchers have developed a vast catalogue of exact methods and handcrafted heuristics for these problems, including savings algorithms, large-neighborhood search, and metaheuristics tuned per variant (Toth & Vigo, 2014). Designing or adapting such a method to a new variant remains costly, as it requires deep domain expertise and substantial engineering effort for each change in constraints.

Neural combinatorial optimization (NCO) has emerged as a way to amortize this design effort by learning solvers directly from data. Foundational works as Pointer Networks (Vinyals et al., 2015) and the Attention Model (Kool et al., 2019) cast VRP as a sequential construction task and learn a policy with reinforcement learning, with POMO (Kwon et al., 2020) and Sym-NCO (Kim et al., 2022) extending sequential construction by exploiting symmetries with multi-start rollouts. More recent VRP *foundation models* train a single backbone over many CVRP variants to transfer across constraints (Berto et al., 2024b; Zhou et al., 2024), and parallel decoders begin to emit actions for several vehicles per step (Berto et al., 2024a; Son et al., 2024). Across these advances, the dominant recipe is the same: an attention-based encoder over the instance and an autoregressive decoder that builds a solution token by token.

Yet the problem itself is intrinsically multi-agent. A VRP instance is only well posed when more than one vehicle is available, and a solution is a partition of customers into a set of tours, one per active vehicle, that is jointly feasible and short in aggregate. Despite this, the prevailing neural recipe treats VRP as a single sequential decision process: one policy emits one customer at a time and returns to the depot to start a new tour, so the fleet plan is recovered only as the byproduct of a long serialized trajectory. This single-agent framing flattens three structural axes of the problem. *State* collapses to the trajectory of the currently active vehicle rather than the joint state of the fleet; *intent* cannot be shared, so two vehicles approaching the same time-windowed region cannot anticipate one another;

[*]Equal contribution [1]Omelet, Seoul, Korea [2]KAIST, Daejeon, Korea [3]Radical Numerics, San Francisco, USA [4]POSCO DX, Seongnam, Korea. Correspondence to: Jinkyoo Park <jinkyoo.park@kaist.ac.kr>.

Accepted at the Workshop on Graph Foundation Models: A New Era for Graph Machine Learning at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

and *allocation*, the assignment of customers to vehicles, is conflated with sequencing into one ordering. Recent parallel decoders relax the serialization but still pose and train the underlying decision problem as if a single driver were swapping identities between actions.

We argue that this mismatch is not a benign modeling choice: it caps quality on hard variants, narrows the construction strategies a policy can express, and weakens out-of-distribution generalization, since a serialized policy memorizes route shapes rather than learning a transferable allocation rule (Bi et al., 2022). We therefore rethink neural VRP as multi-agent decision making and ask what changes when the multi-agent view is taken seriously across encoding, decoding, and rollout. Concretely, we introduce a joint $(\text{agent}, \text{node})$ decision space where "which vehicle moves next" and "where it goes" are decided together, an inter-agent communication block so that each step conditions on the joint fleet state, and a multi-agent multi-start rollout that draws diversity from the agent allocation itself rather than from forced first customers as in single-agent POMO (Kwon et al., 2020). These three moves are not independent tricks but consequences of the same view, and we instantiate them as a single foundation-model-style backbone trained over 16 CVRP variants.

We summarize our contributions as follows:

- We diagnose the prevailing single-agent recipe for neural VRP and show that it collapses three structural axes of the problem, namely state, intent, and allocation, that a multi-agent reformulation naturally separates.
- We translate this diagnosis into a foundation-model-style backbone with three coupled architectural moves: a joint $(\text{vehicle}, \text{customer})$ decision space, inter-agent communication during decoding, and an allocation-driven multi-start rollout.
- Through controlled ablations we show that the joint decision space and the allocation-driven multi-start are each individually load-bearing, accounting for the bulk of the empirical gain.
- We demonstrate that a single trained backbone achieves state-of-the-art performance across in-distribution variants, unseen variant compositions, and real-world out-of-distribution instances, without per-variant fine-tuning.

2. Related Work

Neural construction solvers for VRP. Learning-based construction solvers form the main line of neural VRP research that we revisit. Pointer Networks introduced the autoregressive view of routing as sequence prediction (Vinyals et al., 2015), and the Attention Model recast it with a transformer-style encoder–decoder trained by REINFORCE (Kool et al., 2019). POMO exploited the symmetry of VRP solutions through multi-start rollouts and a shared baseline (Kwon et al., 2020), and Sym-NCO further leveraged problem symmetries to regularize the policy (Kim et al., 2022). A more recent thread scales construction to larger instances by replacing the single-decoder rollout with heavy-decoder or batch-policy designs that solve thousands of nodes in one shot. (Drakulic et al., 2023; Luo et al., 2023) Distinct as these designs are, they share an implicit assumption: VRP is encoded as one agent that switches between vehicle identities along a single serialized trajectory, with the fleet plan recovered only as the byproduct of depot returns.

Foundation and multi-task neural solvers for VRP. A second line scales construction across many CVRP variants within a single model and is the closest baseline to ours. MTPOMO (Liu et al., 2024) trains a single POMO-style backbone over a mixture of CVRP variants with attribute-conditioned inputs, MVMoE replaces the dense backbone with a mixture-of-experts to share capacity across variants while specializing per attribute (Zhou et al., 2024), and RouteFinder unifies attribute composition and mixed-batch training so that one solver covers a broad family of CVRPs (Berto et al., 2024b). Subsequent work pushes this regime further with hierarchical and unified routing models. (Goh et al., 2025; Jari et al., 2025; Jiang et al., 2024; Pan et al., 2025), The effort to scale across variants is concentrated on the encoder side, through attribute embeddings, MoE routing, and mixed-batch reward normalization, and on the training protocol, through joint sampling across variants. The decoder, however, remains single-agent: one policy emits one customer at a time, and "which vehicle moves" is fixed by the depot-return convention rather than learned.

Multi-agent neural combinatorial optimization. A third line treats the fleet as multiple coordinated agents, and is the closest neighbor we must distinguish ourselves from. Decentralized attention networks coordinate agents for min-max multi-TSP (Cao et al., 2021), Equity-Transformer enforces balanced workload across vehicles in min-max routing (Son et al., 2024), DPN adopts a similar perspective for partition-style routing, (Zheng et al., 2024) and a multi-agent neural rewriter learns local edits driven by per-agent state. Centralized training, decentralized execution variants extend this to communication-aware policies. Most directly, PARCO introduces parallel autoregressive decoding with a learned conflict handler so that multiple vehicles act in the same step (Berto et al., 2024a). Two patterns characterize this body of work: (i) the multi-agent

treatment is problem-specific, designed for min-max or partitioned routing rather than the broad CVRP family; or (ii) multi-agent decoding is used as an efficiency trick that parallelizes emission while leaving encoding, training, and rollout single-agent. PARCO is the closest point of comparison, and our differences with it are framing rather than mechanism: PARCO targets parallel decoding efficiency on fixed variants and resolves cross-agent conflicts at decode time, whereas we adopt the multi-agent view as the organizing principle of a foundation-model-scale VRP solver, with a joint (agent, node) decision space, inter-agent communication in the encoder, and an allocation-driven multi-start rollout. These works open the multi-agent direction but leave open whether multi-agent structure is merely a decoding-time trick or a foundational view that should reshape how we encode, decode, and roll out across the VRP family. We argue for the latter.

3. A Multi-Agent View of Neural Vehicle Routing

3.1. Problem setup

A vehicle routing problem (VRP) instance is specified by a depot 0, a customer set $\mathcal{V} = \{1, \ldots, N\}$ with per-customer demands and attributes (such as time windows or backhaul flags), and a fleet $\mathcal{K}$ of vehicles sharing resources such as capacity, time budget, and distance budget. A feasible solution partitions $\mathcal{V}$ into $|\mathcal{K}|$ tours $\{r_k\}_{k\in\mathcal{K}}$, each starting and ending at the depot, satisfying the active side constraints (capacity, time windows, distance limit, open route, backhaul precedence, and so on), and minimizing total tour length $\sum_{k\in\mathcal{K}} \mathrm{len}(r_k)$. This formulation covers CVRP and the standard families of CVRP variants: a particular variant is selected by which side constraints are active.

3.2. Single-agent versus multi-agent decoding

Consider an instance with two vehicles approaching a region with two unvisited customers $\{u, v\}$, and let $h_u^{\mathrm{n}}, h_v^{\mathrm{n}}$ be their encoder embeddings and q_1, q_2 the per-vehicle context queries.

Single-agent decoding. The dominant neural construction recipe picks one customer at a time for a single *active* vehicle $k^\star$:

$$\pi_{\mathrm{single}}(a) \;\propto\; \exp\big(\langle q_{k^\star},\, h_a^n\rangle\big), \quad a \in \{u, v\}, \tag{1}$$

where $k^\star$ is set by the depot-return convention: a customer token extends the active tour, the depot token closes it and starts a new one, and the fleet plan is recovered only as a byproduct of where depot tokens fall in the trajectory. The other vehicle's query $q_{3-k^\star}$ does not enter Equation (1), and its choice is committed only after $k^\star$'s realized action is appended.

Multi-agent decoding. The multi-agent view emits one (vehicle, customer) pair at a time from a joint softmax over the fleet:

$$\pi_{\mathrm{multi}}\big((k, a)\big) \;\propto\; \exp\big(\langle q_k,\, h_a^n\rangle\big), \quad (k, a) \in \mathcal{K} \times \{u, v\}. \tag{2}$$

Both vehicles' queries appear in the same partition function, so the policy chooses jointly which vehicle moves and where it goes.

What changes. The two recipes coincide when only one vehicle is in flight but differ structurally as soon as more than one vehicle is active, which is the regime that defines VRP. We treat Equation (1) as the common substrate of Pointer Networks (Vinyals et al., 2015), the Attention Model (Kool et al., 2019), POMO (Kwon et al., 2020), Sym-NCO (Kim et al., 2022), and recent VRP foundation models (Berto et al., 2024b; Zhou et al., 2024), and Equation (2) as the distribution this paper organizes its architecture around.

3.3. What does the multi-agent view bring to neural VRP?

Once decoding ranges over the joint (vehicle, customer) space, three structural advances become available.

Joint state across the fleet. Every decision is conditioned on the joint state of the entire fleet rather than on the trajectory of a single active vehicle. Each vehicle's partial route, current position, remaining capacity, and elapsed time are first-class context for the decoder, instead of entering only through a feasibility mask. This lets the policy reason about quantities such as which vehicle is closest to a customer cluster, which vehicle has slack capacity, or which vehicle has the loosest remaining time budget, as part of the same softmax that picks the next move.

