LPACR: LOCAL POLICY WITH AUGMENTED CONTEXTUAL REPRESENTATIONS FOR ENHANCING OOD GENERALIZATION IN VRPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Practical vehicle routing problem (VRP) instances exhibit significant variations in problem scale and intrinsic distribution, posing fundamental challenges to cross-scale and cross-distribution generalization capabilities of neural solvers. Although local policies are effective for cross-scale generalization, their restricted receptive fields lack global awareness. Meanwhile, we reveal that homogeneous training samples fail to cover diverse local topological structures that are crucial for cross-distribution generalization. In terms of learning paradigms, supervised learning (SL) suffers from objective misalignment, whereas reinforcement learning (RL) faces convergence instability. To overcome these limitations, we propose LPACR (Local Policy with Augmented Contextual Representations). We introduce a grid-based distribution compression method that leverages the spatial locality of VRP to encode grid-wise statistical information, thereby augmenting the contextual representation of the local policy. To enhance cross-distribution adaptability, we employ a multi-distribution training strategy that can encompass heterogeneous local topological structures. Furthermore, we train LPACR through a two-stage learning paradigm that synergizes SL and RL. Extensive experimental results on synthetic datasets and real-world benchmarks with diverse scales and distributions demonstrate LPACR's robust generalization capabilities.

1 INTRODUCTION

Neural combinatorial optimization (NCO) approaches leverage deep neural networks to solve combinatorial optimization problems (Bengio et al., 2021), such as the vehicle routing problem (VRP) that involves finding the shortest route while satisfying problem-specific constraints (Toth & Vigo, 2002). In contrast to traditional Optimization Research (OR) approaches that rely on hand-crafted heuristics requiring extensive expert knowledge (Helsgaun, 2017), neural approaches adopt a fully data-driven, end-to-end learning framework (Kool et al., 2019). Recent advances have demonstrated neural solvers' impressive in-distribution performance, yielding high-quality solutions with low inference latency (Luo et al., 2025). Nevertheless, robust out-of-distribution (OOD) generalization capabilities of neural VRP solvers remain a fundamental challenge (Zhou et al., 2023).

Due to computational constraints, typical NCO solvers are trained on small-scale, spatially homogeneous datasets (Kwon et al., 2020; Luo et al., 2023), such as the 100-node traveling salesman problem (TSP) (Hoffman et al., 2013) instances sampled from a uniform distribution. Empirical results show that as distributional shifts intensify, their solution quality deteriorates significantly, highlighting the limited OOD generalization capabilities (Joshi et al., 2022). Given that practical VRP instances exhibit significant variations in problem scale and intrinsic distribution, bridging this generalization gap is imperative for real-world adoption (Goh et al., 2025; Chen et al., 2025b).

Regarding cross-scale generalization, the quadratic time complexity inherent in the self-attention mechanism of the Transformer (Vaswani et al., 2017) poses a substantial computational burden on large-scale instances. Moreover, numerous distant nodes exert cumulative influence on the attention weights of high-potential local candidate nodes (Fang et al., 2024; Chen et al., 2025c), disrupting the patterns learned from small-scale instances and degrading solution quality. Although local policy architectures (Chen et al., 2026; Zhou et al., 2026) can address scalability by restricting state and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

action spaces to the local neighbors of the current position, their limited receptive fields sacrifice global contextual awareness, frequently yielding suboptimal decisions (Xiao et al., 2025b). Beyond scalability, our investigation reveals that the diversity of local topological structures is crucial for cross-distribution generalization, whereas a single uniform distribution yields highly homogeneous patterns. In addition, existing methods require additional efforts to train models on instances of diverse distributions (Bi et al., 2022; Jiang et al., 2022). Furthermore, due to the inherent misalignment of optimization objectives, supervised learning (SL) fails to capture sequential dependencies and compromises generalization performance. Alternatively, reinforcement learning (RL) typically requires substantial efforts for convergence.

To overcome these limitations, we propose LPACR, a constructive neural solver that addresses both cross-scale and cross-distribution generalization challenges simultaneously. Specifically, we introduce a grid-based distribution compression (GDC) method which exploits VRP's spatial locality to encode grid-wise statistical features, effectively aggregating global distributional information into compact representations. The generated grid tokens are concatenated with node tokens to augment the contextual representations of the local policy by incorporating global awareness. To enhance cross-distribution adaptability, we employ a simple yet effective multi-distribution training (MDT) strategy comprising uniform, cluster, and mixed distributions to systematically encompass heterogeneous local patterns in training samples, without altering the existing training pipeline. Furthermore, LPACR is trained via a two-stage learning algorithm that synergizes SL and RL, where SL facilitates rapid initialization, and RL fine-tunes the policy to align with task-specific optimization objectives.

We evaluate LPACR on two representative VRP variants, TSP and CVRP. For each variant, we use 42 synthetic datasets covering diverse problem scales and node distributions. In addition, we adopt 9 real-world benchmarks including TSPLib (Reinelt, 1991), VLSI (Rohe, 2013), NationalTSP (Cook, 2009), and CVRPLib (Christofides, 1979; Golden et al., 1998; Li et al., 2005; Uchoa et al., 2017; Arnold et al., 2019; Queiroga et al., 2026). On synthetic datasets, LPACR consistently surpasses neural baselines, and it even outperforms the benchmark HGS (Vidal, 2022) runs under limited time budgets on large-scale CVRP instances. Furthermore, LPACR maintains robust performance on real-world instances whose problem scales are up to 7447× larger than those of its training samples and whose distributions are unseen during training, exhibiting superior OOD generalization capabilities.

Our key contributions are summarized as follows: 1) We propose LPACR, a highly scalable hybrid architecture that leverages the GDC method to impart global awareness to the local policy. 2) We identify the criticality of local topological diversity for cross-distribution generalization and employ the MDT strategy to encompass heterogeneous patterns in training samples. 3) We synergize SL initialization with RL refinement to balance training efficiency and objective alignment, yielding substantial improvements on CVRP instances. 4) Extensive evaluations across synthetic and real-world TSP/CVRP instances verify LPACR's generality and robust OOD generalization capabilities.

2 Related Work

While machine learning relies on the independent and identically distributed assumption, violating this premise poses significant challenges for OOD generalization (Liu et al., 2021; Ye et al., 2021). In the context of NCO, these challenges primarily manifest as cross-scale and cross-distribution generalization (Zhou et al., 2023). The former refers to solving instances with problem scale differing from training data, while the latter involves instances generated from unseen distributions.

Cross-scale Generalization. Representation learning plays a vital role in addressing this issue. Advanced global policies adopt heavy decoder architectures to enhance cross-scale generalization performance (Luo et al., 2023; Drakulic et al., 2023; Luo et al., 2025). Nevertheless, they treat all nodes indiscriminately, leading to a severe computational burden and performance degradation on large-scale instances. By contrast, local policies utilize the locality of VRP solutions by restricting contextual representations to local candidate nodes, significantly enhancing their scalability (Fang et al., 2024; Chen et al., 2026; Zhou et al., 2026). Yet, their limited receptive fields sacrifice global awareness, frequently yielding suboptimal decisions. Hybrid policies strike a balance between these two architectures, though how to effectively integrate global awareness into local decisions remains an open problem (Gao et al., 2024; Xiao et al., 2025a;b). LPACR extends this research line by compressing grid-wise statistical features into compact grid tokens, which are then concatenated with candidate node tokens to augment contextual representations and improve prediction quality.

Several prior works adopt grid/image-based encoding schemes related to LPACR. For example, GELD (Xiao et al., 2025a) partitions nodes into grids and uses a region-average linear attention mechanism to fuse static global information into node embeddings. Unlike GELD, we employ the GDC method at each decoding step to compress dynamic global information into independent grid tokens, which are fed to the decoder alongside node tokens. Our design better preserves dynamic global information, allowing the decoder to selectively attend to global context. GIMF (Chen et al., 2025a) and VAGPO (Liu et al., 2025) construct images whose pixels indicate micro-level node existence. By contrast, our grid-wise statistical features are more informative and encapsulate macro-level characteristics. H-TSP (Pan et al., 2023) adopts a hierarchical framework, where a CNN-based high-level policy predicts 2D coordinates to guide sub-problems' construction. Nevertheless, its low-level policy is confined to each sub-problem, failing to exploit global contextual information.

Cross-distribution Generalization. Existing approaches require additional efforts in training to handle multiple distributions. For example, AMDKD (Bi et al., 2022) trains separate teacher models for each distribution and then aggregates their knowledge through knowledge distillation. Group-DRO (Jiang et al., 2022) demands continual evaluations to update importance weights corresponding to different groups during training. HAC (Zhang et al., 2022) adaptively generates instances across hardness levels for curriculum learning. EL-DRL (Jiang et al., 2023) learns multiple sub-policies and leverages bootstrap with random initialization to enhance their diversity to cope with various distributions. OmniVRP (Zhou et al., 2023) incorporates a hierarchical scheduler for selecting tasks of different distributions. Although these strategies yield performance improvements, the underlying mechanism for enhancing cross-distribution generalization remains underexplained.

