ARC: Leveraging Compositional Representations for Cross-Problem Learning on VRPs

Han-Seul Jeong, Youngjoon Park, Hyungseok Song, Woohyung Lim
LG AI Research
Republic of Korea
{hanseul.jeong, yj.park, hyungseok.song, w.lim}@lgresearch.ai

Abstract

Vehicle Routing Problems (VRPs) with diverse real-world attributes have driven recent interest in cross-problem learning approaches that efficiently generalize across problem variants. We propose ARC (**A**ttribute **R**epresentation via **C**ompositional Learning), a cross-problem learning framework that learns disentangled attribute representations by decomposing them into two complementary components: an *Intrinsic Attribute Embedding (IAE)* for invariant attribute semantics and a *Contextual Interaction Embedding (CIE)* for attribute-combination effects. This disentanglement is achieved by enforcing analogical consistency in the embedding space to ensure the semantic transformation of adding an attribute (e.g., a length constraint) remains invariant across different problem contexts. This enables our model to reuse invariant semantics across trained variants and construct representations for unseen combinations. ARC achieves state-of-the-art performance across in-distribution, zero-shot generalization, few-shot adaptation, and real-world benchmarks.

1 Introduction

Capacitated Vehicle Routing Problem (CVRP) represents a fundamental NP-hard combinatorial optimization challenge [26, 12, 7]. While deep learning-based approximation algorithms within the Neural Combinatorial Optimization (NCO) paradigm have demonstrated near-optimal performance [1, 24, 6, 17, 13, 14, 22], real-world routing applications must address diverse attributes such as time windows [23] or open routing [25]. To efficiently leverage information of shared attributes across multiple VRP variants, recent research has focused on cross-problem learning, where a single unified model is trained to solve multiple VRP variants defined by different attribute combinations [30, 2, 15], improving efficiency and generalization compared to variant-specific models [16].

However, prior works [16, 30, 2, 15] often conflate invariant attribute semantics with contextual effects among attributes, leading to entangled representations that hinder efficient knowledge sharing across different VRP variants. To address this limitation, we propose ARC, which disentangles individual attribute embeddings by decomposing representation into intrinsic components that remain consistent across combinations and contextual components that capture combination-specific interactions. ARC learns distinct attribute representations through analogical compositional learning, ensuring identical attributes maintain their intrinsic semantics regardless of their combinations by enforcing analogous transformations across different problem contexts. Contextual components then model attribute interactions by leveraging the learned intrinsic representations within specific problem contexts, enabling efficient cross-problem learning and zero-shot generalization to unseen combinations.

Extensive experiments demonstrate that ARC outperforms existing baselines on trained configurations while achieving robust zero-shot generalization to unseen attribute combinations and efficient few-

shot adaptation to new attributes, with validation on real-world benchmarks. Our main contributions are as followed:

- We propose ARC, a novel cross-problem learning framework that disentangles attribute representations by decomposing them into intrinsic and contextual components, facilitating effective knowledge sharing across different VRP variants.

- We introduce a compositional learning mechanism that enforces analogical embedding relationships, establishing the first analogical embedding framework for NCO to our knowledge.

- We demonstrate superior performance across four scenarios: (1) in-distribution, (2) zero-shot generalization to unseen attribute combinations, (3) few-shot adaptation to new attributes, and (4) real-world benchmark, CVRPLib.

2 Related Works

Cross-Problem CO Solvers   Recent work has shifted toward cross-problem learning, developing universal architectures capable of solving diverse problems. This research spans two branches: heterogeneous CO tasks [5, 21] and VRP variants with different attribute combinations, to which our work belongs. Existing VRP approaches include joint training and Mixture-of-Experts [16, 30], foundation models [2], and attribute-aware attention mechanisms [15]. However, these methods learn mixed representations where shared attribute semantics are entangled with combination-specific interactions, inducing inefficient knowledge sharing across attribute combinations. Our approach explicitly decomposes attribute representations into intrinsic characteristics and interaction effects.

Compositional Learning   Compositional learning enables models to generalize to novel combinations by learning how individual elements can be systematically recombined. Prior approaches include modular reasoning that decomposes problems into primitive operations [11, 10], algebraic composition of value functions for skill reuse [27, 18], and representation-level compositionality that enforces compositional structure in embedding spaces through analogy-based or contrastive objectives [4, 19, 3]. Our approach leverages analogy-based compositionality in the embedding space for robust generalization across combinatorial tasks.

3 Preliminaries

3.1 Definition of VRP Variants

Each VRP variant, including the fundamental CVRP, is defined by the constraints that correspond to the attributes activated from the set of attributes introduced below. A CVRP instance $\boldsymbol{x} = (c_i, A_i)_{i \in \mathcal{V}}$ is defined on a complete graph G with a node set $\mathcal{V} = \{0, 1, \ldots, N\}$ and edge weights given by Euclidean distances d_{ij}, where a node 0 represents the depot and others correspond to customers. Each node $i \in \mathcal{V}$ is associated with coordinates $c_i \in [0,1]^2$ and attribute features A_i that define the constraints specific to VRP. The goal of VRP is to find an optimal solution $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_T)$, where $\tau_1 = \tau_T = 0$ and intermediate depot visits partition $\boldsymbol{\tau}$ into K routes. Every customer node must be visited exactly once. The objective is to minimize the total travel distance $c(\boldsymbol{\tau}) = \sum_{t=1}^{T-1} d_{\tau_t, \tau_{t+1}}$, while satisfying all constraints defined by the attribute features $\{A_i\}_{i \in \mathcal{V}}$.

Attribute Compositions   VRP variants extend the CVRP, which includes Linehaul and Capacity (Q) attributes, by combining additional active attributes. Each variant must satisfy constraints from Q and the active attributes. We consider five attributes: Backhaul (B), Mixed Backhaul (MB), Open (O), Time Window (TW), and Linehaul (L). All possible VRP variants and detailed attribute specifications are provided in Appendix A. For a instance $\boldsymbol{x}$, active attributes are represented by a binary attribute indicator vector $\mathbb{I}(\boldsymbol{x})$. For attributes B, MB, O, TW, and L, this vector is $(\mathbb{I}_B, \mathbb{I}_{MB}, \mathbb{I}_O, \mathbb{I}_{TW}, \mathbb{I}_L)$, where $\mathbb{I}_A$ is 1 if attribute A is active, 0 otherwise. For example, the Open Vehicle Routing Problem with Time Windows (OVRPTW) includes O and TW, yielding $(0, 0, 1, 1, 0)$.

3.2 Reinforcement Learning for Solving VRP Variants

We frame the VRP as a sequential decision-making process within a Markov Decision Process (MDP) framework, where solutions are constructed autoregressively. This approach aligns with unified modeling strategies for cross-problem learning explored in previous works [2, 15].
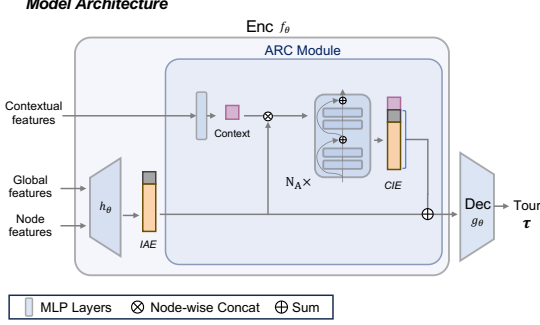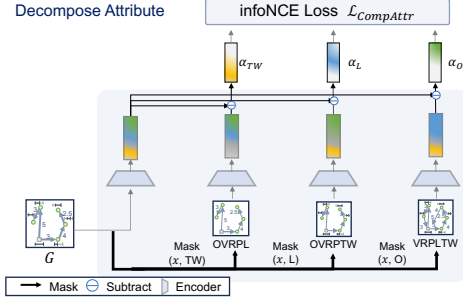
Figure 1: Overall architecture for ARC.

Figure 2: Compositional attribute representation learning

Within the MDP formulation, at step t the state $s_t = (\boldsymbol{x}, \boldsymbol{\tau}_{t-1})$ consists of the instance $\boldsymbol{x}$ and the partial solution $\boldsymbol{\tau}_{t-1} = (\tau_1, \ldots, \tau_{t-1})$. The agent selects the next node $a_t = \tau_t$ subject to feasibility, and the process starts with $\boldsymbol{\tau}_0 = \emptyset$ and terminates at $t = T$ with reward $r(\boldsymbol{\tau}) = -c(\boldsymbol{\tau})$.

We employ an autoregressive policy π_θ with an encoder $f_\theta(\boldsymbol{x})$ and a decoder $g_\theta(s_t)$. The policy defines the conditional probability $\pi_\theta(a_t|s_t) = g_\theta(\tau_t|f_\theta(\boldsymbol{x}), \boldsymbol{\tau}_{t-1})$, and the probability of generating a solution $\boldsymbol{\tau}$ is $\pi_\theta(\boldsymbol{\tau}|\boldsymbol{x}) = \prod_{t=1}^{T} \pi_\theta(a_t|s_t)$. Our goal is to maximize the expected reward $J(\theta) = \mathbb{E}_{\boldsymbol{x} \sim P(\cdot)}[\mathbb{E}_{\boldsymbol{\tau} \sim \pi_\theta(\cdot|\boldsymbol{x})}[r(\boldsymbol{\tau})]]$. We employ REINFORCE algorithm, augmented with the POMO [14] and a per-variant reward normalization scheme [2].

4 Methods: Attribute Representation via Compositional learning (ARC)

Our approach integrates the ARC module into the encoder of a standard encoder-decoder architecture [14], as depicted in Figure 1. We first identify two key properties of compositional VRPs that motivate our design (Sec. 4.1). We then detail the ARC module, which decomposes attribute representations into an *Intrinsic Attribute Embedding (IAE)* for invariant semantics and a *Contextual Interaction Embedding (CIE)* for combination-specific effects (Sec. 4.2). Finally, we introduce the compositional loss designed to learn the IAE by enforcing these properties (Sec. 4.3). Full implementation details are deferred to Appendix B.

4.1 Properties in Attributes for Compositional VRP Variants

P1: Intrinsic Semantics for Individual Attributes   An attribute possesses intrinsic and invariant semantics, maintaining the same constraint definition across all attribute combinations. For example, the attribute L enforces an identical maximum route length limit in both VRPL and OVRPL, despite their distinct underlying problem structures.

P2: Contextual Cross-Attribute Interactions   While attributes have invariant semantics (P1), their composition yields contextual interactions beyond individual attribute effects. For instance, the influence of the attribute L is significantly attenuated when co-occurring with the open-route attribute O. This is because removing the depot return in OVRPL substantially relaxes the length constraint, diminishing L's impact compared to its role in VRPL.