Shared intent before commitment. Because the joint softmax of Equation (2) couples the queries of all vehicles in a single partition function, an agent's choice is shaped by where other agents are about to go, not only by where they have already gone. When two vehicles approach the same time-windowed cluster, they can split the cluster in advance rather than resolving the contention reactively. Intent is exchanged before commitment, and the resulting rollouts are coordinated rather than myopically sequential.

Explicit customer-to-agent allocation. The joint softmax also makes the customer-to-vehicle allocation $\alpha : \mathcal{V} \to \mathcal{K}$ a first-class output of the policy. Allocation and ordering are weakly coupled: many high-quality solutions share an allocation but differ in ordering, and customers in the same geographic or temporal region tend to keep their

allocation across instances and across variants more reliably than they keep their ordering. By representing allocation explicitly, the model can place an inductive bias on the more transferable axis of the problem, which is what we later exploit for rollout diversity and for cross-task generalization.

3.4. Design principles

These observations translate into three design principles that organize the rest of the paper. The action at every decoding step should range over the joint vehicle-customer space rather than over customers alone, so that "which vehicle moves next" becomes part of the learned distribution and the customer-to-vehicle allocation is emitted explicitly. The per-vehicle queries that enter that joint distribution should be computed from a representation of the entire fleet, not only of the active vehicle, so that vehicles can share intent inside the same softmax that picks the next move. And the multi-start rollouts that drive REINFORCE training should diversify over the allocation α rather than over forced first customers as in POMO (Kwon et al., 2020), so that the policy explores qualitatively distinct fleet plans rather than reorderings of one.

The principles are compositional rather than independent. Joint decoding without inter-agent context reduces to a richer mask, since the per-vehicle queries would still be uninformed by the rest of the fleet. Allocation-driven multi-start has nothing to vary if the action space contains no vehicle index. And inter-agent communication has no joint distribution to feed if decoding remains single-agent. A faithful instantiation of the multi-agent view therefore adopts the three principles together.

4. Method

We instantiate the three design principles of Section 3.4 as a single backbone. Section 4.1 introduces the encoder that embeds vehicles as nodes in the same sequence as customers, and Sections 4.2 to 4.4 then realize P1, P2, and P3, respectively. Two architectural components are inherited unchanged from prior work: the CADA constraint prompt (Li et al., 2025) in the encoder, and PARCO's Communication Layers (Berto et al., 2024a) in the communication block. Our contribution lies in the joint $\mathcal{A}_t = \mathcal{K}_t \times \mathcal{T}_t$ consumption of encoder and communication outputs, and in the allocation-driven multi-start rollout that drives training. A schematic of the architecture is given in Figure 1.

4.1. Encoder

The encoder processes a VRP instance together with its active fleet as a single token sequence. Following the agent-as-node convention of Equity-Transformer (Son et al., 2024) and PARCO (Berto et al., 2024a), each of the K vehicles is materialized as an *agent node* carrying that vehicle's per-agent attributes (capacity Q_k, active flag, slot index) on top of the shared depot position, so agents and the N customers share one self-attention. Inputs to a transformer encoder $\mathcal{E}(\cdot)$ are a constraint flag $\tilde{\mathbf{h}} \in \{0,1\}^4$ indicating the active side constraints following CADA (Li et al., 2025), the K agent nodes $\{\mathbf{a}_k\}_{k=1}^K$, and the N customer features $\{\mathbf{h}_i\}_{i=1}^N$:

$$\mathbf{H} \;=\; \mathcal{E}\Big(\tilde{\mathbf{h}},\ \{\mathbf{a}_k\}_{k=1}^K,\ \{\mathbf{h}_i\}_{i=1}^N\Big), \qquad \mathbf{H} \in \mathbb{R}^{(K+N)\times D}. \tag{3}$$

We write h_v^{n} for the row of $\mathbf{H}$ corresponding to token v, which may be either a customer or an agent node. Embedding agents alongside customers lets self-attention propagate agent-aware context into customer embeddings, and vice versa, before any decoding decision is made.

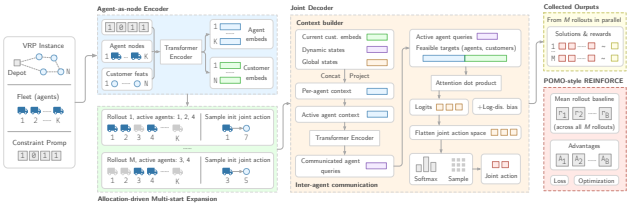

Figure 1. Architecture overview. A constraint-prompted encoder produces node embeddings for both customers and agent nodes; an inter-agent communication block fuses per-vehicle state at every decoding step; a joint softmax over $\mathcal{A}_t = \mathcal{K}_t \times \mathcal{T}_t$ emits one (k, v) per step; allocation-driven multi-start rollouts diversify over vehicle-to-customer allocation rather than over forced first customers.

4.2. Joint (vehicle, target) decision space

To restore explicit allocation (Section 3.3, principle P1), the policy acts in a joint decision space $\mathcal{A}_t = \mathcal{K}_t \times \mathcal{T}_t$, where $\mathcal{K}_t$ is the set of currently active vehicles and $\mathcal{T}_t$ contains both the agent nodes and the customers in $\mathcal{V}_t$ still feasible under the active side constraints. The mask confines an agent-node target to the vehicle that owns it, so the action (k, a_k) closes vehicle k's own tour while (k, v) with a customer v extends it. Every step emits one joint action $a_t = (k_t, v_t) \in \mathcal{A}_t$, so "which vehicle moves where, or closes its tour" is a single learned decision.

Logits. Using the encoder embeddings h_v^{n} from Section 4.1, and letting h_k^{a} denote a per-vehicle query vector summarizing vehicle k's current state (one such vector per active vehicle; the instantiation we use is described in Section 4.3), we form joint logits

$$\ell_{k,v} \;=\; \frac{\langle W_q\, h_k^{\mathrm{a}},\ h_v^{\mathrm{n}}\rangle}{\sqrt{d}} \;+\; \log\!\left(\frac{1}{\mathrm{dist}(\mathrm{pos}_k, v) + \varepsilon}\right), \tag{4}$$

where pos_k denotes the current node of vehicle k and the second term is a zero-parameter, gradient-detached dis-

tance bias that concentrates early-training probability on nearby targets and keeps gradients informative before W_q is trained. The policy is the masked softmax over the flattened $\mathcal{A}_t$,

$$\pi_\theta(a_t \mid s_t) \;=\; \frac{\exp(\ell_{k,v})\,\mathbb{1}[(k,v)\in\mathcal{A}_t]}{\sum_{(k',v')\in\mathcal{A}_t}\exp(\ell_{k',v'})}, \tag{5}$$

whose mask enforces both feasibility under the active side constraints and the rollout-specific active fleet size. Sampling or argmax over this joint distribution yields one (k, v) pair per step; k varies across steps according to the joint distribution rather than according to a depot-return rule. When $\mathcal{K}_t = \{k_t^\star\}$, Equation (5) recovers the single-active-vehicle special case corresponding to the single-agent recipe of Section 3.2.

4.3. Inter-agent communication

To enable intent sharing (Section 3.3, principle P2), an inter-agent communication block runs before each decoding step. We use PARCO's Communication Layers unchanged (Berto et al., 2024a): our contribution is not the communication primitive itself but how the communicated vehicle states are consumed: as queries that enter the joint action softmax rather than as inputs to parallel per-agent decoders followed by post-hoc conflict resolution.

Per-vehicle context is assembled by concatenating the encoder embedding of the vehicle's current node, the embedding of its own agent node, a projection of its dynamic state (remaining capacity, elapsed time, accumulated route length), and a broadcast global progress indicator:

$$\bar{h}^{\mathrm{a}}{}_k \;=\; \mathrm{Proj}\Big(h^{\mathrm{n}}_{\mathrm{pos}_k} \parallel h^{\mathrm{n}}_{a_k} \parallel \phi_{\mathrm{state}}(q_k^{\mathrm{rem}}, \tau_k, L_k) \parallel \phi_{\mathrm{glob}}(\rho_t)\Big) \tag{6}$$

A single Transformer block mixes information across the active fleet,

$$\{\tilde{h}^{\mathrm{a}}{}_k\}_{k\in\mathcal{K}_t} \;=\; \mathrm{TB}\big(\{\bar{h}^{\mathrm{a}}{}_k\}_{k\in\mathcal{K}_t}\big)\,, \tag{7}$$

and the output $\tilde{h}^{\mathrm{a}}{}_k$ replaces h^{a}_k in Equation (4). The block is rerun at every decoding step so that communication tracks the freshly updated joint state.

4.4. Allocation-driven multi-start rollout

To exploit the allocation axis for diversity (Section 3.3, principle P3), we adopt an *allocation-driven multi-start*, a multi-start scheme whose M rollouts differ in their customer-to-vehicle assignment α rather than merely in visit order. Two mechanisms drive this diversity.

Per-rollout capacity scaling. Each rollout $m \in \{1, \ldots, M\}$ is assigned a capacity scaling factor $c_m \sim \mathcal{U}[c_{\mathrm{low}}, c_{\mathrm{high}}]$ drawn from a fixed, task-independent interval. c_m acts as a rollout-local activation signal: the number of vehicles participating in rollout m is determined by comparing the total demand against $c_m \cdot Q$, so that a smaller c_m keeps more vehicles active and a larger c_m concentrates customers onto fewer vehicles. The underlying instance is unchanged; feasibility and the reported cost of each rollout are always evaluated under the original per-vehicle capacity Q, not under $c_m \cdot Q$. Single-agent serialized decoding has no per-rollout active-fleet count to vary, since the number of active vehicles is an emergent byproduct of depot-return positions. This axis is therefore specific to the multi-agent view.

Initial joint action sampling. At $t = 1$, each rollout draws its initial joint action (k_0^m, v_0^m) by multinomial sampling from the joint distribution of Equation (5), rather than forcing a designated agent onto a deterministic first customer as in POMO (Kwon et al., 2020). Two rollouts can therefore begin with the same first customer but assign it to different vehicles.

Training. We use $M = N$ rollouts per instance, matching POMO's budget. All rollouts share the encoder pass and policy parameters. The policy is trained with REINFORCE, using the mean reward across the M rollouts as a per-instance baseline and an exponential-moving-average normalization of reward magnitudes across variants to stabilize the mixed-batch signal.

5. Experiments

We evaluate the multi-agent solver on three questions. First, does the multi-agent view match or improve foundation-model-style baselines on the variants seen at training time? Second, does it transfer better to unseen customer distributions, unseen problem sizes, and unseen variants? Third, which of the three principles of Section 3.4 is each empirical gain attributable to?