To the best of our knowledge, we are the first to highlight the importance of the diversity in local topological structures for tackling the cross-distribution generalization challenge. These structures constitute the key difference between instances sampled from different distributions, and we present a quantitative method to characterize them. Motivated by our investigation results, we propose the MDT strategy to systematically encompass heterogeneous patterns in training samples. This strategy does not alter the subsequent training pipeline, yielding substantial performance improvements.

Learning Paradigms. Current NCO solvers are typically trained using either supervised learning or reinforcement learning. SL methods (Luo et al., 2023; Drakulic et al., 2023) leverage high-quality solutions generated by classical solvers as supervision signals. Despite their training efficiency, SL methods suffer from a fundamental objective misalignment: they focus on mimicking stepwise choices along the labeled solutions, whereas the objective of VRP is to optimize the entire decision sequence. Self-improvement learning (Xiao et al., 2025a) is an emerging paradigm that utilizes the neural solver to refine solutions as pseudo data for training, thereby eliminating reliance on solvers. For example, SIGD (Pirnay & Grimm, 2024) and DGL (Xiao et al., 2025b) exploit beam search to generate multiple trajectories and retain the best one, whereas SIL (Luo et al., 2025) reconstructs partial solutions and adopts positive modifications. However, existing implementations primarily adopt the stepwise supervision scheme, and thus the misalignment pitfall persists. Conversely, RL methods (Huang et al., 2025; Meng et al., 2025) directly optimize the whole trajectories. Variants of REINFORCE (Williams, 1992) augmented with diverse baselines are widely adopted to stabilize convergence (Kool et al., 2019; Kwon et al., 2020). Nevertheless, RL remains hindered by sparse rewards and prohibitive memory overhead, posing significant challenges for training from scratch.

In contrast to these methods, we employ a two-stage learning paradigm, in which SL facilitates rapid initialization, and RL fine-tunes the policy to align with task-specific optimization objectives.

3 Local Policy with Augmented Contextual Representations

A schematic overview of LPACR is presented in Figure 1. In this section, we take TSP as the primary example to introduce its architecture, while CVRP-specific details are provided in Appendix C.

Local Policy Backbone. LEHD (Luo et al., 2023) suggests that the heavy decoder architecture, which recomputes dynamic embeddings at each decoding step, is crucial for improving solution quality. However, the quadratic time complexity of the self-attention mechanism (Vaswani et al., 2017), compounded over N decoding steps, yields an overall $O(N^3)$ time complexity, severely hindering its scalability on large-scale instances. TTPL (Chen et al., 2026) extends LEHD by restricting

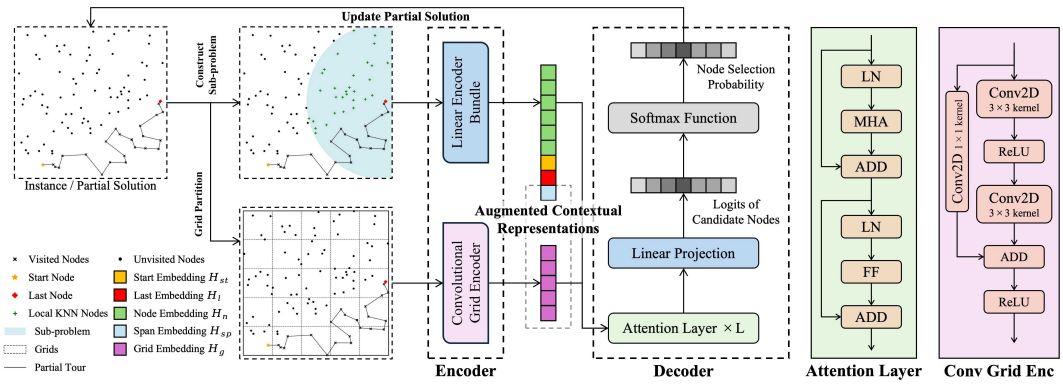


Figure 1: Schematic overview of the LPACR architecture.

its stepwise context to k-NN candidate nodes and applying projection to normalize node coordinates, addressing the scalability issue and significantly improving cross-scale generalization performance.

We adopt these well-established architectures as the backbone of LPACR, and introduce several modifications. 1) The start node provides essential route history (Kool et al., 2019), yet it frequently falls outside the local neighborhood. TTPL directly clips its normalized coordinates to $[0, 1]^2$ in the sub-problem projection process, potentially causing distortion of important semantic information in the context. In contrast to TTPL, we directly project its original coordinates into the start embedding H_{st} through a learnable linear projection, better preserving its semantics within the global context. 2) We employ a span token $X_{\text{sp}} = [x_{\min}, y_{\min}, L]$, where $L = \max(x_{\max} - x_{\min}, y_{\max} - y_{\min})$. A learnable linear projection maps this token to the span embedding H_{sp}, explicitly measuring the spatial information of the sub-problem. 3) We omit the attention layers in the encoder to improve computational efficiency, leaving only a bundle of learnable linear projections for handling different nodes. 4) In the decoder, we adopt SwiGLU (Shazeer, 2020) for the feed-forward sub-layer and RMSNorm-based Pre-Layer Normalization (Zhang & Sennrich, 2019), aligning with modern Transformer architectures such as Qwen3 (Yang et al., 2025) to enhance model expressiveness.

Specifically, at each decoding step, we take the last visited node together with its k-NN nodes as a sub-problem, apply isotropic min-max normalization to project their original coordinates, and linearly map them into embeddings H_{l} and H_{n}. The contextual role is then explicitly encoded into H_{l} via a learnable linear projection. Subsequently, different embeddings are concatenated to form the input token sequence $H^{(0)} = [H_{\text{st}}, H_{\text{n}}, H_{\text{l}}, H_{\text{sp}}, H_{\text{g}}]$ for the decoder. By integrating explicit positional priors H_{sp} and grid-wise statistics H_{g} to provide global awareness, LPACR significantly enriches the contextual representations of the local policy when compared with TTPL.

The decoder processes the token sequence through L stacked attention layers, each comprising a multi-head attention (MHA) sub-layer and a feed-forward (FF) sub-layer. The update of the i-th token embedding at layer l, denoted $h_i^{(l)}$, is computed as Equation 1 shows. Finally, the embeddings of local candidate nodes $H_{\text{n}}^{(L)}$ are mapped into scores through a SwiGLU, and converted into a probability distribution using the softmax function, driving the autoregressive construction process.

$$\bar{H}^{(l-1)} = \text{LN}(H^{(l-1)}),\ \hat{h}_i^{(l)} = h_i^{(l-1)} + \text{MHA}(\bar{h}_i^{(l-1)}, \bar{H}^{(l-1)}),\ \bar{H}^{(l)} = \text{LN}(\hat{H}^{(l)}),\ h_i^{(l)} = \hat{h}_i^{(l)} + \text{FF}(\bar{h}_i^{(l)}) \tag{1}$$

Grid-based Distribution Compression Although local policy-based architectures can effectively alleviate the scalability bottleneck, their limited receptive fields lack necessary global awareness. Bridging this gap requires aggregating global distributional information without incurring prohibitive overhead. We observe that nodes' contributions to local decisions exhibit strong spatial heterogeneity: proximal nodes require token-level precision to resolve subtle structural variations, whereas distant nodes primarily offer macroscopic distributional cues, where individual granularity is secondary. However, conventional neural solvers process all nodes via a unified graph representation, failing to exploit such heterogeneous characteristics (Luo et al., 2023; Gao et al., 2024).

To address this issue, we introduce the grid-based distribution compression method, leveraging a structured grid layout to aggregate distributional information. Specifically, we partition all nodes

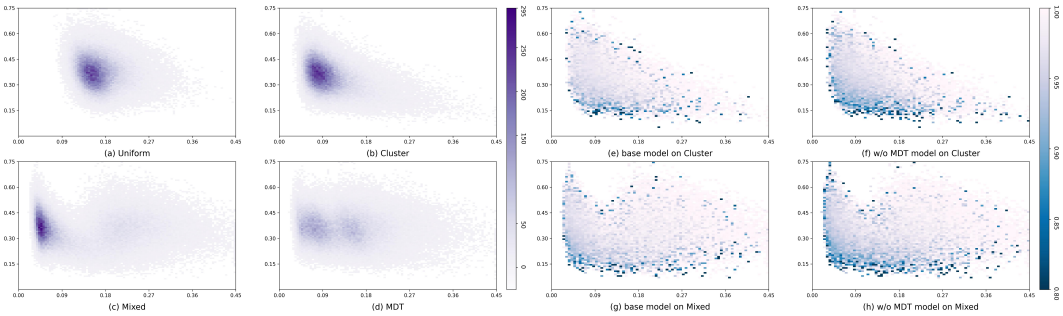


Figure 2: Visualization of local topological structures across (a) Uniform, (b) Cluster, (c) Mixed distributions, and (d) the MDT strategy; (e-h) Prediction consistency with ground-truth decisions for two LPACR variants on datasets sampled from cluster and mixed distributions. The x-axis represents the mean kNN distance, and the y-axis denotes the coefficient of variation of these distances.

into G equal-sized grids based on their coordinates. We empirically set $G = 6 \times 6$, which strikes a balance between spatial granularity and computational efficiency. Subsequently, nodes within each grid are aggregated into statistical features as follows: 1) **Node proportion**. It represents the ratio of nodes within the grid to the total number of nodes, reflecting the relative local density. 2) **Spatial distribution**. These metrics are defined as the centroid coordinates and standard deviations of node coordinates to characterize the spatial layout within each grid. For empty grids, these values are set to the grid's center coordinates and zeros, respectively. 3) **Visit ratio**. It represents the fraction of visited nodes within the grid, indicating local traversal progress. For empty grids, this ratio is initialized to one. 4) **Binary flags**. We adopt two binary indicators specifying whether the start node and the last visited node lie within the grid. 5) **Demand statistics (CVRP only)**. These metrics are defined as the mean and standard deviation of demands within the grid, and are set to zero for empty grids. These statistical features are updated incrementally after each decoding step.