4.2 Attribute Representation via Compositional Learning (ARC)

To explicitly encode properties P1 and P2, our encoder decomposes the final representation $f_\theta(\boldsymbol{x})$ into two components: an *Intrinsic Attribute Embedding (IAE)*, $h_\theta(\boldsymbol{x})$, and a *Contextual Interaction Embedding (CIE)*, $m_\theta(\boldsymbol{x})$, formulated as $f_\theta(\boldsymbol{x}) = h_\theta(\boldsymbol{x}) + m_\theta(\boldsymbol{x})$. The IAE is trained via a compositional loss (Sec. 4.3) to capture the intrinsic semantics of individual attributes (P1), ensuring a consistent representation for an attribute across all problem contexts. The CIE, in contrast, captures the contextual interactions (P2). Conditioned on the IAE, it utilizes an attention mechanism over contextual features (i.e., attribute indicators and global features) to produce context-specific representations. This approach allows our model to reuse the invariant semantics across variants, overcoming a key limitation of prior methods that learn mixed representations.

4.3 Compositional Learning for Intrinsic Attribute Embeddings

To disentangle semantic attribute representations, we enable the model to learn analogical concepts. This section introduces the concept of analogy relationships between attribute combinations and describes our compositional loss designed to encode these relationships in the embedding space.

Analogy-Making over Attributes   An analogy "A is to B as C is to D" (denoted as A:B::C:D) captures the relationship between pairs, suggesting that the transformation from A to B parallels that from C to D. Let x and y be two base VRP problem where an arbitrary attribute A is not active. We denote the extended VRP variants with attribute A activated as $[x+A]$ and $[y+A]$, respectively. The intrinsic semantic of A in both $[x+A]$ and $[y+A]$, as highlighted in (P1), can be expressed through the analogy: $[x+A]:[x]::[y+A]:[y]$. For example, if x represents CVRP, y represents OVRP, and A is length constraint L, this establishes the analogy VRPL:CVRP::OVRPL:OVRP. This analogical property can be expressed in the embedding space as: $h_\theta(x+A) - h_\theta(x) \approx h_\theta(y+A) - h_\theta(y)$.

Learning Compositional Attribute Representation   To enforce the aforementioned analogical consistency on the IAE ($h_\theta(x)$), we employ a contrastive learning objective, InfoNCE loss [20]. The key insight is that the intrinsic semantic of an attribute A —represented by the attribute vector $\alpha_A := h_\theta(x + A) - h_\theta(x)$—should be identifiable and consistent regardless of the base instance x or other activated attributes. It enforces this analogical consistency by distinguishing between same and different attribute semantics. As illustrated in Fig. 2, we firstly extract these attribute vectors by masking the attribute features from the same instance x. For a given attribute A, a transformation vector $h_\theta(x' + A) - h_\theta(x')$ derived from a different instance x' serves as a positive sample, while vectors corresponding to different attributes A' form negative samples. This objective encourages the model to learn a context-invariant representation for each attribute's intrinsic semantics. This compositional loss, $\mathcal{L}_{\text{CompAttr}}(\theta)$ (formally defined in Appendix B.3), is added to the reward $J(\theta)$: $J(\theta) - \lambda \cdot \mathcal{L}_{\text{CompAttr}}(\theta)$.

5 Experiments

Baselines   We compare against PyVRP [8], a state-of-the-art hybrid genetic search metaheuristic based on HGS [28], and recent neural cross-problem VRP solvers: MTPOMO [16], MVMoE [30], RouteFinder (RF-TE) [2], and CaDA [15]. Since our method extends RouteFinder with an additional ARC module, RouteFinder can be considered an ablation of our method. Comparison with CaDA, which claims constraint-awareness, evaluates our embedding decomposition approach.

Experimental Setup   Following [2, 15], we train on graphs with $N = 50$ or 100 nodes using 100,000 instances per epoch, with equal proportions across VRP variants. We evaluate on 1,000 test instances per variant and follow RouteFinder's data generation and training protocols. The results of all neural approaches are averaged over three independent runs. Most neural baselines share a unified codebase[1], and implementations of CaDA and our method are available[2]. Detailed hyperparameters are provided in Appendix C.1. We report **Gap** as the percentage cost increase relative to PyVRP's best solution and **Time** as the total duration to solve all test instances in a single run.

5.1 Experimental Scenarios

In-distribution   Models are trained and tested on the identical set of 16 VRP variants, encompassing all combinations of four base attributes (B, O, L, TW). This scenario evaluates the ability to capture shared attribute semantics and leverage cross-task knowledge from familiar variants.

Out-of-distribution   (1) **Zero-shot Generalization.** To evaluate performance when some attribute combinations are not provided during training, we restrict the training problem types based on the In-distribution setting. Models are trained on seven representative variants (CVRP, OVRP, VRPB, VRPL, VRPTW, OVRPTW, VRPBL), selected from the limited VRP variants in [30] with VRPBL added to maintain equal proportions across different attributes. We then evaluate zero-shot performance on the remaining nine variants to test compositional generalization to novel, complex attribute combinations. (2) **Few-shot Adaptation.** Models pre-trained on In-distribution setting are extended with Efficient Adapter Layers (EAL) [2] to address the unseen attribute, MB, then fine-tuned on 10,000 VRP

---

[1] https://github.com/ai4co/routefinder
[2] https://github.com/hanseul-jeong/ARC

Table 1: In-distribution Performance on 1K test instances. Bold and underline denote best and second-best, respectively. * marks the reference solution used as baseline for gap calculations.

	Solver	n = 50 Gap (%)	Time	n = 100 Gap (%)	Time	Solver	n = 50 Gap (%)	Time	n = 100 Gap (%)	Time
CVRP	HGS-PyVRP	*	10m	*	21m	**VRPTW**	*	10m	*	21m
	MTPOMO	1.407 ± 0.01	1s	2.049 ± 0.04	7s		2.429 ± 0.01	1s	3.934 ± 0.01	8s
	MVMoE	$\underline{1.226 \pm 0.00}$	2s	1.646 ± 0.02	9s		2.337 ± 0.02	2s	3.876 ± 0.01	10s
	RF-TE	1.239 ± 0.05	1s	1.559 ± 0.01	11s		1.933 ± 0.04	1s	3.192 ± 0.06	10s
	CaDA	1.259 ± 0.05	3s	$\underline{1.513 \pm 0.02}$	15s		$\underline{1.927 \pm 0.07}$	3s	$\underline{3.025 \pm 0.04}$	16s
	ARC	$\mathbf{1.154 \pm 0.04}$	1s	$\mathbf{1.429 \pm 0.02}$	10s		$\mathbf{1.772 \pm 0.07}$	1s	$\mathbf{2.840 \pm 0.04}$	11s
OVRP	HGS-PyVRP	*	10m	*	21m	**VRPL**	*	10m	*	21m
	MTPOMO	3.213 ± 0.02	1s	5.102 ± 0.05	7s		1.718 ± 0.03	1s	2.486 ± 0.04	7s
	MVMoE	2.915 ± 0.03	2s	4.608 ± 0.02	9s		1.508 ± 0.00	2s	2.065 ± 0.03	9s
	RF-TE	2.645 ± 0.07	1s	4.133 ± 0.01	11s		$\underline{1.434 \pm 0.07}$	1s	1.881 ± 0.02	9s
	CaDA	$\underline{2.626 \pm 0.08}$	3s	4.029 ± 0.01	15s		1.481 ± 0.07	3s	$\underline{1.848 \pm 0.02}$	15s
	ARC	$\mathbf{2.497 \pm 0.06}$	1s	$\mathbf{3.915 \pm 0.02}$	11s		$\mathbf{1.370 \pm 0.07}$	1s	$\mathbf{1.753 \pm 0.02}$	11s
VRPB	HGS-PyVRP	*	10m	*	21m	**OVRPTW**	*	10m	*	21m
	MTPOMO	3.612 ± 0.02	1s	4.986 ± 0.06	7s		1.564 ± 0.02	1s	3.023 ± 0.03	8s
	MVMoE	3.234 ± 0.04	2s	4.484 ± 0.01	9s		1.528 ± 0.03	2s	2.944 ± 0.06	10s
	RF-TE	2.984 ± 0.08	1s	3.999 ± 0.02	9s		1.285 ± 0.04	1s	2.353 ± 0.07	11s
	CaDA	$\underline{2.973 \pm 0.09}$	3s	$\underline{3.948 \pm 0.02}$	15s		$\underline{1.240 \pm 0.04}$	3s	$\underline{2.289 \pm 0.06}$	16s
	ARC	$\mathbf{2.840 \pm 0.10}$	1s	$\mathbf{3.833 \pm 0.04}$	11s		$\mathbf{1.087 \pm 0.04}$	1s	$\mathbf{1.968 \pm 0.04}$	12s
VRPBL	HGS-PyVRP	*	10m	*	21m	**VRPBLTW**	*	10m	*	21m
	MTPOMO	4.681 ± 0.02	1s	6.304 ± 0.04	7s		2.154 ± 0.01	1s	3.791 ± 0.03	8s
	MVMoE	4.283 ± 0.03	2s	5.718 ± 0.02	9s		2.102 ± 0.02	2s	3.724 ± 0.03	10s
	RF-TE	3.717 ± 0.11	1s	5.020 ± 0.03	10s		1.754 ± 0.04	1s	2.950 ± 0.03	11s
	CaDA	$\underline{3.696 \pm 0.10}$	3s	$\underline{4.953 \pm 0.03}$	15s		$\underline{1.741 \pm 0.05}$	4s	$\underline{2.814 \pm 0.05}$	16s
	ARC	$\mathbf{3.579 \pm 0.13}$	1s	$\mathbf{4.856 \pm 0.01}$	11s		$\mathbf{1.599 \pm 0.05}$	1s	$\mathbf{2.613 \pm 0.04}$	12s
VRPBTW	HGS-PyVRP	*	10m	*	21m	**VRPLTW**	*	10m	*	21m
	MTPOMO	1.922 ± 0.01	1s	3.416 ± 0.03	8s		2.857 ± 0.02	1s	4.429 ± 0.01	8s
	MVMoE	1.885 ± 0.03	2s	3.360 ± 0.04	10s		2.747 ± 0.04	2s	4.343 ± 0.03	10s
	RF-TE	1.534 ± 0.02	1s	2.597 ± 0.04	11s		$\underline{2.280 \pm 0.05}$	1s	3.620 ± 0.06	11s
	CaDA	$\underline{1.527 \pm 0.04}$	3s	$\underline{2.451 \pm 0.03}$	16s		2.322 ± 0.08	3s	$\underline{3.453 \pm 0.03}$	16s
	ARC	$\mathbf{1.375 \pm 0.06}$	1s	$\mathbf{2.281 \pm 0.04}$	11s		$\mathbf{2.109 \pm 0.06}$	1s	$\mathbf{3.263 \pm 0.05}$	11s
OVRPB	HGS-PyVRP	*	10m	*	21m	**OVRPBL**	*	10m	*	21m
	MTPOMO	3.000 ± 0.01	1s	5.344 ± 0.03	7s		3.087 ± 0.00	1s	5.434 ± 0.03	7s
	MVMoE	2.706 ± 0.02	2s	4.791 ± 0.02	9s		2.776 ± 0.04	2s	4.866 ± 0.00	9s
	RF-TE	2.445 ± 0.10	1s	4.288 ± 0.06	10s		2.463 ± 0.10	1s	4.301 ± 0.06	10s
	CaDA	$\underline{2.396 \pm 0.09}$	3s	$\underline{4.121 \pm 0.03}$	15s		$\underline{2.397 \pm 0.09}$	3s	$\underline{4.107 \pm 0.02}$	15s
	ARC	$\mathbf{2.231 \pm 0.10}$	1s	$\mathbf{3.935 \pm 0.04}$	11s		$\mathbf{2.236 \pm 0.09}$	1s	$\mathbf{3.936 \pm 0.04}$	11s
OVRPBLTW	HGS-PyVRP	*	10m	*	21m	**OVRPBTW**	*	10m	*	21m
	MTPOMO	1.317 ± 0.01	1s	2.649 ± 0.03	8s		1.295 ± 0.01	1s	2.622 ± 0.03	8s
	MVMoE	1.289 ± 0.03	2s	2.615 ± 0.06	11s		1.289 ± 0.04	2s	2.604 ± 0.07	11s
	RF-TE	1.073 ± 0.02	1s	1.998 ± 0.06	11s		1.059 ± 0.03	1s	1.997 ± 0.06	11s
	CaDA	$\underline{0.994 \pm 0.03}$	4s	$\underline{1.825 \pm 0.05}$	16s		$\underline{0.987 \pm 0.03}$	4s	$\underline{1.830 \pm 0.05}$	16s
	ARC	$\mathbf{0.905 \pm 0.05}$	1s	$\mathbf{1.641 \pm 0.03}$	12s		$\mathbf{0.902 \pm 0.05}$	1s	$\mathbf{1.639 \pm 0.03}$	12s
OVRPL	HGS-PyVRP	*	10m	*	21m	**OVRPLTW**	*	10m	*	21m
	MTPOMO	3.233 ± 0.04	1s	5.154 ± 0.06	7s		1.574 ± 0.01	1s	3.032 ± 0.03	8s
	MVMoE	2.966 ± 0.02	2s	4.657 ± 0.02	9s		1.549 ± 0.03	2s	2.963 ± 0.05	10s
	RF-TE	2.636 ± 0.06	1s	4.129 ± 0.02	10s		1.288 ± 0.04	1s	2.356 ± 0.07	11s
	CaDA	$\underline{2.635 \pm 0.09}$	3s	$\underline{4.029 \pm 0.02}$	15s		$\underline{1.249 \pm 0.05}$	3s	$\underline{2.290 \pm 0.06}$	16s
	ARC	$\mathbf{2.510 \pm 0.06}$	1s	$\mathbf{3.916 \pm 0.02}$	11s		$\mathbf{1.084 \pm 0.04}$	1s	$\mathbf{1.953 \pm 0.04}$	12s