5.1. Setup

Variants and instances. We use the 16-variant CVRP family, formed by all subsets of {Open route, Time Windows, Distance Limit, Backhaul}. Training instances are drawn at sizes $N \in \{50, 100\}$ with active fleets up to $|\mathcal{K}| = 8$ and customers sampled uniformly in the unit square; the precise generators follow the RouteFinder protocol (Berto et al., 2024b). Test sets have 1,000 instances per variant per size.

Baselines. We compare against five neural baselines and a strong classical reference: (i) MTPOMO (Liu et al., 2024), a multi-task POMO backbone shared across variants with attribute-conditioned inputs; (ii) MVMoE (Zhou et al., 2024), which adds a mixture-of-experts to the multi-

task backbone; (iii) RouteFinder (Berto et al., 2024b), a foundation-model backbone with unified attribute composition, whose encoder prompt mechanism we inherit; (iv) CaDA (Li et al., 2025), the closest encoder baseline using a constraint-aware dual-attention encoder; (v) CCL (Gui et al., 2026), the strongest foundation-VRP baseline at the time of writing. As a classical reference, we use PyVRP/HGS (Wouda et al., 2024; Vidal, 2022), which provides a strong non-learned upper bound on solution quality.

Hardware and training. All experiments are run on 8× NVIDIA A100 80GB GPUs. Training takes roughly 1 day at $N = 50$ and 2.5 days at $N = 100$. We use a training batch size of 256 and a test batch size of 128, optimized with Adam at a learning rate of 3×10^{-4} and weight decay 1×10^{-6}. A multi-step learning-rate scheduler decays by a factor of 0.1 at epochs 270 and 295, and gradients are clipped to norm 1. Training is conducted for a total of 300 epochs. Further architectural and procedural details are deferred to Appendix C.

5.2. Experiment Results

In-distribution comparison. Table 1 reports the average gap to HGS across all 16 variants at $N \in \{50, 100\}$. Our multi-agent solver attains the best neural gap on 21 of the 32 (variant, size) cells, against 11 for the strongest baseline (CCL), and wins every variant whose constraint set contains time windows, establishing a new state of the art on the 16-variant CVRP family. The wins concentrate where inter-agent coupling is strongest: at $N = 50$, ours and CCL split exactly along the time-window axis (8 vs. 8), and at $N = 100$ the larger fleet amplifies coupling and extends our lead to 13 of 16 variants.

Cross-task generalization. Cross-task evaluation is the cleanest empirical test of our central claim that allocation is the more transferable axis of the problem (Section 3.3): held-out variant compositions force a model to recombine constraint behaviors it has never decoded together, and a policy that has only memorized variant-specific route shapes has nothing to fall back on. Table 2 summarizes performance on 32 unseen variants held out from training (see Appendix D.1 for per-task results). Our solver attains the lowest average gap to HGS at both sizes and is the best neural method on $28/32$ variants at $N = 50$ and on all $32/32$ at $N = 100$, more than halving the gap of the strongest baseline (CCL). Table 3 decomposes the 32 unseen variants into four 8-variant sub-families along the multi-depot (MD) and mixed-backhaul (MB) axes at $N = 100$. The cross-task lead grows monotonically with how disjoint the OOD constraint composition is from the training mix: the gap reduction over CCL is smallest on MB-only (∼3.7pp), comparable on MD-only and MD+TW (∼23pp and ∼20pp), and largest on MD+MB where the two unseen attributes coincide (∼24pp), with the same monotone pattern at $N = 50$. The lead is qualitatively different from the in-distribution comparison: where we and CCL traded individual cells within fractions of a percentage point in Section 5.2, the cross-task gap to CCL widens by more than an order of magnitude, consistent with the prediction that an explicit allocation policy generalizes compositionally while serialized policies do not (Bi et al., 2022).

Cross-distribution generalization. The cross-task experiment above stresses the transferability claim along the constraint axis; CVRPLib-X stresses it along a complementary axis: the input geometry itself. We evaluate zero-shot transfer on CVRPLib-X, a benchmark of 65 real-world CVRP instances whose clustered customer layouts, asymmetric depot placements, and sizes up to $N = 1000$ depart sharply from the uniform-square training distribution and would expose any policy that has overfit to a stylized customer prior. On these real-world instances our solver attains the lowest average gap to optimal (8.29%) and is the best neural method on 37 of the 65 instances, more than the other three baselines combined; per-instance results are in Table 8 of Appendix D.2. The lead concentrates at scale: ours wins 15 of the 33 instances with $N \leq 250$ (where all four neural methods cluster within a few percentage points and the per-instance winner is largely noise) and 22 of the 32 instances with $N \geq 500$. On the largest cells, CCL's gap to optimal repeatedly exceeds 30–50 percentage points while ours stays below 30% on every cell, indicating that the allocation-driven policy retains structure precisely where the serialized recipe is known to degrade (Bi et al., 2022). Table 4 summarizes these results.

5.3. Ablation Studies

We isolate two of the design principles from Section 3.4: the allocation-driven training-time multi-start (P3) and the joint $(\text{vehicle}, \text{target})$ action (P1). Table 5 reports the mean gap to HGS over the 16 in-distribution variants at $N = 50$; per-task numbers are in Appendix D.3. Removing multi-start entirely costs 16.2 percentage points, augmentation alone closes most of that gap but still leaves 0.4pp, and replacing our scheme with POMO's forced first-customer multi-start leaves 0.2pp on the table. On the action axis, substituting our joint $(\text{vehicle}, \text{target})$ softmax with PARCO's parallel-action decoder costs 8.1pp on average, with the deterioration uniform across all 16 in-distribution variants (the joint softmax wins every per-task cell of Table 10, and the parallel-action gap exceeds 13% on the open-route and pure-backhaul subsets), indicating that simultaneous selection without joint normalization underuses the available coordination signal.

Table 1. Performance on 16 seen in-distribution tasks. * denotes the strong baseline used to compute the gap. Best neural approach is highlighted in **bold**.

	Methods	$N = 50$			$N = 100$				Methods	$N = 50$			$N = 100$		
		Obj. ↓	Gap ↓	Time ↓	Obj. ↓	Gap ↓	Time ↓			Obj. ↓	Gap ↓	Time ↓	Obj. ↓	Gap ↓	Time ↓
CVRP	HGS-PyVRP	10.372	*	10.4m	15.628	*	20.8m	VRPTW	HGS-PyVRP	16.031	*	10.4m	25.423	*	20.8m
	MTPOMO	10.520	1.423%	2s	15.941	2.030%	8s		MTPOMO	16.419	2.423%	2s	26.433	3.962%	9s
	MVMoE	10.499	1.229%	3s	15.888	1.693%	11s		MVMoE	16.400	2.298%	3s	26.390	3.789%	11s
	RF-TE	10.502	1.257%	2s	15.860	1.524%	8s		RF-TE	16.341	1.933%	2s	26.228	3.154%	8s
	CaDA	10.505	1.287%	2s	15.843	1.376%	8s		CaDA	16.312	1.745%	1s	26.169	2.935%	9s
	CCL	**10.473**	**0.977%**	5s	15.823	1.248%	19s		CCL	16.190	0.979%	5s	25.913	1.928%	21s
	Ours	10.492	1.152%	4s	**15.821**	**1.232%**	16s		Ours	**16.159**	**0.799%**	4s	**25.787**	**1.431%**	18s
OVRP	HGS-PyVRP	6.507	*	10.4m	9.725	*	20.8m	OVRPTW	HGS-PyVRP	10.510	*	10.4m	16.926	*	20.8m
	MTPOMO	6.717	3.194%	2s	10.216	5.028%	8s		MTPOMO	10.676	1.558%	2s	17.442	3.022%	9s
	MVMoE	6.705	3.003%	3s	10.177	4.617%	11s		MVMoE	10.674	1.541%	3s	17.416	2.870%	12s
	RF-TE	6.682	2.658%	2s	10.115	3.996%	8s		RF-TE	10.645	1.264%	2s	17.328	2.352%	9s
	CaDA	6.677	2.585%	1s	10.095	3.805%	8s		CaDA	10.630	1.122%	1s	17.283	2.109%	9s
	CCL	**6.636**	**1.957%**	5s	10.068	3.527%	20s		CCL	10.569	0.543%	6s	17.123	1.164%	21s
	Ours	6.667	2.460%	4s	**10.063**	**3.476%**	17s		Ours	**10.557**	**0.447%**	5s	**17.081**	**0.921%**	18s
OVRPB	HGS-PyVRP	6.898	*	10.4m	10.335	*	20.8m	OVRPBTW	HGS-PyVRP	11.669	*	10.4m	19.156	*	20.8m
	MTPOMO	7.105	2.973%	2s	10.882	5.264%	8s		MTPOMO	11.823	1.307%	3s	19.656	2.592%	9s
	MVMoE	7.089	2.744%	3s	10.841	4.869%	11s		MVMoE	11.816	1.245%	4s	19.637	2.499%	13s
	RF-TE	7.065	2.385%	2s	10.774	4.233%	8s		RF-TE	11.790	1.027%	2s	19.555	2.062%	9s
	CaDA	7.064	2.377%	1s	10.739	3.909%	8s		CaDA	11.775	0.898%	2s	19.495	1.770%	9s
	CCL	**7.008**	**1.568%**	5s	10.666	3.203%	19s		CCL	11.721	0.436%	6s	19.348	1.002%	21s
	Ours	7.033	1.952%	4s	**10.664**	**3.186%**	19s		Ours	**11.720**	**0.436%**	4s	**19.321**	**0.862%**	19s
OVRPBL	HGS-PyVRP	6.899	*	10.4m	10.335	*	20.8m	OVRPBLTW	HGS-PyVRP	11.668	*	10.4m	19.156	*	20.8m
	MTPOMO	7.112	3.053%	2s	10.888	5.318%	8s		MTPOMO	11.823	1.315%	3s	19.658	2.602%	9s
	MVMoE	7.094	2.799%	3s	10.847	4.929%	11s		MVMoE	11.816	1.249%	4s	19.640	2.514%	12s
	RF-TE	7.068	2.417%	2s	10.778	4.266%	8s		RF-TE	11.789	1.017%	2s	19.554	2.061%	9s
	CaDA	7.062	2.339%	1s	10.741	3.928%	8s		CaDA	11.777	0.914%	2s	19.497	1.780%	9s
	CCL	**7.009**	**1.569%**	5s	10.681	3.348%	20s		CCL	11.721	0.442%	6s	19.346	0.992%	22s
	Ours	7.032	1.930%	4s	**10.661**	**3.153%**	17s		Ours	**11.719**	**0.441%**	4s	**19.323**	**0.875%**	19s
OVRPL	HGS-PyVRP	6.507	*	10.4m	9.724	*	20.8m	OVRPLTW	HGS-PyVRP	10.510	*	10.4m	16.926	*	20.8m
	MTPOMO	6.720	3.248%	2s	10.224	5.112%	8s		MTPOMO	10.677	1.572%	2s	17.442	3.020%	9s
	MVMoE	6.706	3.028%	3s	10.184	4.693%	11s		MVMoE	10.677	1.564%	3s	17.418	2.880%	12s
	RF-TE	6.683	2.680%	2s	10.121	4.054%	8s		RF-TE	10.646	1.267%	2s	17.328	2.352%	9s
	CaDA	6.680	2.623%	1s	10.093	3.795%	8s		CaDA	10.631	1.133%	1s	17.280	2.092%	9s
	CCL	**6.637**	**1.968%**	5s	10.067	3.528%	20s		CCL	10.569	0.546%	6s	17.123	1.164%	22s
	Ours	6.667	2.464%	4s	**10.062**	**3.477%**	17s		Ours	**10.557**	**0.447%**	5s	**17.081**	**0.918%**	18s
VRPB	HGS-PyVRP	9.687	*	10.4m	14.377	*	20.8m	VRPBTW	HGS-PyVRP	18.292	*	10.4m	29.467	*	20.8m
	MTPOMO	10.036	3.596%	2s	15.102	5.052%	8s		MTPOMO	18.649	1.938%	2s	30.478	3.426%	9s
	MVMoE	10.007	3.292%	3s	15.023	4.505%	10s		MVMoE	18.632	1.841%	3s	30.437	3.284%	12s
	RF-TE	9.979	3.000%	2s	14.935	3.906%	8s		RF-TE	18.573	1.517%	2s	30.249	2.641%	9s
	CaDA	9.979	3.010%	1s	14.910	3.707%	8s		CaDA	18.543	1.361%	1s	30.174	2.398%	9s
	CCL	**9.916**	**2.352%**	5s	**14.882**	**3.513%**	19s		CCL	18.430	0.738%	6s	29.911	1.506%	21s
	Ours	9.945	2.659%	4s	14.885	3.533%	16s		Ours	**18.427**	**0.737%**	4s	**29.839**	**1.262%**	18s
VRPBL	HGS-PyVRP	10.186	*	10.4m	14.779	*	20.8m	VRPBLTW	HGS-PyVRP	18.361	*	10.4m	29.026	*	20.8m
	MTPOMO	10.679	4.760%	2s	15.718	6.294%	8s		MTPOMO	19.001	3.486%	3s	30.948	6.622%	9s
	MVMoE	10.639	4.384%	3s	15.642	5.771%	11s		MVMoE	18.983	3.388%	3s	30.892	6.429%	12s
	RF-TE	10.569	3.713%	2s	15.523	5.008%	8s		RF-TE	18.910	2.990%	2s	30.705	5.785%	9s
	CaDA	10.576	3.776%	1s	15.490	4.811%	8s		CaDA	18.894	2.902%	1s	30.620	5.491%	9s
	CCL	**10.484**	**2.883%**	5s	**15.407**	**4.249%**	19s		CCL	18.773	2.244%	6s	30.366	4.616%	21s
	Ours	10.522	3.299%	4s	15.437	4.455%	16s		Ours	**18.760**	**2.171%**	4s	**30.273**	**4.296%**	18s
VRPL	HGS-PyVRP	10.587	*	10.4m	15.766	*	20.8m	VRPLTW	HGS-PyVRP	16.356	*	10.4m	25.757	*	20.8m
	MTPOMO	10.775	1.733%	2s	16.157	2.483%	8s		MTPOMO	16.832	2.877%	2s	26.913	4.455%	9s
	MVMoE	10.753	1.525%	3s	16.099	2.113%	11s		MVMoE	16.817	2.783%	3s	26.866	4.272%	12s
	RF-TE	10.747	1.485%	2s	16.057	1.858%	8s		RF-TE	16.728	2.248%	2s	26.706	3.645%	9s
	CaDA	10.749	1.505%	1s	16.036	1.713%	8s		CaDA	16.709	2.130%	1s	26.631	3.394%	9s
	CCL	**10.710**	**1.145%**	5s	**16.009**	**1.541%**	19s		CCL	16.579	1.333%	6s	26.366	2.365%	20s
	Ours	10.731	1.355%	4s	16.012	1.561%	16s		Ours	**16.542**	**1.140%**	4s	**26.221**	**1.801%**	18s