GDC features four key advantages: 1) **Spatial inductive bias**. The grid layout naturally preserves Euclidean proximity, providing a strong prior for learning spatial distributional patterns. 2) **Noise robustness**. Aggregating node statistics within each grid induces local invariance, rendering grid features stable against positional perturbation of nodes. 3) **Vision backbone compatibility**. The structured layout seamlessly integrates vision backbones (He et al., 2016; Gu et al., 2018), leveraging their proficiency in hierarchical feature extraction and facilitating downstream decision-making. 4) **Computational efficiency**. The complexity of subsequent global contextual representations is effectively decoupled from the problem scale N, with $G \ll N$ for large-scale instances.

Convolutional Grid Encoder This module aims to capture spatial dependencies across grid-wise statistical features X_{g}, and encode them into grid embeddings H_{g}. To maintain computational efficiency, we adopt a lightweight yet classic convolutional neural network architecture consisting of two successive 3×3 convolutional layers with unit stride and padding. Each convolution is followed by a ReLU activation function $\sigma(\cdot)$, and we employ a residual connection with a 1×1 convolution to align the dimensions of the hidden space. This design is known for its capability to preserve local invariance while capturing global dependencies. Finally, a learnable modality embedding T_{g} is added to explicitly differentiate grid embeddings from other embeddings in the hidden space (Tan & Bansal, 2019). The encoding process of the convolutional grid encoder is given by Equation 2.

$$H_{\mathrm{g}} = \sigma[Conv_{3\times3}(H_{\mathrm{g}}{}') + Conv_{1\times1}(X_{\mathrm{g}})] + T_{\mathrm{g}}, \quad H_{\mathrm{g}}{}' = \sigma[Conv_{3\times3}(X_{\mathrm{g}})] \tag{2}$$

4 Enhance Cross-distribution Generalization

Local Topological Structures VRP instances sampled from different distributions exhibit distinct patterns in their local topological structures, exerting a significant influence on neural prediction especially for local policies. However, existing works lack investigations into these crucial patterns.

We formalize local topological structures using two quantitative metrics: the mean distance from the kNN nodes to the anchor node, and the coefficient of variation of these distances. As visualized in Figure 2, we analyze structural variations across distribution types and evaluate their influence on

neural prediction quality, measured via stepwise consistency with ground-truth decisions. Details are provided in Appendix F due to space constraints. In brief, our investigation indicates that a single distribution fails to capture a full spectrum of local patterns. Consequently, neural solvers trained on homogeneous data degrade significantly when encountering local structures that deviate from dominant patterns in their training data, thereby hindering cross-distribution generalization.

Multi-distribution Training To systematically encompass heterogeneous patterns in training samples, we present a simple yet effective multi-distribution training strategy. MDT constructs a heterogeneous training pool by sampling from uniform, cluster, and mixed distributions, with an empirical sampling ratio of 3:3:4 for these distributions. This strategy only adjusts the proportions of training distributions without altering the existing training pipeline. Subfigure 2d shows that our strategy broadens the coverage of local patterns and alleviates their distributional imbalance. Moreover, subfigures 2e and 2g show that LPACR trained using MDT exhibits improvements in prediction consistency, microscopically elucidating the underlying reasons for the enhanced solution quality. Considering that most recent methods still adopt a single uniform distribution for training, our work will inspire future research to enhance cross-distribution generalization capabilities efficiently.

5 Two-stage Learning paradigm

Supervised learning and reinforcement learning constitute the dominant paradigms for NCO solvers, yet each exhibits distinct inherent limitations as discussed in Section 2. Therefore, we employ a two-stage learning paradigm that synergizes SL and RL to leverage their complementary strengths.

In SL-based initialization, the model acquires routing priors by imitating expert decisions at each decoding step t, optimizing the negative log-likelihood loss $\mathcal{L}_{\text{NLL}}$. To improve sample efficiency and diversify supervision signals, we leverage the inherent symmetries of VRP solutions (Kwon et al., 2020; Kim et al., 2022) and apply data augmentation to training samples. We randomize the start nodes and traversal directions for TSP solutions, and permute sub-route orders for CVRP solutions.

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{|\mathcal{S}|}\sum\nolimits_{s_i\in\mathcal{S}} \log p_\theta(\tau_i(t)|s_i, \tau_{i,:t-1}), \quad \forall\, t \in [1,\dots,N] \tag{3}$$

Subsequently, RL refines the parameter p_θ to align with the combinatorial objective $R(\tau)$. For each instance s_i, we sample m distinct trajectories $\{\tau_i^1,\dots,\tau_i^m\}$ and compute their objective values $R(\tau_i^j)$. We take $\tau_{i,:t-1}^j$ to denote the partial solution prior to the t-th step along the j-th trajectory. As shown in Equation 4, the policy gradients are approximated via REINFORCE with a shared baseline $b(s)$, which reduces gradient variance and stabilizes training (Kwon et al., 2020). To alleviate the memory bottleneck of RL training, we adopt activation checkpointing (Chen et al., 2016), which trades recomputation overhead for reduced peak GPU memory, enabling a larger training batch size.

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{|\mathcal{S}|\cdot m}\sum\nolimits_{s_i\in\mathcal{S}}\sum\nolimits_{j=1}^{m}\big(R(\tau_i^j)-b(s_i)\big)\nabla_\theta \log p_\theta(\tau_i^j|s_i)\\
b(s_i) &= \frac{1}{m}\sum\nolimits_{j=1}^{m} R(\tau_i^j), \quad p_\theta(\tau_i^j|s_i) = \prod\nolimits_{t=1}^{N} p_\theta(\tau_i^j(t)|s_i,\tau_{i,:t-1}^j)
\end{aligned} \tag{4}$$

6 Experiments

To evaluate LPACR's performance, we conduct extensive experiments on two representative VRP variants, TSP and CVRP, spanning diverse instance scales and spatial distributions. All experiments are executed on a server with one AMD 7763 CPU and one NVIDIA 4090 GPU with 24GB VRAM.

Datasets For SL training, we generate 1 million instances for both TSP-100 and CVRP-100, and adopt Concorde and HGS to obtain expert solutions, respectively. We reduce HGS iterations from the default 20K to 2K to save computational cost. The RL training samples are generated on-the-fly.

The synthetic testing datasets cover 7 distributions (uniform, cluster, mixed, expansion, explosion, implosion, and grid) and 6 scales (100, 300, 500, 1K, 5K, and 10K), yielding 42 subsets per problem type. Implementation details of these synthetic distributions are provided in Appendix B. We generate 1K instances for scale 100, and 128 for each of the remaining subsets.