instances with the MB attribute over 10 epochs. We compare against EAL-compatible baselines RF-TE and CaDA to assess few-shot transfer to this new constraint. (3) **Real-world Benchmark.** To validate generalization of our synthetic-trained model to real-world instances with different scales and distributions, we evaluated on 115 instances from CVRPLib[3] datasets featuring node sizes ranging from 16 to 200 and diverse distribution characteristics differing from the training data as in [15].

5.2 Results

In-distribution As shown in Table 1, ARC consistently outperforms all neural baselines across all 16 VRP variants, achieving results of 1.828% and 2.861% for the instance sizes 50 and 100, respectively, without significant inference overhead and with detailed time complexity analysis provided in Appendix D.2. This superior performance demonstrates that ARC structurally encodes VRP characteristics at the attribute level, effectively capturing shared semantic information to interpret complex VRP variants. These experimental results provide strong empirical evidence that our proposed ARC effectively addresses the limitations of existing SOTA baselines such as MVMoE, MTPOMO, RF-TE, and CaDA, which do not utilize explicit compositional learning like ours.

Zero-shot Generalization Table 2 shows the average performance across multiple tasks in Zero-shot settings, with complete results provided in the Appendix C.2. ARC achieves superior average performance, with MVMoE becoming the second-best algorithm compared to in-distribution results where CaDA was second-best. This shift demonstrates that compositional learning approaches

---

[3] http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Table 2: Performance on 1K test Zero-shot instances. (Bold: best, Underline: second-best)

	Solver	$n = 50$	$n = 100$
Average	MTPOMO	5.160 ± 0.69	7.630 ± 1.21
	MVMoE	$\underline{4.613 \pm 0.59}$	7.524 ± 1.06
	RF-TE	5.065 ± 0.67	7.843 ± 0.77
	CaDA	6.740 ± 1.20	8.044 ± 1.39
	ARC	$\mathbf{4.078 \pm 0.42}$	$\mathbf{6.422 \pm 0.74}$

Table 3: Performance on CVRPLib test instances. Bold denotes best.

Group	MTPOMO	MVMoE	RF-TE	CaDA	ARC
A	3.23%	3.07%	2.82%	3.25%	**2.50%**
B	3.80%	3.89%	2.58%	2.94%	**2.42%**
E	8.12%	7.38%	2.93%	3.81%	**2.82%**
F	10.52%	12.16%	12.95%	11.96%	**8.93%**
M	5.61%	5.31%	**5.08%**	5.54%	5.55%
P	7.87%	6.76%	4.57%	5.15%	**3.22%**
X	5.94%	5.23%	4.48%	4.57%	**4.37%**
Average	6.44%	6.26%	5.06%	5.32%	**4.26%**
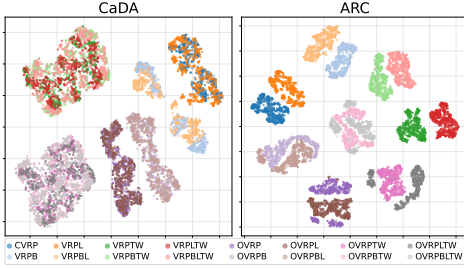
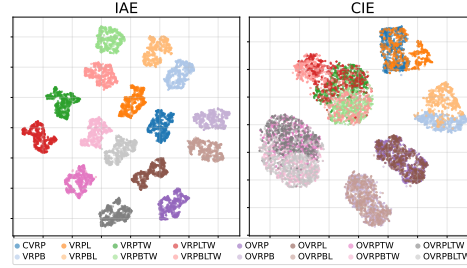Figure 3: t-SNE of emb. across 16 VRP variants ($N = 50$): CaDA (left) vs. ARC (right).

Figure 4: t-SNE of ARC emb. across 16 VRP variants (N = 50): IAE (left) vs. CIE (right)

(MVMoE and ARC) are more effective for generalization to unseen combinations, with ARC showing consistently strong performance across diverse tasks without large variance.

Few-shot Adaptation Table 4 shows that our model consistently attains superior performance across all tasks with newly added attributes MB, significantly outperforming RF-TE and CaDA. Our attribute embeddings inherently preserve compatibility with unseen attribute combinations, enabling efficient reuse of learned representations for rapid adaptation to novel attributes.

Real-World Benchmark We compared the test performance of In-distribution neural solvers trained on $N = 100$. As shown in Table 3, our method achieved the best performance across all sets except Group M, demonstrating that learning attribute-specific characteristics helps achieve superior generalization to instances with different distributional properties. However, the reduced margins in Table 9 highlight the scaling challenge, identifying a direction for future research [9, 29].

Visualizing Embeds To investigate the encoder's representation, we apply t-SNE to visualize the embeddings $f_\theta(x)$ of VRP instances across problem variants. Figure 3 shows ARC forms well-separated clusters corresponding to respective VRP variants, while CaDA shows intermixed clusters with blurred attribute boundaries. Figure 4 confirms our decomposition: IAE clusters by attribute type while CIE shows mixed patterns, validating the separation of intrinsic and contextual components.

Table 4: Few-shot Adaptation Performance on 1K test instances ($N = 50$). Bold denotes best.

Model	OVRPMB	OVRPMBL	OVRPMBLTW	OVRPMBTW	VRPMB	VRPMBL	VRPMBLTW	VRPMBTW
RF-TE	4.453 ± 0.13	4.511 ± 0.10	1.498 ± 0.02	1.486 ± 0.02	4.432 ± 0.13	3.456 ± 0.01	2.267 ± 0.02	1.945 ± 0.02
CaDA	6.440 ± 1.00	6.407 ± 0.99	1.406 ± 0.02	1.391 ± 0.02	4.160 ± 0.48	4.380 ± 0.51	2.166 ± 0.01	1.853 ± 0.01
ARC	$\mathbf{1.475 \pm 0.06}$	$\mathbf{1.506 \pm 0.06}$	$\mathbf{1.277 \pm 0.04}$	$\mathbf{1.285 \pm 0.05}$	$\mathbf{1.677 \pm 0.05}$	$\mathbf{1.925 \pm 0.08}$	$\mathbf{2.062 \pm 0.03}$	$\mathbf{1.749 \pm 0.06}$

6 Conclusion

We introduced ARC, a compositional cross-problem learning framework for VRPs that disentangles attribute representations by decomposing them into intrinsic and contextual components. By enforcing analogical relationships in embedding spaces, ARC enables effective knowledge sharing across problem variants and achieves superior zero-shot generalization to unseen combinations. Extensive experiments demonstrate consistent improvements over existing baselines across in-distribution, zero-shot generalization, and few-shot adaptation, with validation on real-world benchmarks. This work establishes analogical embeddings as an effective approach for cross-problem learning in NCO. Future work will focus on extending this compositional approach to address the scalability challenges of massive-scale, realistic problem instances.

References

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

[2] 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.

[3] Sachin Chanchani and Ruihong Huang. Composition-contrastive learning for sentence embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15836–15848. Association for Computational Linguistics, 2023.

[4] Coline Devin, Daniel Geng, Pieter Abbeel, Trevor Darrell, and Sergey Levine. Plan arithmetic: compositional plan vectors for multi-task control. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 14989–15000, 2019.

[5] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. In *The Thirteenth International Conference on Learning Representations*, 2024.