Table 2. Cross-task generalization on 32 unseen out-of-distribution variants. We report the average gap to HGS and the number of variants on which each method is the best neural baseline. Per-task results are in Tables 6 and 7.

	$N = 50$		$N = 100$	
Method	Avg. Gap ↓	# Best ↑	Avg. Gap ↓	# Best ↑
RF-TE	36.529%	0/32	41.144%	0/32
CaDA	34.866%	4/32	39.834%	0/32
CCL	27.588%	0/32	34.892%	0/32
Ours	**13.965%**	**28/32**	**17.134%**	**32/32**

Table 3. Cross-task generalization decomposed by attribute family at $N = 100$. The 32 unseen variants partition into four 8-variant sub-families along the multi-depot (MD) and mixed-backhaul (MB) axes; each cell averages the gap to HGS over the 8 variants in the sub-family. Best neural method per row in **bold**.

Sub-family (8 variants)	RF-TE	CaDA	CCL	Ours
MB only (no MD)	12.18%	11.32%	13.18%	**9.47%**
MD only (no MB, no TW)	51.08%	53.85%	39.18%	**16.03%**
MD + TW (no MB)	43.04%	35.38%	35.94%	**16.23%**
MD + MB	58.27%	58.79%	51.26%	**27.03%**

Table 4. Cross-distribution generalization on CVRPLib-X (65 real-world instances). Average gap to optimal and the number of instances on which each method is the best neural baseline, decomposed by instance size. Per-instance results are in Table 8.

Method	Avg. Gap ↓	# Best ↑		
	(all 65)	$N \leq 250$	$N \geq 500$	Total
RF-TE	8.595%	6/33	3/32	9/65
CaDA	9.373%	3/33	5/32	8/65
CCL	14.356%	9/33	2/32	11/65
Ours	**8.286%**	**15/33**	**22/32**	**37/65**

Table 5. Ablation summary: mean gap to HGS over the 16 in-distribution variants at $N = 50$. Per-task numbers are in Appendix D.3.

Principle	Variant	Mean Gap ↓
Multi-start (P3)	no multi-start	17.632%
	augmentation only	1.830%
	POMO forced first-customer MS	1.610%
	allocation-driven MS (Ours)	**1.386%**
Action (P1)	parallel action (PARCO)	9.480%
	joint action (Ours)	**1.386%**

6. Discussion

Conceptual takeaway. We rethought neural VRP as multi-agent decision making and identified three structural axes, namely state, intent, and allocation, that the dominant single-agent recipe collapses (Section 3.3). Reading those diagnoses backward yielded a single design heuristic: when a combinatorial structure has explicit relational dimensions, neural solvers should separate those dimensions from the sequential dimension instead of folding them in. For VRP this materializes as a joint $(\mathcal{K}, \mathcal{V})$ decision space, inter-agent communication, and allocation-driven multi-start, and our experiments show that the resulting backbone matches or improves the strongest single-agent baselines and transfers more cleanly to unseen distributions, sizes, and variants. We view the principle as more general than the specific architecture: any cooperative combinatorial problem whose solution is a partition rather than a sequence is a candidate for the same separation.

Limitations and future work. The multi-agent treatment scales the rollout count and inter-agent attention with the fleet size, reduces to POMO at $|\mathcal{K}| = 1$, and has not been tested on $N > 1{,}000$ or on qualitatively new attributes such as pickup-and-delivery and split delivery. Diagnostic analyses we view as complementary, including a diversity decomposition of rollouts, a sweep over fleet size and constraint density, and a linear probe for same-vehicle representation structure, are deferred for space. Beyond VRP, the diagnosis of state, intent, and allocation transfers to scheduling, multi-robot task allocation, and ride-pooling, where we expect the joint decision space and allocation-driven rollout to apply with minor adaptation.

7. Conclusion

Posing VRP as multi-agent decision making is more than a stylistic preference; it changes what the policy can express. The dominant single-agent recipe collapses three structural axes of the problem, namely state, intent, and allocation, that a multi-agent view naturally separates, and our experiments confirm that this separation translates into consistent empirical gains both within and beyond the training distribution. We hope this work surfaces a design choice that prior neural solvers have left implicit – whether to encode relational structure separately from sequential structure – and that separating the two proves useful beyond the specific solver we instantiate here.

Impact Statement

Improved neural VRP solvers have direct application in logistics and last-mile delivery, sectors in which routing efficiency translates into reductions in fuel consumption and operating cost. The same techniques can be repurposed for

ride-pooling and on-demand mobility, where allocation decisions have additional welfare implications (driver workload, passenger waiting times) beyond the aggregate cost minimised here, and we expect responsible deployment to require evaluation against fairness and equity metrics that are out of scope for this paper.

References

Berto, F., Hua, C., Luttmann, L., Son, J., Park, J., Ahn, K., Kwon, C., Xie, L., and Park, J. PARCO: Learning parallel autoregressive policies for efficient multi-agent combinatorial optimization. *arXiv preprint arXiv:2409.03811*, 2024a.

Berto, F., Hua, C., Zepeda, N. G., Hottung, A., Wouda, N., Lan, L., Tierney, K., and Park, J. RouteFinder: Towards foundation models for vehicle routing problems. *arXiv preprint arXiv:2406.15007*, 2024b.

Bi, J., Ma, Y., Wang, J., Cao, Z., Chen, J., Sun, Y., and Chee, Y. M. Learning generalizable models for vehicle routing problems via knowledge distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Cao, Y., Sun, Z., and Sartoretti, G. DAN: Decentralized attention-based neural network for the minmax multiple traveling salesman problem. *arXiv preprint arXiv:2109.04205*, 2021.

Dantzig, G. B. and Ramser, J. H. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959. doi: 10.1287/mnsc.6.1.80.

Drakulic, D., Michel, S., Mai, F., Sors, A., and Andreoli, J.-M. Bq-nco: Bisimulation quotienting for generalizable neural combinatorial optimization. *arXiv preprint arXiv:2301.03313*, 2023.

Goh, Y. L., Ma, Y., Zhou, J., Cao, Z., Dupty, M. H., and Lee, W. S. SHIELD: Multi-task multi-distribution vehicle routing solver with sparsity & hierarchy in efficiently layered decoder, 2025. URL https://openreview.net/forum?id=AMbIvaD4Rr.

Gui, S., Liu, S., Wang, X., and Cao, Z. Chain-of-context learning: Dynamic constraint understanding for multi-task VRPs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AhE6aSlz5g.

Jari, A., Afifi, S., Guibadj, R. N., and Lefèvre, E. G-unirouting: A graph-based unified neural model for solving multi-attribute vehicle routing problems. In *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 491–498. IEEE, 2025.