Table 1: Comparative results on large-scale synthetic datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}	T_{avg}
TSP-1K	LKH	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	1.1s
	LEHD	1.76%±0.08%	9.65%±0.45%	2.98%±0.11%	6.96%±0.15%	5.01%±0.18%	3.04%±0.17%	1.84%±0.09%	3.7s
	SIL	0.98%±0.10%	8.06%±0.40%	1.61%±0.12%	3.39%±0.18%	3.22%±0.27%	3.07%±0.20%	0.83%±0.11%	6.8s
	TTPL	1.53%±0.13%	15.88%±0.79%	2.86%±0.22%	3.40%±0.21%	3.85%±0.28%	3.28%±0.32%	1.73%±0.15%	5.2s
	INViT	5.47%±0.08%	17.58%±0.43%	9.11%±0.21%	8.93%±0.25%	11.13%±0.41%	7.81%±0.19%	4.89%±0.07%	1.1m
	L2R	3.27%±0.07%	15.43%±0.41%	5.95%±0.13%	7.12%±0.26%	8.53%±0.38%	5.85%±0.20%	3.08%±0.07%	6.9s
	ELG	16.78%±0.30%	13.42%±0.41%	18.16%±0.34%	16.83%±0.37%	16.78%±0.35%	16.45%±0.35%	14.97%±0.24%	1.9s
	GELD	2.15%±0.16%	12.15%±0.39%	4.87%±0.27%	4.89%±0.19%	4.46%±0.27%	4.53%±0.23%	2.30%±0.17%	5.4s
	DGL	4.50%±0.19%	13.05%±0.45%	6.94%±0.24%	6.73%±0.24%	8.30%±0.33%	5.53%±0.23%	4.33%±0.19%	4.2s
	LPACR	**0.20%**±0.06%	**2.99%**±0.31%	**0.46%**±0.06%	**0.93%**±0.11%	**0.57%**±0.11%	**0.57%**±0.08%	**0.09%**±0.05%	5.0s
TSP-5K	LKH	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	33.0s
	LEHD	14.57%±0.16%	30.75%±0.98%	16.73%±0.21%	20.88%±0.25%	19.64%±0.25%	15.98%±0.28%	13.54%±0.20%	1.2m
	SIL	1.95%±0.09%	9.11%±0.20%	3.11%±0.10%	4.90%±0.13%	4.76%±0.17%	3.00%±0.12%	1.86%±0.09%	34.1s
	TTPL	1.81%±0.09%	13.41%±0.34%	2.81%±0.10%	3.55%±0.16%	4.29%±0.21%	3.05%±0.17%	1.69%±0.09%	19.4s
	INViT	5.45%±0.05%	13.85%±0.23%	8.46%±0.13%	8.10%±0.14%	9.53%±0.19%	6.70%±0.10%	4.89%±0.04%	6.4m
	L2R	3.20%±0.03%	11.42%±0.22%	4.71%±0.10%	6.14%±0.14%	7.42%±0.17%	4.73%±0.10%	2.86%±0.02%	36.0s
	ELG	20.02%±0.19%	20.85%±0.60%	22.33%±0.39%	21.27%±0.26%	22.07%±0.27%	20.94%±0.23%	18.95%±0.35%	9.6s
	GELD	6.65%±0.11%	15.65%±0.23%	10.66%±0.16%	10.73%±0.13%	10.86%±0.18%	7.25%±0.13%	6.22%±0.12%	26.6s
	DGL	5.53%±0.12%	11.85%±0.24%	7.63%±0.14%	6.97%±0.14%	8.09%±0.14%	6.18%±0.14%	5.42%±0.12%	24.4s
	LPACR	**0.92%**±0.07%	**4.46%**±0.27%	**1.79%**±0.09%	**1.85%**±0.14%	**1.98%**±0.16%	**1.35%**±0.10%	**1.08%**±0.07%	24.8s
TSP-10K	LKH	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	2.5m
	LEHD	24.78%±0.68%	51.88%±2.66%	29.91%±0.57%	31.00%±0.86%	30.30%±1.06%	26.26%±0.76%	24.03%±0.87%	7.9m
	SIL	2.99%±0.07%	15.42%±0.21%	4.91%±0.09%	8.48%±0.12%	7.90%±0.18%	4.27%±0.11%	2.75%±0.07%	1.1m
	TTPL	1.72%±0.06%	12.12%±0.30%	2.53%±0.06%	3.38%±0.14%	4.38%±0.16%	3.07%±0.15%	1.50%±0.06%	47.7s
	INViT	5.41%±0.03%	13.04%±0.21%	8.24%±0.11%	7.49%±0.10%	9.02%±0.11%	6.36%±0.07%	4.98%±0.05%	14.6m
	L2R	3.19%±0.03%	10.23%±0.20%	4.42%±0.08%	5.32%±0.10%	6.70%±0.11%	4.20%±0.07%	2.82%±0.02%	1.2m
	ELG	20.70%±0.13%	20.77%±0.21%	23.12%±1.01%	22.16%±0.23%	25.03%±1.13%	21.40%±0.17%	19.38%±0.14%	19.1s
	GELD	9.95%±0.08%	17.18%±0.20%	13.77%±0.12%	13.25%±0.10%	13.56%±0.13%	10.31%±0.13%	9.28%±0.09%	54.3s
	DGL	6.38%±0.09%	11.39%±0.21%	7.78%±0.10%	6.85%±0.12%	8.14%±0.12%	6.34%±0.11%	5.78%±0.08%	44.9s
	LPACR	**1.26%**±0.06%	**7.16%**±0.29%	**2.40%**±0.08%	**2.23%**±0.12%	**2.67%**±0.13%	**1.24%**±0.09%	**1.15%**±0.06%	49.1s
CVRP-1K	HGS	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	12.0m
	LEHD	2.32%±0.07%	4.97%±0.16%	3.40%±0.09%	4.83%±0.13%	4.36%±0.14%	4.54%±0.80%	2.41%±0.08%	4.1s
	TTPL	1.86% ± 0.07%	4.17% ± 0.09%	3.01% ± 0.09%	3.86% ± 0.09%	3.49% ± 0.11%	2.54% ± 0.09%	1.91% ± 0.07%	5.0s
	INViT	9.04% ± 0.11%	9.77% ± 0.23%	10.00% ± 0.15%	9.03% ± 0.10%	9.26% ± 0.11%	9.01% ± 0.12%	9.31% ± 0.12%	15.4s
	L2R	11.12% ± 0.22%	5.97% ± 0.13%	10.74% ± 0.19%	10.70% ± 0.15%	11.59% ± 0.20%	10.33% ± 0.21%	11.51% ± 0.22%	8.4s
	ELG	7.12% ± 0.15%	8.51% ± 0.39%	7.69% ± 0.20%	8.54% ± 0.26%	8.82% ± 0.39%	7.50% ± 0.24%	7.47% ± 0.19%	3.7s
	DGL	9.81% ± 0.21%	7.28% ± 0.16%	10.03% ± 0.22%	9.04% ± 0.17%	9.77% ± 0.19%	9.34% ± 0.19%	10.46% ± 0.21%	6.4s
	LPACR	**-0.08%**±0.05%	**1.09%**±0.05%	**0.61%**±0.07%	**0.48%**±0.06%	**0.50%**±0.06%	**0.26%**±0.06%	**0.02%**±0.05%	5.7s
CVRP-5K	HGS	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	1.0h
	LEHD	6.31%±0.21%	13.74%±1.59%	9.18%±0.82%	13.85%±1.17%	11.04%±0.72%	28.24%±8.62%	6.27%±0.26%	1.2m
	TTPL	2.42%±0.05%	4.11%±0.06%	3.16%±0.06%	3.57%±0.07%	3.57%±0.08%	3.85%±0.53%	2.30%±0.05%	24.9s
	INViT	7.68%±0.10%	11.90%±0.68%	10.69%±0.40%	7.80%±0.15%	7.62%±0.13%	7.92%±0.31%	7.74%±0.10%	1.7m
	L2R	6.03%±0.14%	2.27%±0.08%	5.16%±0.13%	5.87%±0.13%	6.40%±0.12%	4.58%±0.13%	5.95%±0.14%	43.5s
	ELG	8.51%±0.25%	14.20%±0.75%	10.03%±0.44%	15.32%±2.36%	16.37%±2.84%	10.69%±0.55%	8.18%±0.24%	17.7s
	DGL	4.40%±0.10%	3.36%±0.11%	4.42%±0.10%	3.76%±0.09%	4.01%±0.07%	3.78%±0.10%	4.35%±0.09%	35.0s
	LPACR	**-1.45%**±0.01%	**-0.74%**±0.03%	**-1.03%**±0.02%	**-1.33%**±0.03%	**-1.16%**±0.03%	**-1.42%**±0.04%	**-1.39%**±0.01%	28.3s
CVRP-10K	HGS	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	2.0h
	LEHD	10.21%±0.40%	23.56%±3.06%	14.08%±1.22%	24.05%±5.56%	19.92%±1.93%	28.20%±8.52%	12.40%±1.34%	8.1m
	TTPL	4.36%±0.18%	8.25%±0.54%	5.03%±0.23%	7.25%±0.28%	7.65%±0.30%	5.53%±0.47%	4.73%±0.19%	49.7s
	INViT	8.21%±0.13%	8.73%±0.43%	11.38%±0.48%	8.44%±0.42%	8.87%±0.29%	8.57%±0.46%	8.20%±0.15%	5.3m
	L2R	4.08%±0.09%	1.16%±0.06%	3.64%±0.09%	3.98%±0.09%	4.61%±0.09%	3.40%±0.09%	4.33%±0.09%	1.5m
	ELG	10.79%±0.50%	14.67%±1.19%	14.70%±0.83%	22.60%±2.67%	24.28%±3.49%	13.56%±0.77%	10.50%±0.41%	35.5s
	DGL	3.04%±0.06%	2.61%±0.09%	3.28%±0.07%	2.60%±0.07%	2.98%±0.06%	2.94%±0.08%	3.26%±0.07%	1.2m
	LPACR	**-1.51%**±0.01%	**-0.97%**±0.03%	**-1.12%**±0.01%	**-1.52%**±0.03%	**-1.31**%±0.02%	**-1.49%**±0.03%	**-1.46%**±0.01%	55.2s

The bold value indicates that the method achieves the lowest average relative gap among all evaluated neural solvers.

Additionally, we adopt 9 real-world benchmarks including TSPLib, VLSI, NationalTSP, and 6 subsets from CVRPLib. These real-world benchmarks present substantial distributional complexity and scale variability for evaluating OOD generalization capabilities.