[6] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:77416–77429, 2023.

[7] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[8] Maaike Hoogeboom, Francesco Corman, Wouter Kool, Yves Pochet, and Daniele Vigo. Pyvrp: A high-performance vrp solver package. *Transportation Science*, 2023.

[9] Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. In *The Thirteenth International Conference on Learning Representations*, 2025.

[10] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[11] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[12] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:1936–1949, 2022.

[13] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

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

[15] Han Li, Fei Liu, Zhi Zheng, Yu Zhang, and Zhenkun Wang. Cada: Cross-problem routing solver with constraint-aware dual-attention. *arXiv preprint arXiv:2412.00346*, 2024.

[16] Fei Liu, Xi Lin, Zhenkun Wang, Qingfu Zhang, Tong Xialiang, and Mingxuan Yuan. 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*, pages 1898–1908, 2024.

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

[18] Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9497–9507, 2020.

[19] Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2661–2670, 2017.

[20] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[21] Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. Unico: On unified combinatorial optimization via problem reduction to matrix-encoded general tsp. In *The Thirteenth International Conference on Learning Representations*, 2025.

[22] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35:25531–25546, 2022.

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

[24] Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

[25] Christos D Tarantilis, George Ioannou, Chris T Kiranoudis, and Gregory P Prastacos. A threshold accepting approach to the open vehicle routing problem. *RAIRO-Operations Research*, 38(4):345–360, 2004.

[26] Paolo Toth and Daniele Vigo. *Vehicle Routing Problems*. SIAM, 2002.

[27] Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6401–6409, 2019.

[28] Thibaut Vidal, Teodor G Crainic, Michel Gendreau, and Christian Prins. A unified hybrid genetic search for multi-depot vehicle routing problems. *Computers & Operations Research*, 39(9):2289–2301, 2012.

[29] Yuepeng Zheng, Fu Luo, Zhenkun Wang, Yaoxin Wu, and Yu Zhou. Mtl-kd: Multi-task learning via knowledge distillation for generalizable neural vehicle routing solver. *arXiv preprint arXiv:2506.02935*, 2025.

[30] Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. Mvmoe: Multi-task vehicle routing solver with mixture-of-experts. In *Proceedings of the 41st International Conference on Machine Learning*, pages 61804–61824, 2024.

A Attributes

In this section, we describe the details of attributes we utilized. We adopt the classical route-set notation, which is more convenient for presenting mathematical properties and constraints of VRP variants. In contrast, the main text described solutions using the sequence notation $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_T)$, where depot visits partition the sequence into K vehicle routes, a representation well suited for reinforcement learning as it aligns with the step-by-step construction of solutions. Here, however, a solution is represented as $\boldsymbol{\tau} = \{\tau^1, \tau^2, \ldots, \tau^K\}$, consisting of K routes, where each route $\tau^k = (\tau_0^k, \tau_1^k, \ldots, \tau_{n_k}^k)$ starts and ends at the depot ($\tau_0^k = \tau_{n_k}^k = 0$). Every customer node appears in exactly one route, and the total cost is $c(\boldsymbol{\tau}) = \sum_{k=1}^{K} \sum_{i=0}^{n_k-1} d_{\tau_i^k, \tau_{i+1}^k}$.

Based on this notation, we now detail the attributes $\{A_i\}_{i \in \mathcal{V}}$ that define different VRP variants.

Linehaul and Capacity (Q)   In CVRP, customer nodes ($i > 0$) have a non-negative demand q_i, representing linehaul services (e.g., deliveries), with $A_i = \{q_i\}$. Vehicles have uniform capacity $Q > 0$. The capacity attribute requires that for each route τ^k, the sum of customer demands $\sum_{j \in \tau^k, j \neq 0} q_j$ must not exceed Q.

Open (O)   Vehicles are not required to return to the depot after serving the last customer, i.e., $\tau_{n_k}^k \neq 0$.

Backhaul (B) and Mixed Backhaul (MB)   Unlike the CVRP where only linehaul customers ($q_i \geq 0$) are present, Backhaul or Mixed Backhaul variants also include backhaul customers (pickup, $q_i < 0$), requiring transportation back to the depot. Each route τ^k must satisfy one of two mutually exclusive attributes: Backhaul (B), requiring all linehaul customers be visited before backhaul customers, or Mixed Backhaul (MB), allowing any order. This attribute, $\mu \in \{0, 1\}$ indicating B or MB, is a global feature included in $A_0 = \{\mu\}$.

Time Window (TW)   Each customer node $i \neq 0$ must be visited within a time interval $[e_i, l_i] \in [0, \mathcal{T}]^2$, with a service time $s_i \in [0, \mathcal{T}]$. Customer features are thus defined as $A_i = \{e_i, l_i, s_i\}$. Vehicles arriving before the earliest available time e_i must wait. Service takes s_i time units, after which the vehicle proceeds. The depot has a time window $[0, \mathcal{T}]$, where $\mathcal{T}$ is the time horizon. If TW is not activated, $e_i = 0, l_i = \infty, s_i = 0$ are set for all customers i.

Duration Limit (L)   Each route's total cost $c(\tau^k)$ must not exceed a limit L, i.e., $c(\tau^k) \leq L$. If not activated, $L = \infty$. The global features A_0 include L, i.e., $A_0 = \{L\}$.

Figure 5 provides a visual explanation of the attributes. The complete set of VRP variants that can be constructed using these attributes is shown in Table 5.
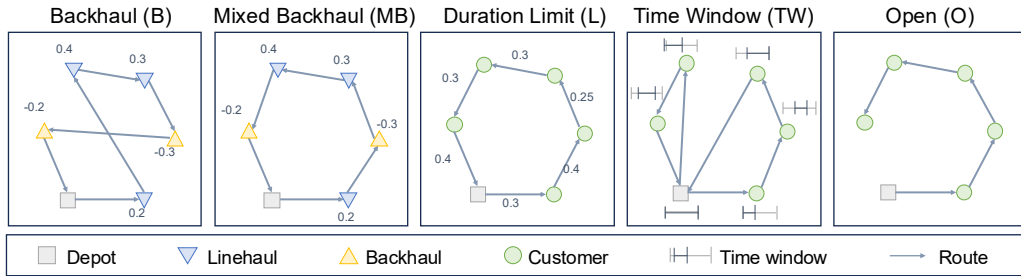

Figure 5: llustration of VRP attributes whose combinations define respective VRP variants.

B Architecture Details

We employed a transformer-based attribute composition model, which represents a general architecture for cross-problem VRP variants [16, 2, 15]. This approach extends single-problem models to multi-task settings by incorporating various attribute information into the structure of single models [13, 14, 12].

Table 5: VRP Variants and Their Attribute Combinations

VRP Variant	Capacity (Q)	Open Route (O)	Backhaul (B)	Mixed (M)	Duration Limit (L)	Time Windows (TW)
CVRP	✓					
OVRP	✓	✓				
VRPB	✓		✓			
VRPL	✓				✓	
VRPTW	✓					✓
OVRPTW	✓	✓				✓
OVRPB	✓	✓	✓			
OVRPL	✓	✓			✓	
VRPBL	✓		✓		✓	
VRPBTW	✓		✓			✓
VRPLTW	✓				✓	✓
OVRPBL	✓	✓	✓		✓	
OVRPBTW	✓	✓	✓			✓
OVRPLTW	✓	✓			✓	✓
VRPBLTW	✓		✓		✓	✓
OVRPBLTW	✓	✓	✓		✓	✓
VRPMB	✓		✓	✓		
OVRPMB	✓	✓	✓	✓		
VRPMBL	✓		✓	✓	✓	
VRPMBTW	✓		✓	✓		✓
OVRPMBL	✓	✓	✓	✓	✓	
OVRPMBTW	✓	✓	✓	✓		✓
VRPMBLTW	✓		✓	✓	✓	✓
OVRPMBLTW	✓	✓	✓	✓	✓	✓

B.1 Encoder

The encoder f_θ comprises two principal structures: a Node Embedder h_θ and an ARC module. The encoder separately processes two distinct categories of information: global and node-specific attributes. Global attributes including open $o \in \{0, 1\}$, duration limit $d_l \in \{0, L_{\max}\}$, and mixed backhaul $\mu \in \{0, 1\}$, (employed in EAL) represent problem-level constraints, whereas node-level constraints such as linehaul demand q^l, backhaul demand q^b, time window (start time e, end time s, and service time l) are associated with individual nodes. The global attribute information is incorporated into the depot representation, while node attribute information $\phi^n = \{\phi_0^n, \cdots, \phi_{H_n-1}^n\}$ is integrated into each customer node i, yielding the initial node embedding $e_i^{(0)}$ as follows:

$$e_i^{(0)} = \begin{cases} W_g[c_0, o, d_l, \mu, l_0]^T & \text{if } i = 0, \\ W_n[c_i, q_i^l, q_i^b, e_i, l_i, s_i]^T & \text{otherwise,} \end{cases} \tag{1}$$

where $W^g \in \mathbb{R}^{E_e \times (H_g+2)}$ and $W^n \in \mathbb{R}^{E_e \times (H_n+2)}$ are learnable linear layers, E_e is the embedding dimension, H_g and H_n are the numbers of global and node-level attributes respectively, and l_0 denotes the time window end value of the depot.

B.1.1 Node Embedder

The node embedding are passed through a NodeEmbedder h_θ which consists of N_E transformer-based NodeEmbBlocks with the following structure:

$$e^{(N_E)} = h_\theta(e^{(0)}) \tag{2}$$
$$= (\text{NodeEmbBlock}_{N_E} \circ \text{NodeEmbBlock}_{N_E-1} \circ \cdots \circ \text{NodeEmbBlock}_1)(e^{(0)}). \tag{3}$$

Each NodeEmbBlock consists of two sub-layers: a Multi-Head Attention (MHA) layer and a Feed Forward ParallelGatedMLP layer. The MHA layer captures dependencies between different positions in the input sequence and the ParallelGatedMLP layer applies non-linear transformations to the features.

$$\hat{e}^{(\ell-1)} = e^{(\ell-1)} + \text{MHA}(\text{RMSNorm}(e^{(\ell-1)}), \text{RMSNorm}(e^{(\ell-1)})), \tag{4}$$

$$e^{(\ell)} = \hat{e}^{(\ell-1)} + \text{ParallelGatedMLP}(\text{RMSNorm}(\hat{e}^{(\ell-1)})), \tag{5}$$

where $e^{(\ell-1)}$ denotes the input to the ℓ-th NodeEmbBlock and $\text{MHA}(a, b)$ denotes Multi-Head Attention of which a serves as the query, while b provides the keys and values, and RMSNorm is a RMS normalization.