Jiang, X., Wu, Y., Wang, Y., and Zhang, Y. Unco: Towards unifying neural combinatorial optimization through large language model. 2024.

Kim, M., Park, J., and Park, J. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Min, S., and Gwon, Y. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Li, H., Liu, F., Zheng, Z., Zhang, Y., and Wang, Z. CaDA: Cross-problem routing solver with constraint-aware dual-attention. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=CS4RyQuTig.

Liu, F., Lin, X., Wang, Z., Zhang, Q., Xialiang, T., and Yuan, M. Multi-task learning for routing problem with cross-problem zero-shot generalization. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1898–1908, 2024.

Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

Pan, W., Xiong, H., Ma, J., Zhao, W., Li, Y., and Yan, J. UniCO: On unified combinatorial optimization via problem reduction to matrix-encoded general TSP. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yEwakMNIex.

Son, J., Kim, M., Choi, S., Kim, H., and Park, J. Equity-transformer: Solving NP-hard min-max routing problems as sequential generation with equity context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Toth, P. and Vigo, D. (eds.). *Vehicle Routing: Problems, Methods, and Applications*. SIAM, second edition, 2014. doi: 10.1137/1.9781611973594.

Vidal, T. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, 2022.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Wouda, N. A., Lan, L., and Kool, W. PyVRP: A high-performance VRP solver package. *INFORMS Journal on Computing*, 2024.

Zheng, Z., Yao, S., Wang, Z., Xialiang, T., Yuan, M., and Tang, K. DPN: Decoupling partition and navigation for neural solvers of min-max vehicle routing problems. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ar174skI9u.

Zhou, J., Wu, Y., Song, W., Cao, Z., and Zhang, J. MV-MoE: Multi-task vehicle routing solver with mixture-of-experts. In *International Conference on Machine Learning (ICML)*, 2024.

A. Formal Remarks on the Multi-Agent View

This appendix collects two formal remarks that support the framing in Section 3. The remarks are observations about the relationship between the single-agent and multi-agent decoding recipes rather than new theoretical results, and they are stated here primarily to make the reductions made informally in Sections 3.2 and 4.4 explicit.

A.1. Single-agent decoding as a constrained joint policy

Recall the single-agent and joint decoding distributions of Equations (1) and (5). We make precise the claim, used informally in Sections 3.2 and 4.2, that the single-agent recipe is the special case of the joint policy in which the active vehicle set is restricted to a singleton determined externally by the depot-return convention.

Proposition A.1 (Single-agent recipe as a constrained joint policy)**.** *Fix a step t with feasible action set $\mathcal{A}_t \subseteq \mathcal{K}_t \times \mathcal{T}_t$ and joint logits $\{\ell_{k,v}\}$ as in Equation (4). Let $k_t^\star \in \mathcal{K}_t$ denote the active vehicle prescribed by the depot-return convention (the most recently opened tour that has not yet returned to the depot). Then the masked joint policy*

$$\pi_\theta(a_t \mid s_t) \;=\; \frac{\exp(\ell_{k,v})\mathbb{1}[(k,v) \in \mathcal{A}_t]}{\sum_{(k',v')\in\mathcal{A}_t} \exp(\ell_{k',v'})}$$

restricted to the constrained action set $\mathcal{A}'_t = \{k_t^\star\} \times \mathcal{T}_t$ coincides, up to the parameter-free distance bias of Equation (4) and the relabelling $q_{k_t^\star} = W_q\, h^{\mathrm{a}}_{k_t^\star}/\sqrt{d}$, with the single-agent policy of Equation (1).

Proof. The mask $\mathbb{1}[(k,v) \in \mathcal{A}'_t]$ zeroes every (k,v) with $k \neq k_t^\star$, so the partition function reduces to $Z = \sum_{v\in\mathcal{T}_t} \exp(\ell_{k_t^\star,v})$. After the relabelling, the dot-product term of $\ell_{k_t^\star,v}$ is exactly $\langle q_{k_t^\star}, h^{\mathrm{n}}_v\rangle$, which is the logit of Equation (1). The remaining additive distance bias depends only on positional inputs, carries no learnable parameters, and is the only mismatch between the two recipes; stripping it from Equation (4) (or, equivalently, adding the same bias to Equation (1)) makes the reduction exact. □

Two consequences are worth noting. First, the reduction is one-way: the single-agent recipe is recovered from the joint one by fixing $k_t^\star$ externally, but the joint policy cannot in general be recovered from a single-agent policy because the depot-return convention does not determine a distribution over $\mathcal{K}_t$. Second, when $|\mathcal{K}_t| = 1$ throughout a rollout the two recipes are identically distributed, which matches the empirical observation that the multi-agent advantage degenerates at $|\mathcal{K}| = 1$ (Section 6). The conditions under which $|\mathcal{K}_t| > 1$ during a rollout are exactly the conditions under which we expect the multi-agent view to add information, and these are precisely the variants and instance regimes on which our experiments report the largest gains (Section 5.2).

A.2. Allocation-driven multi-start versus POMO multi-start

The multi-start ablation in Section 5.3 compares allocation-driven multi-start with two alternatives: no multi-start at all, augmentation alone, and POMO's forced first-customer multi-start. The following remark formalises the basic structural difference between the last two and is what motivates the empirical comparison.

Remark A.2 (Initial-commitment expressivity). POMO's multi-start with budget M assigns each rollout a unique first customer v_0^m, so the map $m \mapsto v_0^m$ is by construction injective and any two POMO rollouts on the same instance differ in their first customer. Allocation-driven multi-start of Section 4.4 samples a joint pair (k_0^m, v_0^m), so two rollouts may share v_0^m while differing in k_0^m, and the support of initial commitments is the joint product set $\mathcal{K}_t \times \mathcal{T}_t$ rather than only $\mathcal{T}_t$.

The remark is structural rather than empirical, and it does not by itself imply better quality. What it does say is that POMO's diversity is concentrated on one axis of the initial commitment, while allocation-driven multi-start places mass on both axes; whether the second axis is informative for downstream tour quality is the question that the per-task numbers in Table 9 are designed to answer. The observed gap between forced multi-start and allocation-driven multi-start (1.610% vs. 1.386% in mean gap) is therefore consistent with, and not larger than, what Remark A.2 can predict.

B. Variant Taxonomy and Evaluation Protocol

B.1. Attribute taxonomy

We organize all evaluated variants over a six-attribute alphabet on top of the base CVRP, in which each attribute is a binary flag controlling one structural extension of the routing problem. The four attributes used during training are open route (O), time windows (TW), distance limit (L), and backhaul (B), and the two additional attributes encountered only at test time are multi-depot (MD) and mixed backhaul (MB). A variant is identified by the subset of attributes that is active; for instance, OVRPBLTW combines open route, backhaul, distance limit, and time windows on top of base CVRP, while MDOVRPMBLTW additionally activates multi-depot and replaces the strict-backhaul flag with mixed backhaul.

In-distribution variants. Training covers all $16 = 2^4$ subsets of {O, TW, L, B}, exactly the 16 variants reported in Table 1: CVRP, OVRP, VRPB, VRPL, VRPTW, OVRPB, OVRPL, OVRPTW, VRPBL, VRPBTW, VRPLTW, OVRPBL, OVRPBTW, OVRPLTW, VRPBLTW, OVRPBLTW. Each in-distribution training instance samples customer coordinates uniformly in the unit square and demands as in the RouteFinder protocol (Berto et al., 2024b), with $N \in \{50, 100\}$ and active fleets up to $|\mathcal{K}| = 8$.

Out-of-distribution variants. The cross-task evaluation in Section 5.2 draws from 32 variants in which at least one of MD or MB is active. By construction, 16 of these 32 activate MD without MB, and the remaining 16 activate MB (with or without MD). None of the 32 variants is seen during training, so the evaluation requires the model to extrapolate the constraint composition encoded by its constraint prompt to attribute combinations it has never decoded.

CVRPLib-X. For cross-distribution evaluation we use the 65 instances of the X benchmark released as part of CVRPLib. The X instances are real-world style CVRP instances with sizes $N \in [100, 1000]$ and a deliberate diversity of customer placements (random, clustered, mixed), depot positions (central, eccentric, random), and demand profiles, designed to expose generalization failures of solvers tuned to a single stylized prior. Optima for each instance are taken from the published best-known solutions distributed with the benchmark.

B.2. Test instances and metrics

Test sets contain 1,000 instances per (variant, size) cell for in-distribution and cross-task evaluation, and the released 65 instances for CVRPLib-X. For every cell we report the average gap to the strong reference solver in percent, computed as $100 \cdot (\text{cost}_{\text{model}} - \text{cost}_{\text{ref}})/\text{cost}_{\text{ref}}$, where the reference is HGS-PyVRP for the synthetic in-distribution and cross-task evaluations and the published optimum for CVRPLib-X. We additionally report wall-clock inference time per (variant, size) cell on a single GPU. All neural baselines decode each test instance under the same per-instance rollout budget for fairness; the ablation tables in Tables 9 and 10 use the joint augmentation-and-sample budget 8×50 (eight symmetry augmentations crossed with fifty samples) at $N = 50$, and the budgets used in Table 1 are matched between methods on a per-cell basis. Feasibility is enforced through the action mask of Equation (5), so all reported gaps are computed on solutions that satisfy the active side constraints by construction.

C. Implementation Details

We document here the architectural and procedural choices left implicit in Section 4. The aim is reproducibility rather than completeness; the exact training-time numerics (optimizer constants, learning-rate schedule, batch sizes, and total compute) follow the RouteFinder protocol (Berto et al., 2024b) and are summarized at the end of this section.

C.1. Encoder and constraint prompt

The encoder follows the architecture of RouteFinder (Berto et al., 2024b) with the constraint-aware prompt of CaDA (Li et al., 2025) prepended to the input sequence; we change neither component. Each input token (a customer, an agent node, or the constraint prompt) is linearly projected to a shared embedding dimension $D = 128$, and a stack of 6 multi-head self-attention blocks, each with 8 heads and a feed-forward sublayer of hidden dimension 512, produces the row-by-row output $\mathbf{H} \in \mathbb{R}^{(K+N)\times D}$ used by the decoder. The constraint flag $\tilde{\mathbf{h}} \in \{0, 1\}^4$ has one slot per training attribute {O, TW, L, B} exactly as in CaDA. On out-of-distribution variants that activate MD or MB, the model is evaluated zero-shot without retraining and without extending the constraint prompt; the absolute gap reductions reported in Appendix D.1 for these

variants are therefore a property of the trained model rather than of the prompt vocabulary.