Training Setup We employ Adam (Kingma & Ba, 2014) as the optimizer for both SL and RL, with an initial learning rate of 1e-4. In the SL phase, the learning rate decays by a factor of 0.97 every 3 epochs. We set the batch size to 1000 and train for 30 epochs, with each epoch processing 1M training samples in random order. In the RL phase, we train for 20 epochs, each comprising 1000 batches of 16 instances. For each instance, we sample 100 trajectories to compute the shared

Table 2: Comparative results on real-world benchmarks.

Method	TSPLIB	VLSI	National	set-X	set-XL	set-CMT	set-Golden	set-Li	set-AGS
AMDKD	29.4%±3.4%	64.8%±3.2%	85.7%±17.5%	32.8%±2.4%	>99.9%	7.7%±4.0%	44.0%±4.6%	>99.9%	>99.9%
OmniVRP	25.5%±2.5%	68.2%±5.8%	69.9%±9.7%	12.4%±0.6%	89.2%±15.4%	6.5%±3.1%	12.2%±0.8%	27.7%±3.6%	>99.9%
LEHD	7.9%±1.3%	15.9%±1.6%	23.6%±4.2%	12.5%±1.9%	29.5%±3.0%	0.1%±2.3%	5.4%±0.5%	**5.6%**±1.3%	27.1%±3.6%
SIL	6.2%±0.6%	10.5%±1.2%	14.4%±1.7%	94.7%±7.3%	62.6%±5.8%	94.3%±12.1%	44.7%±11.4%	6.4%±1.2%	18.3%±3.7%
TTPL	5.4%±1.0%	4.4%±0.2%	6.7%±0.7%	9.7%±1.0%	19.6%±2.1%	**-1.7%**±2.4%	6.2%±0.5%	10.0%±2.0%	22.9%±4.3%
INViT	9.1%±0.6%	10.2%±0.3%	11.5%±0.9%	13.5%±0.6%	16.6%±0.8%	6.8%±2.9%	12.7%±0.9%	10.8%±1.9%	24.2%±2.7%
L2R	7.0%±0.6%	7.1%±0.3%	8.5%±0.7%	13.2%±0.5%	13.1%±0.7%	6.9%±3.0%	13.2%±0.7%	13.8%±1.9%	15.5%±0.9%
ELG	11.8%±1.0%	22.5%±0.8%	22.6%±2.2%	11.7%±0.6%	29.8%±3.8%	3.8%±3.1%	10.9%±0.7%	13.4%±1.6%	29.7%±2.7%
GELD	6.2%±0.7%	12.3%±0.6%	14.6%±1.4%	/	/	/	/	/	/
DGL	6.5%±0.5%	10.5%±0.3%	9.7%±0.8%	18.8%±1.0%	16.9%±1.3%	10.8%±3.5%	20.1%±1.4%	15.9%±1.8%	20.9%±2.2%
LPACR	**2.1%**±0.3%	**3.5%**±0.2%	**3.7%**±0.5%	**7.5%**±0.7%	**11.0%**±1.2%	-1.6%±2.5%	**4.7%**±0.5%	6.9%±1.2%	**9.9%**±2.0%

The bold value indicates that the method achieves the lowest average relative gap among all evaluated neural solvers.
The underlined value indicates that the method fails on some large-scale instances, and the results are computed from the remaining ones.
The *>99.9%* value indicates that the method yields an average relative gap larger than 99.9%, exhibiting very limited performance.

Table 3: Ablation study results on real-world benchmarks.

Method	TSPLIB	VLSI	National	set-X	set-XL	set-CMT	set-Golden	set-Li	set-AGS
LPACR	**2.1%**±0.3%	**3.5%**±0.2%	**3.7%**±0.5%	**7.5%**±0.7%	**11.0%**±1.2%	-1.6%±2.5%	**4.7%**±0.5%	6.9%±1.2%	**9.9%**±2.0%
w/o GDC	4.1%±0.6%	5.6%±0.3%	6.2%±0.7%	16.2%±2.4%	13.7%±1.7%	0.0%±2.3%	4.9%±0.5%	**5.4%**±1.1%	12.8%±2.6%
w/o MDT	4.5%±0.6%	7.2%±0.4%	8.8%±0.9%	9.4%±1.0%	11.8%±1.0%	**-2.7%**±2.5%	6.3%±0.7%	17.0%±2.7%	11.8%±2.0%
w/ SL+	2.3%±0.3%	3.7%±0.2%	5.1%±0.7%	10.2%±0.9%	12.9%±1.4%	4.6%±3.8%	6.6%±0.9%	16.0%±2.3%	12.5%±2.1%

The bold value indicates that the method achieves the lowest average relative gap among all evaluated configurations.

baseline. The learning rate decays by a factor of 0.9 per epoch. For TSP and CVRP, the SL phase takes 81 and 84 hours while the RL phase takes 114 and 118 hours.

Regarding model hyperparameters, we set the embedding dimension to 128. The decoder stacks $L = 6$ attention layers, where each MHA sub-layer has 8 attention heads, and the inner dimension of 512 in FF sub-layer. We set $K = 50$ to construct local sub-problems in the training phase.

Baselines We compare LPACR against **1) classical OR solvers**: Concorde, LKH3, and HGS, and **2) constructive neural solvers**: AMDKD, OmniVRP, LEHD, TTPL, INViT, L2R, ELG, SIL, GELD, and DGL. We run all these neural baselines using their original checkpoints on our datasets. Detailed descriptions and configurations of these baselines are provided in Appendix G.

Inference Setup We set the sub-problem scale $K = 100$ for TSP and $K = 65$ for CVRP, as our empirical evaluation demonstrates that slightly larger local sub-problems improve practical solution quality during inference. For a fair comparison, all neural solvers are constrained to greedy decoding with a batch size of 1. For TSP instances, inference originates from the node indexed as 0.

Evaluation Metrics Performance is evaluated via relative gap (G) and inference time (T). The gap measures the relative deviation from the reference solution, with lower values indicating higher solution quality. Gaps are reported as mean ± standard error of the mean (SEM) for each subset. Regarding reference solution acquisition, we adopt Concorde to derive optimal solutions for TSP instances with scale $N \leq 500$, and LKH3 (with trials $= N$ and runs $= 1$) to generate near-optimal solutions for larger-scale instances. For CVRP, we run HGS with the default 20K iterations for $N \leq 500$, and set linearly increasing runtime limits of 0.2, 1.0, and 2.0 hours for $N =$ 1K, 5K, and 10K, respectively, to control computational overhead. We report the average inference times, recorded in seconds (s), minutes (m), or hours (h), to quantify computational efficiency.

6.1 Comparative Results

Table 1 presents comparative results on large-scale synthetic instances with scale $N \geq 1K$, and the corresponding results on small-scale instances are provided in Appendix H. Notably, LPACR demonstrates robust OOD generalization capabilities and achieves state-of-the-art solution quality, consistently outperforming all neural baselines across synthetic datasets.

Cross-scale Performance LPACR maintains robust solution quality on instances sampled from its training distributions, with scales varying from 100 to 10K nodes. Remarkably, on the uniform

TSP-10K subset, it achieves an average relative gap of 1.26%, yielding a 27% relative reduction compared to the best-performing neural baseline method, TTPL. Moreover, LPACR outperforms the LKH3 solvers on most CVRP instances, and surpasses the HGS solver on CVRP-5K/10K instances. Regarding computational efficiency, LPACR's inference time scales nearly linearly with problem scale, consistent with its near-constant per-step complexity (as analyzed in Appendix E).

Cross-distribution Generalization Performance Most neural solvers are trained solely on the uniform distribution, and exhibit significant performance degradation on non-uniform distributions, particularly the cluster distribution. In contrast, LPACR outperforms them by a substantial margin not only on complex training distributions (cluster, mixed) but also on unseen testing distributions (expansion, explosion, implosion, grid), demonstrating its superior cross-distribution generalization capability for handling complex local topological structures. Nevertheless, notable gaps persist for cluster distributed TSP instances, suggesting that further exploration remains necessary.

Performance on Real-world Benchmarks Existing methods are typically evaluated on limited real-world benchmarks, whereas our paper provides wider coverage. As summarized in Table 2, neural baseline methods exhibit significant performance variation across different benchmarks. By contrast, LPACR achieves the lowest average relative gaps on 7 subsets and competitive rankings on the remaining 2 subsets, confirming its robustness in practical scenarios.

6.2 Ablation Study

We conduct ablation studies to verify the contributions of each key component in LPACR. Most configurations adopt the same training budget as LPACR, except that the "w/ SL+" setting omits the RL training phase and adopts 30 additional SL training epochs. The "w/o GDC" setting restricts the context to local sub-problems without incorporating grid embeddings, and the "w/o MDT" setting takes training samples generated from a single uniform distribution. Results on real-world benchmarks are presented in Table 3 while results on synthetic datasets are presented in Appendix I.

First, ablating the GDC method from LPACR yields consistent performance degradation across synthetic and most real-world benchmarks, highlighting the necessity of global contextual awareness. Second, training exclusively on the uniform distribution leads to slight improvements in several synthetic TSP-100 subsets. For most of the remaining datasets, the MDT strategy results in better solution quality for both uniform and non-uniform distributions, demonstrating the effectiveness of local topological diversity for enhancing cross-distribution generalization. Finally, conducting more SL training may be helpful for improving performance on small-scale TSP instances, but it consistently falls behind the two-stage learning paradigm on large-scale instances. In addition, RL refinement yields substantial gains on CVRP instances.