The ParallelGatedMLP function is defined as:

$$\text{ParallelGatedMLP}(x) = W_{p_3}(\text{SiLU}(W_{p_1}x) \odot (W_{p_2}x)), \tag{6}$$

where $\odot$ denotes element-wise multiplication, SiLU is the Sigmoid Linear Unit (Swish) activation function, and $W_{p_1}, W_{p_2}, W_{p_3}$ are learnable linear layers. Consistent with prior work [2, 15], we use $N_E = 6$ layers.

B.1.2 ARC Module

The ARC module utilizes the final output of Node Embedder (i.e., IAE), attribute indicator, and global attribute features. Given an input x with attribute indicator $\mathbb{I}^{\text{attr}} = (\mathbb{I}_B, \mathbb{I}_{\text{MB}}, \mathbb{I}_O, \mathbb{I}_{\text{TW}}, \mathbb{I}_L)$, where $\mathbb{I}_A$ is 1 if attribute A is active and 0 otherwise, the initial CIE is defined as:

$$\mathbf{m}^{(0)} = W_{m_2}\text{LayerNorm}(W_{m_1} \cdot \text{Concat}[\mathbb{I}^{\text{attr}}, o, dl, \mu]^T), \tag{7}$$

where W_{m_1}, W_{m_2} are learnable linear layers, Concat is a feature-wise concatenation and LayerNorm is a layer normalization.

The layers of ARC module, composed of N_A MixerBlocks, takes IAE $\mathbf{e}$ and CIE $\mathbf{m}$ as inputs and employs a structure similar to the Global Layer [15]. The ℓ-th layer MixerBlock takes global embedding $\mathbf{m}^{(\ell-1)}$ and IAE $\mathbf{e}_m^{(\ell-1)}$ as inputs (i.e., $\mathbf{e}_m^{(0)} = \mathbf{e}^{(N_E)}$), The MixerBlock is formulated as follows:

$$\hat{\mathbf{m}}^{(\ell-1)} = \text{GlobalModule}(\mathbf{m}^{(\ell-1)}, \mathbf{e}_m^{(\ell-1)}), \tag{8}$$

$$\hat{\mathbf{e}}_m^{(\ell-1)} = \text{GlobalModule}(\mathbf{e}_m^{(\ell-1)}, \mathbf{m}^{(\ell-1)}), \tag{9}$$

$$\mathbf{m}^{(\ell)} = \hat{\mathbf{m}}^{(\ell-1)} + (\hat{\mathbf{e}}_m^{(\ell-1)}W_{g_1}), \tag{10}$$

$$\mathbf{e}_m^{(\ell)} = \hat{\mathbf{e}}_m^{(\ell-1)} + (\hat{\mathbf{m}}^{(\ell-1)}W_{g_2}), \tag{11}$$

with W_{g_1} and W_{g_2} are learnable linear layers. The $\text{GlobalModule}(a, b)$, which takes primary input a and auxiliary input b to produce output $\hat{a}$, is computed as follows:

$$\tilde{a} = \text{RMSNorm}\left(a + \text{MHA}\left(a, \text{Concat}[a, b]\right)\right), \tag{12}$$

$$\hat{a} = \text{RMSNorm}\left(\tilde{a} + \text{ParallelGatedMLP}\left(\tilde{a}\right)\right). \tag{13}$$

The final output embedding of our encoder, denoted as $f_\theta(x)$, is computed by combining the IAE $h_\theta(x)$ and the CIE $m_\theta(x)$. Specifically, we sum these two outputs:

$$f_\theta(\boldsymbol{x}) = h_\theta(\boldsymbol{x}) + m_\theta(\boldsymbol{x}). \tag{14}$$

B.2 Decoder

The decoder outputs the action probability for each node at each t-th step based on the encoded node embeddings. We compute the context embedding $\boldsymbol{g}_c$ using the embedding of the previously selected node τ_{t-1} and the attribute feature values at t-th step as follows:

$$\boldsymbol{g}_c = W_d \cdot \text{Concat}[\boldsymbol{h}_{\tau_{t-1}}, c_t^l, c_t^b, z_t, l_t, o_t]^T, \tag{15}$$

11

where c_t^l, c_t^b are remaining capacity of vehicle for linehaul and backhaul, respectively and z_t, l_t, o_t are current time, the remaining length of partial solution, and the indicator of the open route, respectively.

We calculate the probability of selecting each node using the context embedding as a query and the previously encoded values $\boldsymbol{h}$ as key and value. To generate feasible solutions, we mask nodes that cannot be visited at each decoding step based on activated constraints. The probability values u_i for actions are calculated as follows:

$$q_c = \text{MHA}(\boldsymbol{g}_c, \boldsymbol{h}_{0:N}), \tag{16}$$

$$u_i = \begin{cases} \xi \cdot \tanh \left(\frac{q_c(\boldsymbol{h}_i)^\top}{\sqrt{\text{E}q}} \right) & \text{if } i \in I_t, \\ -\infty & \text{otherwise,} \end{cases} \tag{17}$$

where I_t is a feasible node set at step t, ξ is a clipping hyperparameter, and E_q is the dimension of query $\boldsymbol{g}_c$. We compute action probability by applying Softmax to probability value u_i.

B.3 Learning Compositional Attribute Representation

Given a function $\mathcal{A}(\cdot)$ that represents the set of activated attributes in a problem instance $\boldsymbol{x}$ from a batch containing instances of various problem types, we can extract an attribute vector α for any non-empty subset $\text{A} \subseteq \mathcal{A}(\boldsymbol{x})$ (where $\mathcal{A}(\boldsymbol{x}) \neq \emptyset$) as follows:

$$\alpha_\text{A} = h_\theta(\boldsymbol{x}) - h_\theta(\text{mask}(\boldsymbol{x}, \text{A})), \tag{18}$$

where $h_\theta(\cdot)$ is the Node Embedder and $\text{mask}(\boldsymbol{x}, \text{A})$ is a masking function that removes the feature of attribute A from instance $\boldsymbol{x}$. We construct an attribute pool $\mathcal{P}$ by collecting all attribute vectors from all instances in the batch. For any attribute vector α drawn from the pool, we classify attribute vectors corresponding to the same attribute types as the positive class and those corresponding to different attribute types as the negative class. We sample one attribute vector from the positive class as the positive sample α^+. If the size of the negative class for α is B, we compute the compositional attribute loss as follows:

$$\mathcal{L}_{\text{CompAttr}}(\theta) = -\mathbb{E}_{\alpha \sim \mathcal{P}} \left[\log \frac{\exp(f(\alpha, \alpha^+)/\beta)}{\exp(f(\alpha, \alpha^+)/\beta) + \sum_{j=1}^{B} \exp(f(\alpha, \alpha_j^-)/\beta)} \right], \tag{19}$$

where β is a temperature. The pseudocode for this process is presented in Algorithm 1.

Algorithm 1 Compute Compositional Loss.

Require: Batch $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_B\}$, temperature β, attribute function $\mathcal{A}(\cdot)$
1: $\mathcal{P} \leftarrow \{\}$ ▷ Initialize attribute pool
2: **for** $i = 1$ to B **do**
3:      **for** $\mathtt{A} \in \mathcal{A}(\boldsymbol{x}_i)$ **do**
4:          $\alpha \leftarrow h_\theta(\boldsymbol{x}_i) - h_\theta(\mathtt{mask}(\boldsymbol{x}_i, \mathtt{A}))$ ▷ Extract attribute vector
5:          $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\alpha, \mathtt{A})\}$
6:      **end for**
7: **end for**
8: $\mathcal{L} \leftarrow 0$ ▷ Initialize Compositional Loss
9: **for** $(\alpha, \mathtt{A}) \in \mathcal{P}$ **do**
10:      $\alpha^+ \sim \{\alpha' \mid (\alpha', \mathtt{A}') \in \mathcal{P}, \mathtt{A}' = \mathtt{A}\} \setminus \{\alpha\}$ ▷ Sample a positive attribute vector
11:      $\mathcal{N} \leftarrow \{\alpha' \mid (\alpha', \mathtt{A}') \in \mathcal{P}, \mathtt{A}' \neq \mathtt{A}\}$ ▷ Get negative class
12:      $\mathcal{L} \leftarrow \mathcal{L} - \log \frac{\exp(f(\alpha, \alpha^+)/\beta)}{\exp(f(\alpha, \alpha^+)/\beta) + \sum_{\alpha^- \in \mathcal{N}} \exp(f(\alpha, \alpha^-)/\beta)}$
13: **end for**
14: $\mathcal{L}_{\text{CompAttr}} \leftarrow \mathcal{L}/|\mathcal{P}|$ ▷ Averaged loss
15: **return** $\mathcal{L}_{\text{CompAttr}}$

C Experiments

C.1 Hyperparameters

To assess the performance differences based on hyperparameters in our proposed ARC, we compared the average gap across 1000 samples for all problems in the validation set, using a baseline model with node size $N = 50$, the number of ARC module layers $N_A = 1$, loss weight $\lambda = 1.0$, temperature $\beta = 0.1$. The results are presented in Figure 6 through Figure 11. Figures 6 and 9 demonstrate performance differences based on the number of ARC module layers. Among the hyperparameters, the change in layer count N_A had the most significant impact on the gap. For the In-distribution setting, stable and superior performance was observed when the number of layers was 3 or higher, whereas for the Zero-shot setting, performance significantly deteriorated with 4 or more layers. This indicates that while increasing model capacity does not reduce performance when all information is provided during training, our loss function can act as a regularizer for unseen combinations up to $N_A = 3$, but cannot prevent performance degradation with further increases in model capacity.

In our contrastive loss, β represents sensitivity to individual logit values, with larger values reducing this sensitivity. As shown in Figures 7 and 10, the standard error tends to decrease as β increases, with a substantial reduction observed at values above 0.14.

However, excessively large β values may lead to performance trade-offs by treating all logit values similarly. In our experiments, we observed that the gap decreases up to $\beta = 0.12$ for the In-distribution setting and $\beta = 0.14$ for the Zero-shot setting, before increasing again. Therefore, appropriate β values should be selected based on the specific environment.

Regarding the loss scaling parameter λ, Figures 8 and 11 show similar results for the In-distribution setting across different values, while for Zero-shot setting, comparable average values were observed except when selecting the very small value of 0.5. We used $N_A = 3, \lambda = 0.12, \lambda = 0.8$, for all experiments, based on the the In-distribution results. Additional performance improvements may be possible through appropriate hyperparameter settings for specific experiments.
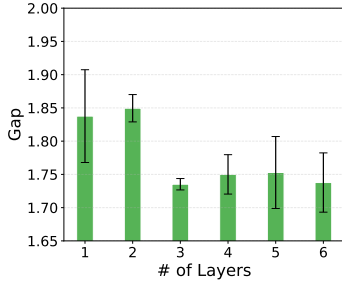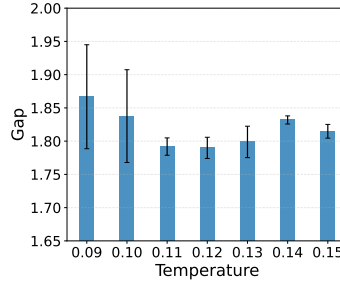

Figure 6: Params. analysis: # of Layers (ID, $N = 50$).