C.2. Distance-bias term

The joint logits in Equation (4) include a parameter-free term $\log(1/(\mathrm{dist}(\mathrm{pos}_k, v) + \varepsilon))$ whose role is purely to keep gradients informative early in training. The Euclidean distance $\mathrm{dist}(\cdot, \cdot)$ uses the unit-square coordinates of the depot at evaluation time and the agent's current position during rollout, ε is a small numerical constant that prevents division by zero at coincident positions, and the term carries no learnable parameters and contributes no gradient to the encoder or to W_q (the gradient is detached at the input to the bias). The bias is applied to all $(k, v) \in \mathcal{A}_t$, including agent-node targets, where $\mathrm{dist}(\mathrm{pos}_k, a_k) = 0$ and the bias defaults to its ε-clipped value. Because the bias is shared across baselines (it is also present in the action-ablation baseline of Table 10), the cross-method comparisons in Section 5.3 are not confounded by its presence.

C.3. Inter-agent communication block

The communication block reuses PARCO's `CommunicationLayers` module (Berto et al., 2024a) unchanged. Inputs are the per-vehicle context vectors $\bar{h}^{\mathrm{a}}{}_k$ defined in Section 4.3, and outputs are the communicated queries $\tilde{h}^{\mathrm{a}}{}_k$ that replace h^{a}_k in the joint logits of Equation (4). The block is a single transformer layer (8-head self-attention plus a feed-forward sublayer of hidden dimension 512, matching the encoder) and is rerun at every decoding step on the active fleet only, so its sequence length is $|\mathcal{K}_t|$ rather than $K + N$ and its per-step cost is small relative to the encoder pass. Our contribution lies in how the output is consumed (as queries entering a joint softmax) rather than in the block itself; Table 10 verifies that the same communication block paired with PARCO's parallel decoding yields a 9.480% mean gap, against 1.386% when the same block feeds the joint softmax of Equation (5).

C.4. Allocation-driven multi-start

Per-rollout capacity scaling. Each rollout m samples a scaling factor $c_m \sim \mathcal{U}[c_{\mathrm{low}}, c_{\mathrm{high}}]$ from a fixed interval that does not depend on the variant; we use $c_{\mathrm{low}} = 1.2$ and $c_{\mathrm{high}} = 2.0$ throughout. The scaling factor enters only through the criterion that determines how many vehicles are active in rollout m: an additional vehicle is activated whenever the cumulative customer demand exceeds $c_m \cdot Q$, with Q the nominal per-vehicle capacity. The original per-vehicle capacity Q is preserved in the action-mask feasibility check and in the cost computation, so the reported tour length of every rollout is evaluated under the unmodified instance. The scaling factor therefore controls the active-fleet count $|\mathcal{K}_t|$ across rollouts without changing the underlying problem, which is the structural mechanism that the single-agent recipe lacks (Section 4.4).

Initial action sampling. At step $t = 1$, each rollout draws its initial joint action (k_0^m, v_0^m) by multinomial sampling from Equation (5). This is the procedural difference from POMO multi-start formalized in Remark A.2; we use multinomial sampling rather than top-M selection to retain stochastic exploration over the joint distribution. Setting the budget to $M = N$ rollouts per instance matches POMO's budget and keeps the per-instance compute aligned with the POMO-derived baselines that we compare against.

C.5. Training and inference protocol

Training follows the RouteFinder mixed-batch protocol (Berto et al., 2024b): each mini-batch samples a variant uniformly from the 16 in-distribution variants and an instance from the variant's distribution, and the policy is trained with REINFORCE using the multi-start mean reward as a per-instance baseline. Reward magnitudes vary substantially across variants (a CVRPTW solution length is on a different scale from a CVRP solution length), so we apply an exponential-moving-average normalization of per-variant reward statistics before combining them into the policy gradient, again as in RouteFinder. At inference time we apply the same eight-symmetry augmentation as in POMO (Kwon et al., 2020) and POMO-derived baselines, take the best tour over the augmented samples, and report the resulting cost.

Optimization and schedule. We optimize with Adam at a learning rate of 3×10^{-4} and a weight decay of 1×10^{-6}. A multi-step learning-rate scheduler decays the learning rate by a factor of 0.1 at epochs 270 and 295, and gradients are clipped to norm 1. Training runs for a total of 300 epochs. We use a training batch size of 256 and a test batch size of 128.

Compute. Training is run on 8× NVIDIA A100 80GB GPUs, and a single training run takes approximately 1 day at $N = 50$ and 2.5 days at $N = 100$. Wall-clock inference times reported in Table 1 are measured on a single NVIDIA RTX 4090 GPU.

Baselines. For all neural baselines (MTPOMO, MVMoE, RouteFinder, CaDA, CCL), we use the publicly released checkpoints from the original works rather than retraining. Because our training setup (dataset, variant distribution, and instance generator) matches the protocol used by these baselines, the released checkpoints are directly comparable under our evaluation.

D. Per-Task Experimental Results

This section expands the aggregate numbers reported in Section 5 into per-task tables, and accompanies each table with a short description of the patterns relevant to the multi-agent claim of Section 3.

D.1. Per-task cross-task generalization results

Tables 6 and 7 expand the summary in Table 2 into per-task numbers for all 32 unseen out-of-distribution variants, at $N = 50$ and $N = 100$ respectively. Two observations support the cross-task narrative of Section 5.2. First, the multi-agent solver is the best neural method on 28 of 32 variants at $N = 50$ and on all 32 variants at $N = 100$, and on the four variants on which CaDA wins at $N = 50$ (VRPMB, OVRPMB, VRPMBL, and OVRPMBL) the gap to ours is at most 1.5 percentage points, so the cross-task lead is consistent rather than concentrated. Second, beyond the by-family summary in Table 3, the lead is concentrated on individual multi-depot variants combined with time windows or distance limits (e.g., MDVRPTW, MDOVRPTW, MDVRPLTW, MDVRPBTW, MDVRPBLTW), where our solver reduces the strongest baseline's 30–45% gap at $N = 100$ to roughly 14–19%, a more than two-fold reduction in absolute gap. Variants in which only mixed backhaul is added on top of base CVRP (VRPMB, VRPMBL, VRPMBTW, VRPMBLTW and their open-route counterparts) admit smaller absolute gains, consistent with the observation that backhaul-only modifications do not strongly couple agents through a shared resource.

D.2. Per-instance CVRPLib-X results

Table 8 reports per-instance gaps to the published optima for all 65 instances of the CVRPLib-X benchmark referenced in the cross-distribution paragraph of Section 5.2. The aggregate average gap of our solver is 8.286% against 14.356% for the strongest in-distribution baseline (CCL), and our solver attains the lowest gap on 37 of 65 instances against 9, 8, and 11 for the three baselines respectively. The per-instance pattern shows two regimes. On the smaller instances ($N \leq 250$), all four neural methods are within a few percentage points of one another and the per-instance winner is essentially noise: ours wins on 14 of 33 small instances, with each baseline winning a comparable share, and absolute gaps stay below 10% for every method on most cells. On the larger instances ($N \geq 500$), the picture diverges: ours is the best neural method on 23 of the 32 large instances, and CCL's gap repeatedly exceeds 30 to 50 percentage points on the largest cells (X-n819-k171, X-n837-k142, X-n856-k95, X-n916-k207, X-n957-k87, X-n979-k58, X-n1001-k43), while ours stays below 20% on 30 of the 32 large cells and below 30% on every cell. The widening gap on $N \geq 500$ is consistent with the prediction in Section 5.2 that serialized rollouts compound errors over thousands of steps; the multi-agent decomposition spreads the same horizon across $|\mathcal{K}|$ shorter tours and is correspondingly less sensitive to instance size.

Table 8. Per-instance results on the CVRPLib-X benchmark (65 instances). Gap := (cost − optimal)/optimal. Best (lowest Gap) per row is highlighted in **bold**.

Instance	N	Optimal	RF-TE		CaDA		CCL		Ours	
			Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓
X-n101-k25	100	27591	29087	5.422%	29185	5.777%	**28900**	**4.744%**	29066	5.350%
X-n106-k14	105	26362	27159	3.023%	26952	2.238%	27173	3.076%	**26873**	**1.938%**
X-n110-k13	109	14971	15314	2.291%	15262	1.944%	15334	2.425%	**15212**	**1.610%**
X-n115-k10	114	12747	13281	4.189%	**13169**	**3.311%**	13282	4.197%	13400	5.131%
X-n120-k6	119	13332	13765	3.248%	13735	3.023%	13821	3.668%	**13673**	**2.565%**
X-n125-k30	124	55539	58570	5.457%	**57405**	**3.360%**	57732	3.949%	58697	5.686%
X-n129-k18	128	28940	29598	2.274%	**29397**	**1.579%**	29548	2.101%	29583	2.225%
X-n134-k13	133	10916	11624	6.486%	11512	5.460%	11476	5.130%	**11376**	**4.214%**
X-n139-k10	138	13590	**13756**	**1.221%**	13877	2.112%	13938	2.561%	13854	1.943%