6.3 Sensitivity Analysis

We conduct sensitivity analysis on the key hyperparameters, grid resolution G, to understand its impact. Results are presented in Section J. We retrain LPACR with different fixed or dynamic grid resolutions. In brief, LPACR is robust across different grid resolutions, and the default $G = 6 \times 6$ yields the best results on most evaluated datasets.

7 Conclusion

In this paper, we propose LPACR, a novel constructive VRP solver that addresses both cross-scale and cross-distribution generalization challenges within a unified framework. By integrating the local policy with a grid-based distribution compression method, LPACR effectively injects global contextual priors into local decision-making. We identify local topological diversity as a critical factor for cross-distribution generalization and introduce a multi-distribution training strategy to systematically encompass heterogeneous local patterns. Furthermore, we employ a two-stage learning algorithm that synergizes SL and RL to balance rapid initialization with global objective alignment. Extensive experiments on synthetic datasets and real-world benchmarks demonstrate that LPACR achieves state-of-the-art OOD generalization performance across TSP and CVRP variants.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the manuscript. We have not used generative AI tools for formulating the core research ideas, research questions or scientific conclusions. We have reviewed and revised all AI-assisted text. We take responsibility for the final content of this work, including all text, analyses, claims, figures, and conclusions.

REFERENCES

David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The Traveling Salesman Problem: A Computational Study. In *The Traveling Salesman Problem*. Princeton university press, 2011.

Florian Arnold, Michel Gendreau, and Kenneth Sörensen. Efficiently solving very large-scale routing problems. *Computers & operations research*, 107:32–42, 2019.

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

Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning Generalizable Models for Vehicle Routing Problems via Knowledge Distillation. *Advances in Neural Information Processing Systems*, 35:31226–31238, 2022.

Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, and Heike Trautmann. Evolving diverse TSP instances by means of novel and creative mutation operators. In *Proceedings of the 15th ACM/SIGEVO conference on foundations of genetic algorithms*, pp. 58–71, 2019.

Jinbiao Chen, Jiahai Wang, Zhiguang Cao, and Yaoxin Wu. Neural Multi-Objective Combinatorial Optimization via Graph-Image Multimodal Fusion. In *International Conference on Learning Representations*, volume 2025, pp. 33964–33983, 2025a.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Xinwei Chen, Yurui Li, Yifan Yang, Li Zhang, Shijian Li, and Gang Pan. FGDC: A fine-grained divide-and-conquer approach for extending NCO to solve large-scale Traveling Salesman Problem. *Expert Systems with Applications*, 286:127950, 2025b.

Xinwei Chen, Li Zhang, Shijian Li, and Gang Pan. Boosting Zero-Shot Generalization Performance of Constructive NCO Approaches for Large-Scale TSP. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2025c.

Yuanyao Chen, Rongsheng Chen, Fu Luo, and Zhenkun Wang. Improving Generalization of Neural Combinatorial Optimization for Vehicle Routing Problems via Test-Time Projection Learning. *Advances in Neural Information Processing Systems*, 38:76543–76585, 2026.

Nicos Christofides. The vehicle routing problem. *Combinatorial Optimization*, 1979.

William J Cook. National Traveling Salesman Problems. Data set, University of Waterloo, 2009. URL `https://www.math.uwaterloo.ca/tsp/world/countries.html`.

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.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. INViT: A Generalizable Routing Problem Solver with Invariant Nested View Transformer. In *International Conference on Machine Learning*, pp. 12973–12992. PMLR, 2024.

Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards Generalizable Neural Solvers for Vehicle Routing Problems via Ensemble with Transferrable Local Policy. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6914–6922, 2024.

Yong Liang Goh, Zhiguang Cao, Yining Ma, Jianan Zhou, Mohammed Haroon Dupty, and Wee Sun Lee. Shield: Multi-task multi-distribution vehicle routing solver with sparsity and hierarchy. In *Forty-second International Conference on Machine Learning*, 2025.

Bruce L Golden, Edward A Wasil, James P Kelly, and I-Ming Chao. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In *Fleet Management and Logistics*, pp. 33–56. Springer, 1998.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Keld Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

Keld Helsgaun. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. *Roskilde: Roskilde University*, 12:966–980, 2017.

Karla L Hoffman, Manfred Padberg, Giovanni Rinaldi, et al. Traveling salesman problem. *Encyclopedia of Operations Research and Management Science*, 1:1573–1578, 2013.

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.

Yuan Jiang, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. Learning to Solve Routing Problems via Distributionally Robust Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9786–9794, 2022.

Yuan Jiang, Zhiguang Cao, Yaoxin Wu, Wen Song, and Jie Zhang. Ensemble-based Deep Reinforcement Learning for Vehicle Routing Problems under Distribution Shift. *Advances in Neural Information Processing Systems*, 36:53112–53125, 2023.

Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1):70–98, 2022.

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.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, Learn to Solve Routing Problems! In *International Conference on Learning Representations*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Feiyue Li, Bruce Golden, and Edward Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179, 2005.

Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards Out-Of-Distribution Generalization: A Survey. *arXiv preprint arXiv:2108.13624*, 2021.

Shiyan Liu, Bohan Tan, Zhiguang Cao, and Yan Jin. VAGPO: Vision-augmented Asymmetric Group Preference Optimization for Graph Routing Problems. *arXiv preprint arXiv:2508.01774*, 2025.

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.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. Boosting Neural Combinatorial Optimization for Large-Scale Vehicle Routing Problems. In *The Thirteenth International Conference on Learning Representations*, 2025.

Dian Meng, Zhiguang Cao, Jie Gao, Yaoxin Wu, and Yaqing Hou. UniteFormer: Unifying Node and Edge Modalities in Transformers for Vehicle Routing Problems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. H-TSP: Hierarchically Solving the Large-Scale Traveling Salesman Problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9345–9353, 2023.

Jonathan Pirnay and Dominik G Grimm. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *arXiv preprint arXiv:2403.15180*, 2024.

Eduardo Queiroga, Rafael Martinelli, Anand Subramanian, Eduardo Uchoa, and Thibaut Vidal. The xl instances and the cvrplib best known solution challenge. *arXiv preprint arXiv:2601.11467*, 2026.

Gerhard Reinelt. TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

Andre Rohe. VLSI Data Sets. Data set, University of Waterloo, 2013. URL `https://www.math.uwaterloo.ca/tsp/vlsi/index.html`.

Noam Shazeer. GLU Variants Improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Hao Tan and Mohit Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 5100–5111, 2019.

Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. SIAM, 2002.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. *Advances in neural information processing systems*, 30, 2017.

Thibaut Vidal. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140:105643, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yubin Xiao, Di Wang, Rui Cao, Xuan Wu, Boyang Li, and You Zhou. GELD: A unified neural model for efficiently solving traveling salesman problems across different scales. *Pattern Recognition*, pp. 112865, 2025a.

Yubin Xiao, Yuesong Wu, Rui Cao, Di Wang, Zhiguang Cao, Xuan Wu, Peng Zhao, Yuanshu Li, You Zhou, and Yuan Jiang. DGL: Dynamic Global-Local Information Aggregation for Scalable VRP Generalization with Self-Improvement Learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 8669–8677, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Haotian Ye, Chuanlong Xie, Tianle Cai, Ruichen Li, Zhenguo Li, and Liwei Wang. Towards a Theoretical Framework of Out-of-Distribution Generalization. *Advances in Neural Information Processing Systems*, 34:23519–23531, 2021.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. *Advances in neural information processing systems*, 32, 2019.

Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. Learning to solve travelling salesman problem with hardness-adaptive curriculum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9136–9144, 2022.

Changliang Zhou, Xi Lin, Zhenkun Wang, and Qingfu Zhang. Learning to Reduce Search Space for Generalizable Neural Routing Solver. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6841–6852, 2026.

Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards Omni-generalizable Neural Methods for Vehicle Routing Problems. In *International Conference on Machine Learning*, pp. 42769–42789. PMLR, 2023.

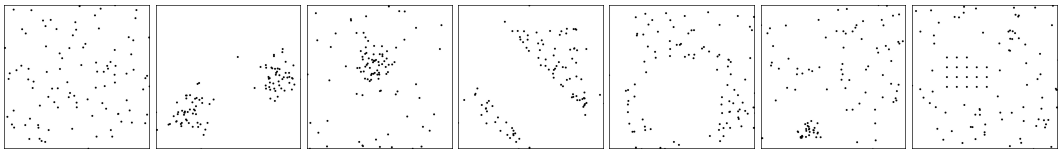
A PRELIMINARIES

In this section, we present the problem statement for VRPs. We define a VRP instance with N nodes on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}_{i=1}^{N}$ represents the set of nodes (e.g., cities in TSP or customers and the depot in CVRP), and $\mathcal{E}$ contains edges (v_i, v_j) for all $i \neq j$. A solution (tour) π is defined as a sequence of nodes in $\mathcal{V}$. This paper focuses on 2D Euclidean VRPs, where the objective value C_π is the total length of π. The goal is to find the optimal solution $\pi^* = argmin_{\pi \in \mathcal{S}} C_\pi$, where $\mathcal{S}$ denotes the discrete space of all feasible solutions satisfying problem-specific constraints. For a TSP instance, the node features are Euclidean coordinates $v_i = \{x_i, y_i\}$, and a feasible solution must visit each node exactly once and return to the start node. For a CVRP instance, an additional depot node v_0 is included, each node v_i has a demand δ_i ($v_i = \{x_i, y_i, \delta_i\}$), and the vehicle has a capacity limit Q. A feasible CVRP solution consists of multiple sub-tours originating from and returning to the depot, where the cumulative demand of each sub-tour does not exceed the capacity, and each customer node is visited exactly once.