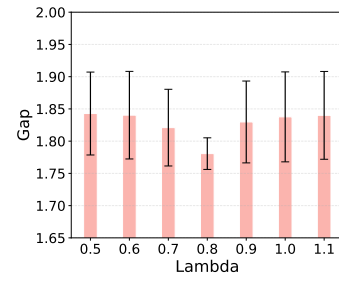Figure 7: Params. analysis: Temperature β (ID, $N = 50$).

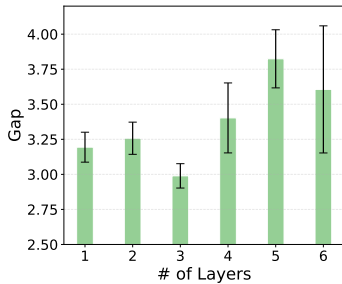Figure 8: Params. analysis: loss weight λ (ID, $N = 50$).

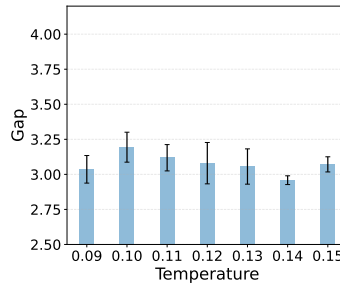
Figure 9: Params. analysis: # of Layers (OOD, $N = 50$).

Figure 10: Params. analysis: Temperature β (OOD, $N = 50$).

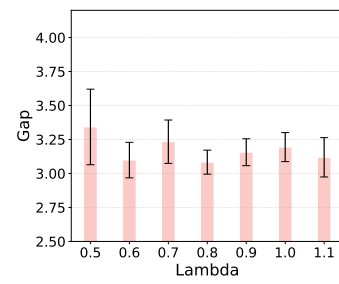Figure 11: Params. analysis: loss weight λ (OOD, $N = 50$).

C.2 Performance on Training Problems

Table 6 shows the total performance in Zero-shot settings, ARC achieves superior average performance, notably 4.078% for $N = 50$ and 6.422% for $N = 100$. It outperforms neural baselines by an even larger margin than in In-distribution. This robustness is particularly evident in challenging VRP variants like OVRPB and OVRPBL. For $N = 50$, ARC significantly surpasses these baselines, achieving 4.731% on OVRPB and 6.090% on OVRPBL, while the baseline performance ranges from 7.121% to 8.351% for OVRPB and from 7.686% to 10.984% for OVRPBL. This zero-shot generalization stems from ARC's explicit learning of attribute compositions, which capture the semantic structure necessary for reasoning about unseen combinations. Unlike other baselines that only implicitly model such structures, our method directly encodes attribute relationships. This demonstrates the strong generalization potential derived from a precise semantic understanding of individual attributes when encountering novel VRP variants.

Table 6: Performance on 1K test Zero-shot instances. Bold and underline denote best and second-best, respectively. Parentheses indicate gap changes from seen performance.

	Solver	$n = 50$	$n = 100$		Solver	$n = 50$	$n = 100$
VRPBTW	MTPOMO	4.276 ± 0.07 (+2.354)	6.851 ± 0.10 (+3.435)	VRPLTW	MTPOMO	3.172 ± 0.02 (+0.315)	4.941 ± 0.04 (+0.512)
	MVMoE	**3.917 ± 0.08** (+2.032)	6.499 ± 0.00 (+3.138)		MVMoE	3.352 ± 0.13 (+0.604)	5.168 ± 0.13 (+0.825)
	RF-TE	4.035 ± 0.05 (+2.501)	**6.240 ± 0.11** (+3.644)		RF-TE	3.765 ± 0.54 (+1.484)	6.230 ± 0.32 (+2.610)
	CaDA	6.067 ± 1.94 (+4.540)	6.659 ± 0.19 (+4.208)		CaDA	2.840 ± 0.10 (+0.519)	4.488 ± 0.15 (+1.034)
	ARC	4.214 ± 0.11 (+2.839)	6.697 ± 0.14 (+4.416)		ARC	**2.710 ± 0.07** (+0.601)	**4.220 ± 0.03** (+0.957)
OVRPB	MTPOMO	7.692 ± 0.26 (+4.691)	13.735 ± 0.31 (+8.391)	OVRPBL	MTPOMO	9.027 ± 0.53 (+5.940)	13.730 ± 0.29 (+8.296)
	MVMoE	7.214 ± 0.50 (+4.508)	12.404 ± 0.08 (+7.613)		MVMoE	7.686 ± 0.46 (+4.909)	13.082 ± 0.22 (+8.216)
	RF-TE	7.121 ± 0.76 (+4.676)	10.500 ± 0.32 (+6.212)		RF-TE	8.564 ± 1.33 (+6.102)	11.991 ± 1.09 (+7.690)
	CaDA	8.351 ± 1.45 (+5.955)	10.224 ± 0.61 (+6.103)		CaDA	10.984 ± 2.08 (+8.587)	14.435 ± 2.45 (+10.328)
	ARC	**4.731 ± 0.28** (+2.500)	**8.877 ± 0.35** (+4.942)		ARC	**6.090 ± 0.19** (+3.854)	**9.694 ± 0.54** (+5.758)
OVRPBLTW	MTPOMO	6.377 ± 1.19 (+5.061)	6.402 ± 0.50 (+3.753)	OVRPBTW	MTPOMO	3.507 ± 0.07 (+2.212)	5.960 ± 0.07 (+3.339)
	MVMoE	5.031 ± 0.76 (+3.742)	7.624 ± 0.71 (+5.009)		MVMoE	**3.180 ± 0.08** (+1.891)	5.685 ± 0.12 (+3.081)
	RF-TE	3.904 ± 0.59 (+2.831)	7.706 ± 0.23 (+5.708)		RF-TE	3.497 ± 0.07 (+2.438)	5.730 ± 0.17 (+3.733)
	CaDA	5.895 ± 1.00 (+4.901)	**5.949 ± 0.13** (+4.125)		CaDA	3.591 ± 0.10 (+2.605)	**5.673 ± 0.10** (+3.843)
	ARC	**3.890 ± 0.09** (+2.985)	6.587 ± 0.11 (+4.946)		ARC	3.749 ± 0.10 (+2.847)	6.334 ± 0.11 (+4.695)
OVRPL	MTPOMO	3.764 ± 0.01 (+0.531)	5.564 ± 0.07 (+0.410)	OVRPLTW	MTPOMO	3.756 ± 0.90 (+2.182)	3.905 ± 0.10 (+0.873)
	MVMoE	**3.472 ± 0.02** (+0.507)	5.105 ± 0.00 (+0.448)		MVMoE	2.823 ± 0.49 (+1.273)	4.403 ± 0.50 (+1.440)
	RF-TE	7.142 ± 3.17 (+4.506)	8.065 ± 1.94 (+3.936)		RF-TE	2.791 ± 0.68 (+1.503)	5.152 ± 0.26 (+2.796)
	CaDA	13.622 ± 4.86 (+10.987)	14.696 ± 5.24 (+10.667)		CaDA	4.962 ± 1.32 (+3.714)	3.072 ± 0.18 (+0.782)
	ARC	4.961 ± 0.40 (+2.451)	**4.997 ± 0.19** (+1.081)		ARC	**1.762 ± 0.21** (+0.678)	**2.656 ± 0.16** (+0.703)
VRPBLTW	MTPOMO	4.867 ± 0.08 (+2.713)	7.587 ± 0.18 (+3.796)	Average	MTPOMO	5.160 ± 0.69	7.630 ± 1.21
	MVMoE	4.846 ± 0.41 (+2.744)	7.749 ± 0.25 (+4.024)		MVMoE	4.613 ± 0.59	7.524 ± 1.06
	RF-TE	4.767 ± 0.49 (+3.013)	8.969 ± 0.58 (+6.019)		RF-TE	5.065 ± 0.67	7.843 ± 0.77
	CaDA	**4.348 ± 0.10** (+2.607)	**7.198 ± 0.08** (+4.385)		CaDA	6.740 ± 1.20	8.044 ± 1.39
	ARC	4.591 ± 0.07 (+2.992)	7.736 ± 0.06 (+5.123)		ARC	**4.078 ± 0.42**	**6.422 ± 0.74**

Table 7 shows the performance of models trained with the Zero-shot setting on the problem variants that were included in training: CVRP, OVRP, VRPL, VRPB, VRPTW, OVRPTW, and VRPBL. Similar to Table 1, ARC achieved the best performance, followed by CaDA and RF-TE. Interestingly, consistent patterns of performance improvement or deterioration compared to the In-distribution setting were observed across problem types, regardless of the baseline model. For VRPTW, OVRP, and OVRPTW, we can observe a general decline in neural solver performance compared to Table 1. This performance decline, despite the increased number of samples per problem type, suggests that learning from multiple problem combinations simultaneously provides greater benefits than simply increasing the sample count for individual problem types. This suggests the effectiveness of the cross-problem approach that utilizes a unified model for learning across problem variants. Analysis of problems where our methodology showed significant improvement compared to other baselines is presented in Appendix D.1.

C.3 Real-World Dataset

To evaluate the scalability of our methodology in real-world settings, we present instance-level performance on the X group from CVRPLib. Table 8 shows the evaluation results for instances with node sizes below 251. Similar to Table 3, ARC demonstrated the best performance, followed by RF-TE and CaDA in order of performance. Additionally, results for instances with node sizes above 500 are presented in Table 9. For these larger instances, RF-TE achieved the best performance with

Table 7: Performance on 1K seen problem test instances of trained with the Zero-shot setting. Bold and underline denote best and second-best, respectively. Parentheses indicate gap changes from seen performance.