continued on next page

Table 8 – continued from previous page

Instance	N	Optimal	RF-TE		CaDA		CCL		Ours	
			Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓
X-n143-k7	142	15700	16257	3.548%	16195	3.153%	16213	3.268%	**16037**	**2.153%**
X-n148-k46	147	43448	**45036**	**3.655%**	45761	5.324%	45522	4.774%	46012	5.901%
X-n153-k22	152	21220	23478	10.641%	23154	9.114%	**23105**	**8.883%**	23300	9.802%
X-n157-k13	156	16876	17339	2.744%	17344	2.773%	**17123**	**1.464%**	17204	1.944%
X-n162-k11	161	14138	14664	3.720%	14814	4.781%	**14638**	**3.537%**	14641	3.565%
X-n167-k10	166	20557	21432	4.256%	21437	4.281%	**20940**	**1.863%**	21107	2.680%
X-n172-k51	171	45607	48118	5.506%	48181	5.644%	48142	5.558%	**47939**	**5.113%**
X-n176-k26	175	47812	**51400**	**7.504%**	52698	10.219%	52533	9.874%	52974	10.796%
X-n181-k23	180	25569	26106	2.100%	26099	2.073%	26218	2.538%	**26060**	**1.924%**
X-n186-k15	185	24145	25148	4.154%	25461	5.450%	25096	3.939%	**24971**	**3.425%**
X-n190-k8	189	16980	17868	5.230%	18470	8.775%	17572	3.486%	**17504**	**3.092%**
X-n195-k51	194	44225	47390	7.157%	46726	5.655%	**46568**	**5.298%**	47869	8.240%
X-n200-k36	199	58578	61199	4.474%	61198	4.473%	61433	4.874%	**60959**	**4.065%**
X-n204-k19	203	19565	20594	5.259%	20497	4.764%	20557	5.070%	**20387**	**4.206%**
X-n209-k16	208	30656	31876	3.980%	32092	4.684%	32453	5.862%	**31487**	**2.714%**
X-n214-k11	213	10856	11670	7.498%	11812	8.806%	11560	6.485%	**11489**	**5.831%**
X-n219-k73	218	117595	120349	2.342%	120464	2.440%	124930	6.238%	**118943**	**1.146%**
X-n223-k34	222	40437	42251	4.486%	42359	4.753%	**42103**	**4.120%**	42191	4.338%
X-n228-k23	227	25742	28798	11.872%	27988	8.725%	**27731**	**7.727%**	29688	15.333%
X-n233-k16	232	19230	20713	7.712%	20638	7.322%	**20296**	**5.543%**	20435	6.266%
X-n237-k14	236	27042	29595	9.441%	30451	12.606%	29943	10.728%	**29227**	**8.080%**
X-n242-k48	241	82751	**85704**	**3.569%**	85780	3.660%	86015	3.944%	86884	4.995%
X-n247-k50	246	37274	**40642**	**9.036%**	41037	10.096%	40954	9.873%	41748	12.006%
X-n251-k28	250	38684	**40134**	**3.748%**	40663	5.116%	40484	4.653%	40185	3.883%
X-n502-k39	501	69226	71742	3.634%	72427	4.624%	72438	4.640%	**71393**	**3.130%**
X-n513-k21	512	24201	28542	17.937%	30037	24.115%	27700	14.458%	**26968**	**11.438%**
X-n524-k153	523	154593	173568	12.274%	**171656**	**11.037%**	178458	15.437%	174991	13.195%
X-n536-k96	535	94846	103499	9.123%	102768	8.352%	106288	12.064%	**102143**	**7.694%**
X-n548-k50	547	86700	**100681**	**16.126%**	102813	18.585%	106464	22.796%	100838	16.308%
X-n561-k42	560	42717	49406	15.659%	50410	18.009%	48094	12.587%	**47607**	**11.447%**
X-n573-k30	572	50673	55876	10.268%	56622	11.740%	**55593**	**9.709%**	56126	10.761%
X-n586-k159	585	190316	205224	7.833%	205385	7.918%	232361	22.092%	**203415**	**6.883%**
X-n599-k92	598	108451	**116732**	**7.636%**	117727	8.553%	129174	19.108%	117147	8.019%
X-n613-k62	612	59535	67530	13.429%	68696	15.388%	**67489**	**13.360%**	67551	13.464%
X-n627-k43	626	62164	67415	8.447%	68838	10.736%	71602	15.182%	**66858**	**7.551%**
X-n641-k35	640	63684	70670	10.970%	73329	15.145%	73053	14.712%	**69361**	**8.916%**
X-n655-k131	654	106780	112134	5.014%	110761	3.728%	122640	14.853%	**109485**	**2.534%**
X-n670-k130	669	146332	168867	15.400%	**165711**	**13.243%**	183348	25.296%	175307	19.801%
X-n685-k75	684	68205	77727	13.961%	78145	14.574%	77958	14.300%	**77429**	**13.525%**
X-n701-k44	700	81923	91008	11.090%	92254	12.611%	94292	15.098%	**89776**	**9.586%**
X-n716-k35	715	43373	49503	14.133%	51313	18.306%	49893	15.032%	**48365**	**11.509%**
X-n733-k159	732	136187	148369	8.945%	**148357**	**8.936%**	163255	19.876%	153700	12.860%
X-n749-k98	748	77269	**84985**	**9.986%**	85634	10.826%	88803	14.927%	85874	11.138%
X-n766-k71	765	114417	130109	13.715%	**128140**	**11.994%**	138265	20.843%	140420	22.727%
X-n783-k48	782	72386	82894	14.517%	84805	17.157%	87558	20.960%	**82117**	**13.443%**
X-n801-k40	800	73311	85941	17.228%	89329	21.849%	92808	26.595%	**85218**	**16.242%**
X-n819-k171	818	158121	174037	10.066%	173461	9.701%	231883	46.649%	**169868**	**7.429%**
X-n837-k142	836	193737	208396	7.566%	208225	7.478%	274880	41.883%	**208063**	**7.395%**
X-n856-k95	855	88965	98455	10.667%	100057	12.468%	123333	38.631%	**96828**	**8.838%**
X-n876-k59	875	99299	107300	8.057%	110332	11.111%	115958	16.777%	**107205**	**7.962%**
X-n895-k37	894	53860	64631	19.998%	67511	25.345%	65179	21.016%	**61772**	**14.690%**
X-n916-k207	915	329179	353584	7.414%	353009	7.239%	509703	54.841%	**350865**	**6.588%**
X-n936-k151	935	132715	162220	22.232%	**154841**	**16.672%**	195452	47.272%	172308	29.833%
X-n957-k87	956	85465	102788	20.269%	105200	23.091%	135339	58.356%	**101754**	**19.060%**
X-n979-k58	978	118976	129946	9.220%	133144	11.908%	148811	25.076%	**127163**	**6.882%**
X-n1001-k43	1000	72355	85828	18.621%	89231	23.324%	113779	57.251%	**83629**	**15.582%**
Avg. Gap			8.595%		9.373%		14.356%		**8.286%**	
# Best (Best/Total)			9/65		8/65		11/65		**37/65**	

D.3. Per-task ablation results

Tables 9 and 10 expand the summary in Table 5 into per-task numbers for the multi-start and action-formulation ablations at $N = 50$.

Multi-start ablation. Table 9 compares no-multi-start, augmentation-only, POMO's forced first-customer multi-start, and our allocation-driven multi-start, with the test-time budget held at 8×50 across all four columns. Three patterns emerge. First, the no-multi-start column has mean gap 17.632%, which sets a lower bound for the value of any multi-start scheme; the gap on backhaul-with-distance-limit variants (VRPB, VRPBL) reaches 28 to 32% in this column and collapses to 2 to 4% as soon as any multi-start is reintroduced, showing that the multi-start budget is the dominant lever for these variants regardless of the scheme. Second, the marginal gain from forced multi-start to allocation-driven multi-start is

Table 6. Per-task results on 32 unseen out-of-distribution tasks (N=50).

Tasks	RF-TE		CaDA		CCL		Ours	
	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓
VRPMB	9.879	8.861%	**9.781**	**7.749%**	9.943	9.538%	9.832	8.320%
MDCVRP	12.559	56.957%	12.335	54.083%	10.829	34.846%	**9.028**	**12.425%**
MDOVRP	6.876	29.051%	6.825	28.069%	6.413	20.155%	**5.982**	**12.076%**
MDVRPB	12.725	60.956%	12.654	60.054%	11.679	47.450%	**8.940**	**12.868%**
MDVRPL	12.618	57.370%	12.426	54.988%	11.278	40.248%	**9.079**	**12.908%**
OVRPMB	6.949	13.690%	**6.872**	**12.440%**	6.917	13.149%	6.887	12.651%
VRPMBL	10.239	8.050%	**10.163**	**7.215%**	10.376	9.456%	10.216	7.765%
MDOVRPB	7.556	31.861%	7.622	33.009%	6.876	19.787%	**6.393**	**11.375%**
MDOVRPL	6.871	28.946%	6.807	27.746%	6.442	20.716%	**5.984**	**12.126%**
MDVRPBL	12.831	61.175%	12.585	58.011%	11.483	43.846%	**9.055**	**13.434%**
MDVRPMB	12.856	76.544%	12.493	71.550%	11.540	58.118%	**8.913**	**22.124%**
MDVRPTW	17.818	48.941%	16.971	41.739%	16.107	34.403%	**13.733**	**14.595%**
OVRPMBL	6.949	13.686%	**6.871**	**12.423%**	6.932	13.404%	6.882	12.582%
VRPMBTW	17.298	8.074%	17.198	7.434%	16.988	6.099%	**16.838**	**5.161%**
MDOVRPBL	7.550	31.772%	7.606	32.713%	6.858	19.463%	**6.405**	**11.571%**
MDOVRPMB	7.617	47.411%	7.541	45.953%	6.720	29.826%	**6.362**	**22.917%**
MDOVRPTW	10.618	34.976%	10.204	29.610%	9.783	24.150%	**8.940**	**13.457%**
MDVRPBTW	18.591	37.364%	19.025	40.629%	18.425	36.029%	**15.666**	**15.662%**
MDVRPLTW	18.127	51.276%	17.125	42.765%	16.769	39.728%	**13.778**	**14.801%**
MDVRPMBL	12.744	74.112%	12.434	69.825%	11.474	56.409%	**8.949**	**21.992%**
OVRPMBTW	11.132	6.265%	11.087	5.849%	10.966	4.658%	**10.964**	**4.639%**
VRPMBLTW	17.597	7.982%	17.495	7.337%	17.514	7.433%	**17.122**	**5.027%**
MDOVRPBTW	11.399	32.332%	11.190	29.807%	10.774	24.830%	**9.878**	**14.451%**
MDOVRPLTW	10.599	34.731%	10.196	29.501%	9.721	23.331%	**8.947**	**13.506%**
MDOVRPMBL	7.602	47.108%	7.540	45.940%	6.733	30.088%	**6.362**	**22.927%**
MDVRPBLTW	19.048	40.583%	19.243	42.015%	18.544	36.702%	**15.708**	**15.796%**
MDVRPMBTW	17.830	48.485%	18.394	53.345%	17.103	42.314%	**14.562**	**21.170%**
OVRPMBLTW	11.138	6.317%	11.090	5.875%	10.993	4.922%	**10.955**	**4.555%**
MDOVRPBLTW	11.389	32.207%	11.185	29.743%	10.767	24.749%	**9.880**	**14.470%**
MDOVRPMBTW	11.055	40.153%	10.833	37.232%	10.171	28.688%	**9.412**	**19.091%**
MDVRPMBLTW	18.222	51.554%	18.511	54.026%	17.940	49.174%	**14.588**	**21.305%**
MDOVRPMBLTW	11.054	40.136%	10.818	37.044%	10.203	29.114%	**9.413**	**19.118%**
Avg. Gap	36.529%		34.866%		27.588%		**13.965%**	
# Best (Best/Total)	0/32		4/32		0/32		**28/32**	

small but consistent across all 16 variants (mean 0.2 percentage points, with our scheme winning every single per-task cell), which is the empirical content of Remark A.2. Third, the gain tends to be larger on time-windowed variants and on variants combining backhaul with a distance limit: VRPTW, VRPLTW, VRPBLTW, and VRPBL each see improvements over forced multi-start of 0.25 to 0.4 percentage points, broadly the variants in which the time-window or capacity-tightness axis couples agents most tightly and in which aligning the cluster with the right vehicle in advance pays off.