Constructive neural solvers typically adopt an encoder-decoder architecture, which learns node selection heuristics for solution construction. The encoder maps raw instance features into hidden representations, while the decoder leverages these representations and the current solving context to compute selection probabilities for the next node. The selected node is appended to the end of the partial solution, and this repeats until a complete solution is obtained. Thus, the probability of the solution τ_i is factorized via the chain rule: $p_\theta(\tau_i|s_i) = \prod_{t=1}^{N} p_\theta(\tau_i(t)|s, \tau_{i,:t-1})$, where θ, $\tau_i(t)$, and $\tau_{i,:t-1}$ denote the learned policy parameters, the node selected at the t-th step, and the partial solution prior to the t-th step, respectively.

B DETAILS OF INSTANCE DISTRIBUTIONS

In this section, we detail the distributions used to generate synthetic instances for the training and testing phases, following implementation protocols from recent works (Bossek et al., 2019; Bi et al., 2022; Zhang et al., 2022; Fang et al., 2024). Specifically, as illustrated in Figure 3, we employ uniform, cluster, mixed, expansion, explosion, implosion, and grid distributions to generate diverse node coordinates.



(a) Uniform (b) Cluster (c) Mixed (d) Expansion (e) Explosion (f) Implosion (g) Grid

Figure 3: Illustration of synthetic VRP instances generated from various distributions.

Uniform This is the most common setting used in literature, where the 2-dimensional coordinates $\{x_i, y_i\}_{i=1}^{N}$ of N nodes are sampled from a uniform distribution over the $[0, 1]^2$ unit square.

Cluster This setting features n_c clusters, each containing a similar number of nodes. We follow the setting of AMDKD (Bi et al., 2022) to generate cluster distributed instances as training samples, where each cluster i follows a normal distribution $\mathcal{N}^2(\mu_i, \sigma_i^2)$ with the mean μ_i sampled uniformly from $[0.2, 0.8]$ and the standard deviation $\sigma_i = 0.07$. Outlier nodes outside the $[0, 1]^2$ unit square are clipped within the region. However, this setting induces severe cluster overlap in large-scale instances. Therefore, we adapt the Gaussian Mixture generator proposed in HAC (Zhang et al., 2022) to generate samples for larger-scale instances. Specifically, it considers n_c clusters, each following a normal distribution $\mathcal{N}^2(\mu_i, \sigma_i^2)$, where the mean μ_i is sampled uniformly from $[0, l]$, and the standard deviation $\sigma_i = 1$. We set $n_c = \lfloor N/50 \rfloor$, $l = 4 \times n_c$ for $N \leq 500$; $n_c = 20$, $l = 60$ for $N = 1K$; $n_c = 30$, $l = 60$ for $N = 5K$; and $n_c = 40$, $l = 60$ for $N = 10K$

Mixed This setting comprises a combination of uniform and cluster distributions, where half the nodes are generated from each distribution.

Expansion This setting mutates uniformly distributed nodes through a randomly generated linear function $y = ax + b$. The intercept b is sampled from $[0, 1]$, and the slope a is subsequently sampled

uniformly from $[0, 3]$ if $b < 0.5$, or from $[-3, 0]$ if $b \geq 0.5$. All nodes with orthogonal distance to the linear function no greater than d_r are shifted radially outward to a farther position, with new orthogonal distance being $d_r + \gamma_i$. For each shifted node i, γ_i is sampled from an exponential distribution with rate parameter $\lambda = 10$, i.e., $\gamma_i \sim E(\lambda)$. We set $d_r = 0.3$ for $N = 100$, and $d_r = 0.2$ for $N \geq 300$.

Explosion This setting mutates uniformly distributed nodes by imitating the particles affected by an explosion. To simulate an explosion, a center node is uniformly sampled from $[0, 1]^2$. Subsequently, each node i within a circle of radius R centered at this center is exploded outward by a distance γ_i, along the direction from the center to the node. The radius R is sampled uniformly from $[0.1, 0.5]$, and the distance γ_i is sampled from an exponential distribution with a rate parameter $\lambda = 10$.

Implosion This setting applies an implosion mechanism to mutate uniformly distributed nodes. In contrast to the explosion distribution, all nodes within the circle converge toward a smaller circle of radius $R' = \min(R, |\xi|)$, where $\xi \sim \mathcal{N}(0, 1)$, along the directions from each corresponding node to the center.

Grid This setting employs a grid permutation to mutate uniformly distributed nodes. We first sample the lower left coordinates of a square box uniformly from $[0, 1 - R_g]$, where R_g is the side length of the box, and then all nodes inside the box are rearranged into a regular square grid. We set $R_g = 0.4$ if $N = 100$, and $R_g = 0.3$ otherwise.

All these generated coordinates are finally scaled proportionally into the $[0, 1]^2$ unit square through Min-Max normalization for numerical consistency.

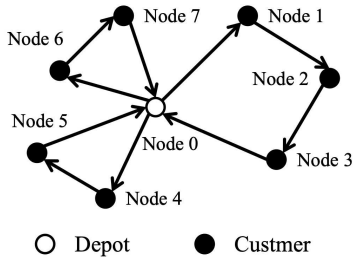
Demand and Capacity in CVRP Instances The coordinates of the depot and customer nodes in CVRP instances are generated according to the aforementioned distributions. For demand and vehicle capacity, we follow conventional settings (Bi et al., 2022; Kool et al., 2019; Kwon et al., 2020) where the demand δ_i of node i is uniformly sampled from $\mathcal{U}(1, 2, \ldots, 9)$ and the capacity of the vehicle is set to $Q = 50$.

C Details of CVRP Implementation

Binary Representation of CVRP Solutions As described in Section A, CVRP contains a depot node and an additional capacity constraint. To satisfy this constraint, vehicles must periodically return to the depot to maintain sufficient space for serving the remaining customers. However, this characteristic leads to inconsistencies in the number of remaining customers across parallel instances, hindering batched inference. Following implementations of existing neural solvers (Kool et al., 2019; Luo et al., 2023; Drakulic et al., 2023), we adopt a binary representation for CVRP solutions. Specifically, binary flags are assigned to each customer, where 1 signifies a route initiation from the depot, while 0 denotes a direct continuation from the previously visited customer. For example, a CVRP solution {0, 1, 2, 3, 0, 4, 5, 0, 6, 7, 0} (where 0 represents the depot) and its corresponding binary representation are illustrated in Figure 4.

Model Architecture The raw node feature v_i in a CVRP instance is a 3D vector comprising the 2D coordinates $\{x_i, y_i\}$ and the demand δ_i, with $\delta_0 = 0$ for the depot. We normalize demands by the vehicle capacity Q, i.e., $\hat{\delta}_i = \delta_i / Q$, to ensure numerical stability. Given the feature v_0 of the first visited node (i.e., the depot), the start embedding is obtained by a linear projection such that $H_{\text{st}} = v_0 W_{\text{st}} + b_{\text{st}}$, where weight $W_{\text{st}} \in \mathbb{R}^{3 \times d}$ and bias $b_{\text{st}} \in \mathbb{R}^d$ are learnable parameters, and $d = 128$ denotes the embedding dimension. Similarly, candidate node embeddings H_n are computed via a shared linear projection: $H_n = \hat{V}_n W_n + b_n$, with $W_n \in \mathbb{R}^{3 \times d}$ and $b_n \in \mathbb{R}^d$ are shared learnable parameters. Here, the matrix $\hat{V}_n$ stacks the normalized features $\hat{v}_i = \{x'_i, y'_i, \hat{\delta}_i\}$ for each candidate node i in the current normalized sub-problem. Regarding the convolutional grid encoder, its input dimensions are adapted to match the expanded statistical grid features for CVRP, while preserving the core architecture.

In the decoder, the normalized remaining capacity (i.e., available space divided by the vehicle capacity Q) is concatenated with the embedding of the last visited node H_l at each decoding step. This concatenated vector is then projected into the hidden space via a learnable linear layer, serving as a crucial contextual token for calculating node selection probabilities.



(a) CVRP solution

1	**2**	**3**	**4**	**5**	**6**	**7**
1	**0**	**0**	**1**	**0**	**1**	**0**

(b) Binary representation of the CVRP solution

Figure 4: Illustration of a CVRP solution and its corresponding binary representation.