	Solver	$n=50$	$n=100$		Solver	$n=50$	$n=100$
CVRP	MTPOMO	1.197 ± 0.01 (-0.210)	1.643 ± 0.03 (-0.406)	VRPTW	MTPOMO	2.529 ± 0.03 (+0.100)	4.198 ± 0.02 (+0.264)
	MVMoE	1.071 ± 0.01 (-0.155)	1.388 ± 0.01 (-0.258)		MVMoE	2.494 ± 0.05 (+0.156)	4.156 ± 0.03 (+0.281)
	RF-TE	1.129 ± 0.06 (-0.110)	1.360 ± 0.05 (-0.200)		RF-TE	2.094 ± 0.07 (+0.161)	3.487 ± 0.07 (+0.295)
	CaDA	1.156 ± 0.02 (-0.103)	1.306 ± 0.01 (-0.208)		CaDA	2.157 ± 0.09 (+0.230)	3.254 ± 0.09 (+0.229)
	ARC	**1.034 ± 0.04** (-0.120)	**1.225 ± 0.02** (-0.204)		ARC	**1.864 ± 0.05** (+0.092)	**3.030 ± 0.06** (+0.190)
OVRP	MTPOMO	3.529 ± 0.04 (+0.316)	5.336 ± 0.07 (+0.234)	VRPL	MTPOMO	1.486 ± 0.01 (-0.232)	2.076 ± 0.02 (-0.411)
	MVMoE	3.204 ± 0.00 (+0.289)	4.883 ± 0.01 (+0.274)		MVMoE	1.354 ± 0.02 (-0.154)	1.806 ± 0.02 (-0.259)
	RF-TE	2.729 ± 0.09 (+0.084)	4.157 ± 0.04 (+0.024)		RF-TE	1.304 ± 0.08 (-0.130)	1.655 ± 0.05 (-0.226)
	CaDA	2.727 ± 0.05 (+0.101)	4.027 ± 0.04 (-0.002)		CaDA	1.357 ± 0.04 (-0.123)	1.604 ± 0.02 (-0.243)
	ARC	**2.504 ± 0.07** (+0.007)	**3.938 ± 0.05** (+0.023)		ARC	**1.192 ± 0.05** (-0.178)	**1.526 ± 0.01** (-0.228)
VRPB	MTPOMO	3.110 ± 0.02 (-0.501)	4.395 ± 0.02 (-0.591)	OVRPTW	MTPOMO	1.774 ± 0.01 (+0.209)	3.394 ± 0.04 (+0.371)
	MVMoE	2.855 ± 0.02 (-0.379)	3.986 ± 0.04 (-0.498)		MVMoE	1.776 ± 0.04 (+0.248)	3.342 ± 0.03 (+0.398)
	RF-TE	2.794 ± 0.10 (-0.191)	3.733 ± 0.05 (-0.266)		RF-TE	1.447 ± 0.06 (+0.162)	2.663 ± 0.06 (+0.309)
	CaDA	2.829 ± 0.07 (-0.144)	3.645 ± 0.01 (-0.303)		CaDA	1.485 ± 0.08 (+0.245)	2.458 ± 0.08 (+0.170)
	ARC	**2.556 ± 0.08** (-0.284)	**3.538 ± 0.03** (-0.295)		ARC	**1.215 ± 0.03** (+0.128)	**2.198 ± 0.04** (+0.230)
VRPBL	MTPOMO	4.236 ± 0.03 (-0.445)	5.692 ± 0.03 (-0.612)	Average	MTPOMO	2.551 ± 0.43	3.819 ± 0.58
	MVMoE	3.955 ± 0.02 (-0.328)	5.248 ± 0.05 (-0.470)		MVMoE	2.387 ± 0.39	3.544 ± 0.56
	RF-TE	3.510 ± 0.13 (-0.206)	4.716 ± 0.05 (-0.304)		RF-TE	2.144 ± 0.34	3.110 ± 0.48
	CaDa	3.563 ± 0.04 (-0.132)	4.585 ± 0.03 (-0.368)		CaDa	2.182 ± 0.34	2.983 ± 0.47
	ARC	**3.185 ± 0.12** (-0.394)	**4.435 ± 0.06** (-0.421)		ARC	**1.936 ± 0.31**	**2.841 ± 0.46**

an average gap of 12.319%, compared to our method's 12.506%. This suggests that while our method generalizes well to distributions different from the training set, it does not perform as exceptionally on very large-sized problems. This suggests that further research on improving scalability to larger problem sizes could be beneficial.

Table 8: Performance on CVRPLib instances ($N \le 251$).

Set-X	MTPOMO		MVMoE		RF-TE		CaDA		ARC	
Instance	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
X-n101-k25	29399	6.553%	29076	5.382%	29035	5.234%	29185	5.777%	28927	4.842%
X-n106-k14	28029	6.323%	27443	4.101%	27150	2.989%	26952	2.238%	26852	1.859%
X-n110-k13	15100	0.862%	15327	2.378%	15314	2.291%	15262	1.944%	15309	2.258%
X-n115-k10	13412	5.217%	13475	5.711%	13338	4.636%	13169	3.311%	13458	5.578%
X-n120-k6	14051	5.393%	13782	3.375%	13765	3.248%	13735	3.023%	13659	2.453%
X-n125-k30	59015	6.259%	58430	5.205%	58522	5.371%	57405	3.360%	57936	4.316%
X-n129-k18	30176	4.271%	29334	1.361%	29598	2.274%	29397	1.579%	29536	2.059%
X-n134-k13	11707	7.246%	11462	5.002%	11585	6.129%	11512	5.460%	11605	6.312%
X-n139-k10	14058	3.444%	14099	3.745%	13812	1.634%	13877	2.112%	13962	2.737%
X-n143-k7	16626	5.898%	16349	4.134%	16257	3.548%	16195	3.153%	16185	3.089%
X-n148-k46	46648	7.365%	45857	5.545%	45036	3.655%	45761	5.324%	45243	4.131%
X-n153-k22	23514	10.811%	23649	11.447%	23478	10.641%	23154	9.114%	23299	9.797%
X-n157-k13	17886	5.985%	17493	3.656%	17339	2.744%	17344	2.773%	17230	2.098%
X-n162-k11	14486	2.461%	14705	4.010%	14664	3.720%	14814	4.781%	14642	3.565%
X-n167-k10	21662	5.375%	21503	4.602%	21412	4.159%	21437	4.281%	21226	3.254%
X-n172-k51	48560	6.475%	47883	4.990%	48118	5.506%	48181	5.644%	48022	5.295%
X-n176-k26	51989	8.736%	52117	9.004%	51400	7.504%	52698	10.219%	52400	9.596%
X-n181-k23	26572	3.923%	26417	3.317%	26097	2.065%	26099	2.073%	26249	2.659%
X-n186-k15	25236	4.519%	25151	4.166%	25140	4.121%	25461	5.450%	25277	4.688%
X-n190-k8	18222	7.314%	18988	11.826%	17892	5.371%	18470	8.775%	17877	5.283%
X-n195-k51	48829	10.410%	47201	6.729%	47390	7.157%	46726	5.655%	46649	5.481%
X-n200-k36	62050	5.927%	61720	5.364%	61199	4.474%	61198	4.473%	61330	4.698%
X-n204-k19	20643	5.510%	20584	5.208%	20608	5.331%	20497	4.764%	20631	5.449%
X-n209-k16	32298	5.356%	32358	5.552%	31876	3.980%	32092	4.684%	32170	4.939%
X-n214-k11	11699	7.765%	11597	6.826%	11670	7.498%	11812	8.806%	11713	7.894%
X-n219-k73	122070	3.805%	124451	5.830%	120348	2.341%	120464	2.440%	120158	2.180%
X-n223-k34	43123	6.642%	42695	5.584%	42251	4.486%	42359	4.753%	42337	4.699%
X-n228-k23	28233	9.677%	28171	9.436%	28798	11.872%	27988	8.725%	28098	9.152%
X-n233-k16	20644	7.353%	20656	7.415%	20758	7.946%	20638	7.322%	20739	7.847%
X-n237-k14	30066	11.183%	29778	10.118%	29595	9.441%	30451	12.606%	29759	10.047%
X-n242-k48	88666	7.148%	87281	5.474%	85704	3.569%	85780	3.660%	85952	3.868%
X-n247-k50	41610	11.633%	41345	10.922%	40639	9.028%	41037	10.096%	40817	9.505%
X-n251-k28	41206	6.519%	41347	6.884%	40399	4.433%	40663	5.116%	40466	4.607%
Average Gap		6.465%		5.888%		5.103%		5.257%		5.037%

Table 9: Performance on CVRPLib instances ($N \geq 500$).

Set-X	MTPOMO		MVMoE		RF-TE		CaDA		ARC	
Instance	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap	Obj.	Gap
X-n502-k39	75858	9.580%	77037	11.283%	71836	3.770%	72427	4.624%	72357	4.523%
X-n513-k21	34192	41.283%	32695	35.098%	28566	18.036%	30037	24.115%	29084	20.177%
X-n524-k153	176706	14.304%	171622	11.015%	174075	12.602%	171656	11.037%	168443	8.959%
X-n536-k96	109781	15.747%	106205	11.976%	103337	8.952%	102768	8.352%	102899	8.491%
X-n548-k50	110634	27.606%	104455	20.479%	100914	16.394%	102813	18.585%	101488	17.057%
X-n561-k42	55564	30.075%	53385	24.974%	49455	15.774%	50410	18.009%	49376	15.589%
X-n573-k30	60460	19.314%	61611	21.585%	55937	10.388%	56622	11.740%	54785	8.115%
X-n586-k159	226529	19.028%	213299	12.076%	205770	8.120%	205385	7.918%	204824	7.623%
X-n599-k92	130376	20.217%	126678	16.807%	116819	7.716%	117727	8.553%	117498	8.342%
X-n613-k62	78323	31.558%	73687	23.771%	67347	13.122%	68696	15.388%	68850	15.646%
X-n627-k43	77282	24.320%	70710	13.748%	67339	8.325%	68838	10.736%	68295	9.863%
X-n641-k35	83223	30.681%	72080	13.184%	70687	10.996%	73329	15.145%	71792	12.732%
X-n655-k131	121032	13.347%	119388	11.807%	112087	4.970%	110761	3.728%	110721	3.691%
X-n670-k130	182652	24.820%	166856	14.026%	169056	15.529%	165711	13.243%	162787	11.245%
X-n685-k75	93216	36.670%	82525	20.996%	77687	13.902%	78145	14.574%	78560	15.182%
X-n701-k44	92855	13.344%	90220	10.128%	90970	11.043%	92254	12.611%	91299	11.445%
X-n716-k35	59066	36.181%	52582	21.232%	49709	14.608%	51313	18.306%	49981	15.235%
X-n733-k159	175228	28.667%	156453	14.881%	148786	9.251%	148357	8.936%	147100	8.013%
X-n749-k98	102540	32.705%	92308	19.463%	85048	10.067%	85634	10.826%	85627	10.817%
X-n766-k71	133109	16.337%	129647	13.311%	130052	13.665%	128140	11.994%	129370	13.069%
X-n783-k48	107925	49.097%	96175	32.864%	83165	14.891%	84805	17.157%	83808	15.779%
X-n801-k40	92027	25.530%	87149	18.876%	86024	17.341%	89329	21.849%	87830	19.805%
X-n819-k171	192568	21.785%	178857	13.114%	174609	10.427%	173461	9.701%	172161	8.879%
X-n837-k142	230660	19.058%	230022	18.729%	208252	7.492%	208225	7.478%	209050	7.904%
X-n856-k95	118219	32.883%	105661	18.767%	98393	10.597%	100057	12.468%	100648	13.132%
X-n876-k59	114340	15.147%	114169	14.975%	107229	7.986%	110332	11.111%	108640	9.407%
X-n895-k37	106400	97.549%	70002	29.970%	64525	19.801%	67511	25.345%	65160	20.980%
X-n916-k207	388519	18.027%	373542	13.477%	352732	7.155%	353009	7.239%	352056	6.950%
X-n936-k151	200710	51.234%	161068	21.364%	163073	22.875%	154841	16.672%	157210	18.457%
X-n957-k87	126800	48.365%	123712	44.752%	102964	20.475%	105200	23.091%	103485	21.085%
X-n979-k58	139389	17.157%	131894	10.858%	129770	9.072%	133144	11.908%	131469	10.500%
X-n1001-k43	133680	84.756%	89126	23.179%	85998	18.856%	89231	23.324%	87914	21.504%
Average Gap	30.199%		18.836%		12.319%		13.618%		12.506%	