Action-formulation ablation. Table 10 reports the same model trained with our joint $(\text{vehicle}, \text{target})$ softmax (Equation (5)) versus the alternative in which the same encoder and communication block feed PARCO's parallel-action decoder. The mean gap rises from 1.386% to 9.480% when the joint softmax is replaced, and the deterioration is uniform across the 16 variants: every per-task cell is dominated by the joint softmax, with the largest absolute parallel-action gaps observed on the open-route and pure-backhaul subset (OVRP, OVRPL, VRPBL, VRPB), each in excess of 13%. This is consistent with the framing remark in Section 4.3 that communication is most useful when the cross-agent context shapes the choice rather than being followed by a post-hoc resolution stage; on time-windowed variants (e.g., OVRPBLTW, OVRPBTW) where the action mask already prunes most cross-agent conflicts, the joint softmax retains a smaller but still substantial advantage.

E. Limitations and Broader Impacts

We expand here on the limitations note in Section 6.

Table 7. Per-task results on 32 unseen out-of-distribution tasks (N=100).

Tasks	RF-TE		CaDA		CCL		Ours	
	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓
VRPMB	14.888	10.189%	14.710	8.822%	15.322	13.438%	**14.764**	**9.306%**
MDCVRP	19.684	67.107%	20.628	75.392%	16.769	41.675%	**13.511**	**14.156%**
MDOVRP	10.683	34.368%	10.605	33.387%	10.095	26.715%	**9.233**	**15.880%**
MDVRPB	18.721	61.761%	19.494	68.604%	18.185	56.977%	**13.620**	**17.554%**
MDVRPL	20.100	70.498%	20.867	77.336%	17.494	47.699%	**13.544**	**14.340%**
OVRPMB	10.711	18.899%	10.490	16.449%	10.780	19.642%	**10.517**	**16.725%**
VRPMBL	15.198	10.375%	15.016	9.011%	15.761	14.464%	**15.125**	**9.849%**
MDOVRPB	11.752	35.659%	11.790	36.137%	10.959	26.274%	**10.082**	**16.359%**
MDOVRPL	10.703	34.620%	10.574	32.985%	10.272	29.033%	**9.206**	**15.730%**
MDVRPBL	19.606	68.898%	19.827	70.875%	18.446	58.520%	**13.759**	**18.224%**
MDVRPMB	18.698	76.325%	19.458	83.817%	18.997	79.149%	**13.644**	**28.657%**
MDVRPTW	29.468	53.283%	25.955	34.783%	26.901	39.557%	**22.134**	**14.798%**
OVRPMBL	10.709	18.877%	10.486	16.399%	10.883	20.748%	**10.511**	**16.625%**
VRPMBTW	28.256	10.840%	28.317	11.074%	28.100	10.239%	**27.122**	**6.401%**
MDOVRPBL	11.761	35.771%	11.788	36.106%	10.983	26.545%	**10.065**	**16.004%**
MDOVRPMB	11.748	53.650%	11.692	53.010%	10.876	42.044%	**10.069**	**31.500%**
MDOVRPTW	18.299	41.356%	17.205	32.804%	17.009	31.200%	**14.833**	**14.590%**
MDVRPBTW	30.681	39.926%	30.176	37.559%	30.677	39.759%	**25.947**	**18.293%**
MDVRPLTW	29.640	53.976%	26.200	35.862%	27.977	45.055%	**22.154**	**14.924%**
MDVRPMBL	19.216	80.892%	19.460	83.444%	18.479	73.647%	**13.696**	**28.711%**
OVRPMBTW	18.449	8.724%	18.478	8.901%	18.427	8.590%	**17.863**	**5.266%**
VRPMBLTW	28.604	10.805%	28.658	11.002%	28.374	9.907%	**27.451**	**6.331%**
MDOVRPBTW	19.590	36.897%	19.305	34.918%	18.764	30.960%	**16.828**	**17.441%**
MDOVRPLTW	18.232	40.817%	17.221	32.923%	16.838	29.857%	**14.816**	**14.241%**
MDOVRPMBL	11.751	53.691%	11.670	52.726%	10.830	41.416%	**10.050**	**31.229%**
MDVRPBLTW	31.044	41.408%	30.537	39.033%	30.667	39.535%	**25.970**	**18.139%**
MDVRPMBTW	29.650	54.395%	29.383	53.001%	28.388	47.661%	**24.112**	**25.434%**
OVRPMBLTW	18.452	8.739%	18.476	8.887%	18.404	8.443%	**17.861**	**5.244%**
MDOVRPBLTW	19.553	36.632%	19.341	35.154%	18.853	31.599%	**16.828**	**17.440%**
MDOVRPMBTW	18.888	46.216%	18.735	45.019%	17.823	37.814%	**15.869**	**22.690%**
MDVRPMBLTW	29.825	55.132%	29.611	53.989%	29.000	50.703%	**24.102**	**25.256%**
MDOVRPMBLTW	18.846	45.873%	18.770	45.286%	17.803	37.665%	**15.870**	**22.710%**
Avg. Gap	41.144%		39.834%		34.892%		**17.134%**	
# Best (Best/Total)	0/32		0/32		0/32		**32/32**	

Table 9. Per-task multi-start ablation at $N = 50$. All variants share the same test-time budget of 8×50 (8 augmentations, 50 samples). Mean values are in Table 5.

Task	no-multistart		Aug		Forced multistart		Ours	
	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓
CVRP	11.9358	15.077%	10.5093	1.324%	10.4834	1.074%	**10.4802**	**1.043%**
OVRP	7.5049	15.336%	6.6877	2.777%	6.6707	2.516%	**6.6579**	**2.319%**
OVRPB	7.9865	15.780%	7.0512	2.221%	7.0342	1.975%	**7.0219**	**1.796%**
OVRPBL	7.9935	15.865%	7.0510	2.203%	7.0361	1.987%	**7.0227**	**1.793%**
OVRPBLTW	13.0767	12.073%	11.7454	0.663%	11.7366	0.588%	**11.7150**	**0.403%**
OVRPBTW	13.0443	11.786%	11.7455	0.656%	11.7362	0.576%	**11.7141**	**0.386%**
OVRPL	7.5333	15.772%	6.6890	2.797%	6.6710	2.520%	**6.6571**	**2.307%**
OVRPLTW	11.8505	12.755%	10.5800	0.666%	10.5785	0.652%	**10.5522**	**0.402%**
OVRPTW	11.8506	12.756%	10.5803	0.669%	10.5771	0.638%	**10.5516**	**0.396%**
VRPB	12.4025	28.032%	10.0097	3.331%	9.9461	2.675%	**9.9274**	**2.482%**
VRPBL	13.4492	32.036%	10.6141	4.203%	10.5335	3.412%	**10.5006**	**3.089%**
VRPBLTW	21.5205	17.208%	18.8215	2.508%	18.7930	2.353%	**18.7464**	**2.099%**
VRPBTW	21.4529	17.280%	18.4816	1.037%	18.4607	0.922%	**18.4118**	**0.655%**
VRPL	12.5712	18.742%	10.7580	1.615%	10.7239	1.293%	**10.7184**	**1.241%**
VRPLTW	19.6271	19.999%	16.6037	1.514%	16.5898	1.429%	**16.5258**	**1.038%**
VRPTW	19.4955	21.611%	16.2077	1.102%	16.2148	1.147%	**16.1470**	**0.724%**
Mean	13.3309	17.632%	11.5085	1.830%	11.4866	1.610%	**11.4594**	**1.386%**

Scaling assumptions. The joint softmax of Equation (5) flattens an $|\mathcal{K}_t| \times |\mathcal{T}_t|$ action grid at every step, and the communication block of Section 4.3 runs an attention over the active fleet at every step. Per-step cost therefore grows as $O(|\mathcal{K}_t|\,|\mathcal{T}_t|)$ for the softmax and as $O(|\mathcal{K}_t|^2)$ for the communication block, and the overall rollout cost scales linearly in $|\mathcal{K}_t|$ on top of the customer-side cost. For the fleet sizes considered in our experiments ($|\mathcal{K}| \leq 8$ during training, and up to a few hundred on the largest CVRPLib-X instances) this overhead is dominated by the encoder pass; for substantially larger fleets, the communication block would be the first component to revisit.

Reduction at $|\mathcal{K}| = 1$. Proposition A.1 shows that the joint policy reduces to the single-agent recipe when $|\mathcal{K}_t| = 1$, so the multi-agent advantage is bounded above by the gap between single-active-vehicle and multi-active-vehicle decoding. On VRP variants in which fleet size is fixed to one (e.g., the standard TSP), the proposed framework offers no improvement over POMO-style baselines by construction, and we do not claim such an improvement.

Untested regimes. We have not evaluated on $N > 1{,}000$, on attribute classes structurally different from those in our six-attribute alphabet (in particular pickup-and-delivery, split delivery, and stochastic demands), or on non-Euclidean distance metrics. The multi-agent view is expected to transfer to these settings to the extent that the underlying problem still decomposes into per-agent partial routes that compete for shared resources, but this is a hypothesis for future work rather than a claim supported by the present experiments.

Comparison budget. The neural baselines in Tables 1, 2 and 8 are reported under matched per-instance rollout budgets within each cell, while HGS-PyVRP is run under its standard time budget as a non-neural reference. The classical reference is therefore not directly comparable in compute, and the gap-to-HGS metric should be read as "how close does each neural method come to a strong classical solver under a fixed neural-side budget" rather than as a wall-clock comparison.

Table 10. Per-task action-formulation ablation at $N = 50$ (joint vs. PARCO's parallel decoding). Mean values are in Table 5.

Task	Ours		Parallel Action (PARCO)	
	Obj. ↓	Gap ↓	Obj. ↓	Gap ↓
CVRP	**10.4802**	**1.043%**	11.3155	9.097%
OVRP	**6.6579**	**2.319%**	7.4592	14.634%
OVRPB	**7.0219**	**1.796%**	7.6618	11.073%
OVRPBL	**7.0227**	**1.793%**	7.6721	11.206%
OVRPBLTW	**11.7150**	**0.403%**	12.0356	3.150%
OVRPBTW	**11.7141**	**0.386%**	12.0317	3.108%
OVRPL	**6.6571**	**2.307%**	7.4597	14.642%
OVRPLTW	**10.5522**	**0.402%**	11.1458	6.050%
OVRPTW	**10.5516**	**0.396%**	11.1446	6.039%
VRPB	**9.9274**	**2.482%**	11.0001	13.555%
VRPBL	**10.5006**	**3.089%**	11.7245	15.105%
VRPBLTW	**18.7464**	**2.099%**	19.6448	6.992%
VRPBTW	**18.4118**	**0.655%**	19.3085	5.558%
VRPL	**10.7184**	**1.241%**	11.6116	9.677%
VRPLTW	**16.5258**	**1.038%**	18.1500	10.967%
VRPTW	**16.1470**	**0.724%**	17.7664	10.826%
Mean	**11.4594**	**1.386%**	12.3207	9.480%