D   Analysis

D.1   In-Distribution Performance: Visual Comparison of Gap Ratio

Figure 12 illustrates the gap ratios between our method ARC and comparable baselines RF-TE and CaDA, based on the performance gaps reported in Table 1. The results demonstrate consistently superior performance across all problems that simultaneously incorporate both O and TW attributes (i.e., OVRPTW, OVRPLTW, OVRPBTW, OVRPBLTW). As noted in Appendix C.2, problems such as VRPTW, OVRP, and OVRPTW showed performance improvements by leveraging information from problem instances with different combinations. This suggests that ARC achieves substantial performance gains on problems incorporating these attributes by effectively leveraging O and TW attribute information from various problem combinations through our compositional learning methodology.
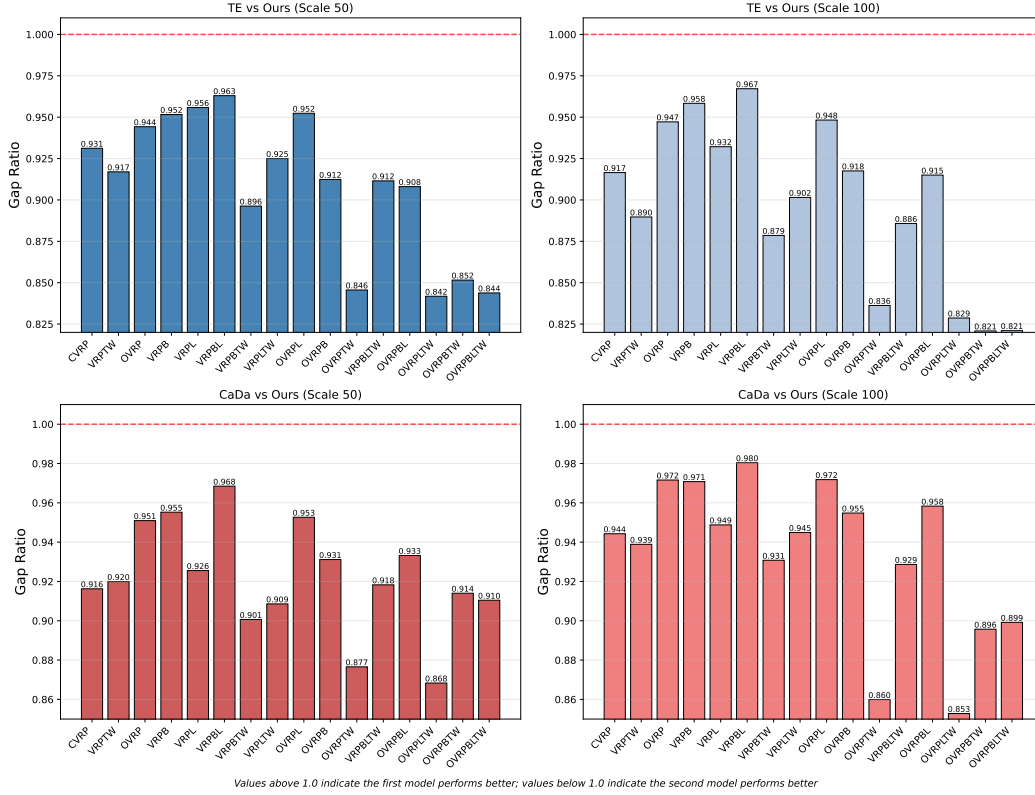

Figure 12: Visualizing Gap ratio
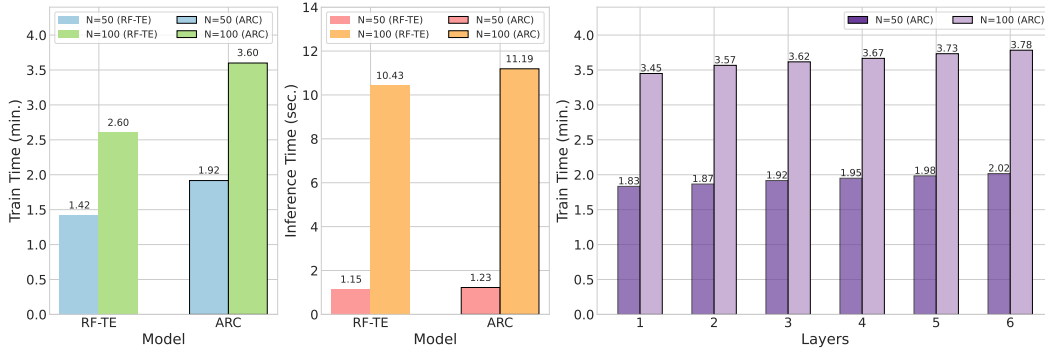
D.2 Train/Test Efficiency

Figure 13: Computational Efficiency Comparison. Training time per epoch (left), inference time (middle), and training time by number of layers (right).

We compared the time required per epoch during training (100,000 instances) and evaluation (1,000 instances) for our proposed model with node sizes $N = 50$ and $N = 100$ illustrated in Figure 13. All experiments were conducted on a single NVIDIA Tesla A40 GPU and two CPU cores of AMD EPYC 7413 24-Core Processor for both training and testing. For traditional solvers, we allocated 16 CPU cores As shown in the left panel, while our method requires longer training time than RF-TE, training for $N = 100$ can be completed within a single day. Furthermore, the middle panel demonstrates that the computation of $\mathcal{L}_{\text{CompAttr}}$ is only required during training, resulting in significantly faster inference times. The right panel shows differences in training speed based on the number of ARC module layers. Due to the time-intensive nature of the sequential solution selection in the decoding step, the difference between layer counts of 1 and 6 are only 0.19 and 0.33 minutes for $N = 50$ and $N = 100$, respectively.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist",**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and Introduction (Section 1) state the importance of the compositional learning of the attributes and summarize the main contributions, a novel cross-problem learning solver for VRPs decomposing embeddings into intrinsic and contextual embeddings to disentangle attribute representations. They also claim superior performance and generalization capabilities.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: While a dedicated "Limitations" section is not present, the paper discusses some limitations. For instance, Appendix C.3 (Real-World Dataset) notes that the model "it does not perform as exceptionally on very large-sized problems".

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper primarily presents an empirical study introducing a novel architecture and learning methodology. It does not focus on presenting new theoretical results that would require formal proofs. The methodological foundations, including Properties (P1) and (P2) that motivate the design, are described in Section 4, Methods, but these are not presented as formal theorems.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper disclose detailed experiment settings in Section 5 and Pseudocode in Appendix. The model architecture is also described well in Section 4.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper provides a URL: "https://anonymous.4open.science/r/compositional-co-30FE/". Details on data generation are notes that we "follow RouteFinder's data generation and training protocols" in Section 5.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Their experimental settings also described in Section 5 and Appendix. Section 5 describes the number of training, test instances, including graph size. Due to spce limitations, details are notes that we "follow RouteFinder's data generation and training protocols" in Section 5. Appendix C is referenced for further details on hyperparameters and related studies. The experimental settings (In-distribution, Zero-shot Generalization, Few-shot Adaptation, Real-world Benchmark) are clearly defined in Section 5.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Tables 1, 2, 4, 6, and 7 report results with what appear to be standard deviations. The paper also states, "For neural methods, results are averaged over 3 runs with different random seeds".

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We descrbied in Appendix D.2 as "All experiments were conducted on a single NVIDIA Tesla A40 GPU and two CPU cores of AMD EPYC 7413 24-Core Processor for both training and testing. For traditional solvers, we allocated 16 CPU cores". Execution time is noted in Appendix.

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The research focuses on algorithmic development for Vehicle Routing Problems, a subfield of combinatorial optimization. Based on the paper's content, there is no indication of ethical violations related to data privacy, bias, or harm as outlined in the NeurIPS Code of Ethics.

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: The paper does not include a dedicated section discussing broader societal impacts. While the research could lead to positive impacts by improving efficiency in logistics and transportation (potentially reducing costs and emissions), these are not explicitly detailed, nor are potential negative societal impacts discussed. The primary focus is on the technical contributions of the ARC model.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: The research involves models for solving Vehicle Routing Problems. These models and the data do not inherently pose a high risk for misuse in the sense typically associated with large language models or image generation models. Therefore, specific safeguards beyond standard academic open-sourcing practices are not directly applicable.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites numerous existing works and baselines, such as PyVRP, MTPOMO, MVMoE, RouteFinder, and CaDA. It also uses the CVRPLib benchmark, which is cited with a URL footnote 3. The use of a unified codebase from RouteFinder for some baselines is mentioned in Section 5.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The primary new asset introduced is the ARC model and its implementation. The paper provides a detailed description of the model architecture, the learning methodology, and the experimental setup. A link to an anonymized code repository is provided, suggesting that documentation would be available with the code release.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The research described in the paper does not involve crowdsourcing or experiments with human subjects. It is focused on algorithmic development and evaluation using synthetic and benchmark datasets.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The research does not involve human subjects; therefore, IRB approval or discussion of participant risks is not applicable.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core methodology of the research focuses on a novel neural network architecture and compositional learning for Vehicle Routing Problems. There is no mention or indication in the paper that Large Language Models (LLMs) are an important, original, or non-standard component of these core research methods.