While the internal attention layers align with the TSP implementation, the final output layer differs. Whereas the TSP output layer generates exactly c logits for the given c candidate nodes, the CVRP output layer produces $2c$ logits to align with the binary representation. These logits are transformed into node selection probabilities via the softmax function. The former c values correspond to the probability of continuing the current route to the next customer, while the latter c values represent the probability of initiating a new route from the depot to that customer.

Furthermore, to guarantee solution feasibility, a route must return to the depot whenever a customer's demand exceeds the remaining capacity. This constraint is enforced via a deterministic feasibility check at each decoding step, which automatically overrides infeasible decisions' binary flag to 1 without altering their probability, ensuring strict constraint satisfaction.

D Local Sub-problem Construction

LPACR leverages the spatial locality of VRPs by restricting stepwise search spaces to local sub-problems, enabling efficient scaling to large-scale instances. Although this constrains the search to a subset of the global solution space and inevitably introduces deviations, it captures invariant structural patterns and drastically reduces computational overhead, thereby significantly improving cross-scale generalization performance. To construct the local sub-problem, we select the last visited node as the anchor node. Subsequently, the K nearest unvisited neighbors of the anchor node, together with the anchor node itself, constitute the local sub-problem for the current step.

While this construction is intuitive for TSP, a complication arises from the binary representation of CVRP solutions. The return-to-depot decision is directly integrated into the next customer visitation decision, and denoted by a binary flag. Ideally, when initializing a new route from the depot, the incumbent sub-problem should take the depot as its anchor node. This complication can be addressed by identifying customer nodes that immediately succeed the depot (e.g., nodes 4 and 6 in the preceding example), reconstructing the corresponding local sub-problems, performing an additional inference step, and overriding previous decisions. However, this practice renders the training process fragmented, and yields marginal gains in performance while doubling inference latency according to our preliminary explorations. Therefore, we directly use the last visited customer as the anchor node to construct the local sub-problem for simplicity. Nevertheless, this issue merits further exploration in the future.

Another challenge concerns the expert decisions in the SL phase. Although most expert decisions lie close to the last visited nodes due to the locality of VRPs, exceptional cases still exist where these decisions fall outside the local sub-problems. To address this issue, we first select $K - 1$ nearest neighbors while excluding the expert decision, and then explicitly append the expert decision to construct the sub-problem. Otherwise, we would lose valid supervision signals for these samples. We adopt the former approach in our implementation. As for CVRP, customer nodes that immediately succeed the depot exhibit a higher tendency to fall outside. We adopt the same treatment as in TSP, but this practice may cause training instability. Fortunately, in the RL refinement phase, the policy focuses on optimizing local sub-problems without considering these outliers. This may also explain the remarkable performance improvement brought by RL for CVRP instances in the ablation studies.

E TIME COMPLEXITY ANALYSIS

We employ the k-NN algorithm to retrieve the K nearest unvisited neighbors of the last visited node and construct the local sub-problem. Precomputing a complete k-NN set for all nodes would require an $O(N^2)$ matrix, which is not memory-efficient for large-scale instances. Therefore, we recompute the k-NN nodes at each decoding step. Specifically, we calculate the distances between the last visited node and all unvisited nodes in $O(N)$ time, and then retrieve the K nearest neighbors using torch.topk, which runs in $O(N)$ expected time on modern deep learning frameworks. The grid statistics can be initialized in $O(N)$ complexity by traversing all nodes and are incrementally updated at each decoding step, also in $O(N)$ time.

Subsequently, $T = K + G + 3$ tokens form the input sequence to the decoder. The decoder's self-attention cost is $O(T^2)$. Since both K and G are constants much smaller than N for large-scale instances, this term is asymptotically negligible compared to the $O(N)$ cost of k-NN retrieval and grid feature computation. Consequently, the per-step theoretical complexity is $O(N)$, and the overall decoding complexity is $O(N^2)$ over the $O(N)$ autoregressive decoding steps.

In practice, the local sub-problem construction and grid feature computations are fully vectorized and memory-contiguous, and thus execute efficiently on modern hardware. In contrast, the decoder's self-attention, despite its lower asymptotic complexity relative to N, involves multiple matrix operations, softmax, and repeated kernel launches at every step. As a result, the decoder dominates the actual wall-clock runtime.

Overall, LPACR is highly scalable and memory-efficient, with consistent stepwise runtime across different problem scales. The dominant practical cost lies in the autoregressive decoder.

F INVESTIGATION ON LOCAL TOPOLOGICAL STRUCTURES

Distributional Visualization Based on quantitative metrics, we project local sub-problems with diverse structural attributes into a 2D indicator space for visualization. Each anchor node's local sub-problem is mapped to a point in this space, where the x-axis represents the mean kNN distance and the y-axis denotes the coefficient of variation of these distances. We employ a color-encoded density heatmap, in which darker shades indicate a higher concentration of sub-problems sharing similar topological characteristics. As illustrated in Figures 2a – 2c, we visualize TSP-100 testing instances generated from uniform, cluster, and mixed distributions, respectively. Figure 2d presents the structural distribution of instances sampled using our proposed MDT strategy with the same total number of instances.

Several observations emerge from the visualization results. First, there are pronounced structural shifts across distribution types, with dominant patterns varying substantially in mean kNN distance. Second, the MDT strategy yields a more balanced configuration that mitigates the structural disparities among individual distribution types. Notably, it achieves broader coverage in the indicator space compared to the uniform distribution used for conventional training. This expanded coverage exposes LPACR to a richer spectrum of local topological structures, which we hypothesize contributes to its superior cross-distribution generalization performance.

Impact on Neural Predictions Given that nodes proximal to the current position typically receive higher attention weights during the forward propagation, we hypothesize that the local topological structure associated with the last visited node exerts a critical impact on neural predictions. To empirically validate this, we evaluate the quality of neural predictions along ground-truth trajectories. Specifically, we define a binary consistency metric that equals 1 when the model's stepwise decision matches the optimal decision, and 0 otherwise. We employ two LPACR variants from our ablation studies: the base model trained via our proposed MDT strategy, and the w/o MDT model trained exclusively on uniformly distributed instances. Both models are assessed on TSP-100 testing datasets sampled from cluster and mixed distributions.

To visualize how structural variations influence neural predictions, we project these consistency scores onto the indicator space, using the local structural characteristics of the anchor node (i.e., the last visited node at each decoding step) as the spatial coordinates. Consistency scores corresponding to sub-problems with similar structures are aggregated and averaged to generate a spatial consistency heatmap. Darker shades indicate lower average prediction consistency, highlighting structural

regimes where the model exhibits performance degradation. Pure white regions denote structural patterns absent from the dataset, and thus lack evaluation data.

As shown in Figures 2e - 2h, the model trained exclusively on uniform data exhibits pronounced consistency degradation on local structures characterized by lower mean distance and coefficient of variation, which deviate from the dominant patterns in the uniform distribution. In contrast, the base model trained via the MDT strategy maintains higher consistency with ground-truth, aligning with its superior cross-distribution generalization performance.

G Details of Baseline Methods

In this section, we present detailed descriptions and configurations of the baseline methods used in the comparative experiments.

Classical OR Solvers We employ Concorde (Applegate et al., 2011), LKH3 (Helsgaun, 2017), and HGS (Vidal, 2022) for this category. These solvers are well-recognized benchmarks for VRPs.

Concorde is an exact solver capable of finding the optimal solution for a given TSP instance. Since its execution time becomes prohibitive on large-scale instances, we restrict it to small-scale TSP instances ($N \leq 500$). We execute it through PyConcorde with default hyperparameters.

LKH3 (Helsgaun, 2017) is an effective implementation of the Lin-Kernighan heuristic (Helsgaun, 2000; Lin & Kernighan, 1973), widely adopted as a core baseline in VRP research. We execute it through the PyLKH Python wrapper. For TSP instances, we use its default hyperparameters, where the number of trials equals the problem scale N, and the number of runs is 10. For large-scale CVRP instances ($N \geq 1K$), the number of runs is reduced to 1 for efficiency. Nevertheless, solving a single instance still requires several hours, and the solver occasionally fails to return feasible solutions.

HGS implements a hybrid genetic search algorithm for effective local search refinement of CVRP solutions. It can find a better solution with more iterations, but at the cost of increased runtime. We use the default number of 20K iterations for solving small-scale CVRP instances, while reducing this number to 1K for large-scale instances where $N \geq 1K$. Notably, we use 2K iterations when labeling CVRP training samples to balance solution quality and computational efficiency. We execute it through the PyHygese Python wrapper.

All these classical solvers are executed in parallel on CPUs using Python's Multiprocessing library.

Constructive Neural Solvers Methods in this category learn to make sequential decisions to construct solutions from scratch, and our proposed method also belongs to this category. We adopt several recent neural solvers with state-of-the-art performance for comparison, and these methods can be further categorized into subgroups as follows.

AMDKD (Bi et al., 2022) and OmniVRP (Zhou et al., 2023) inherit a similar model architecture from POMO (Kwon et al., 2020), aiming to enhance cross-distribution generalization through specialized training strategies. AMDKD trains teacher models on multiple exemplar distributions and leverages the knowledge distillation paradigm to aggregate their diverse knowledge, yielding a generalist student model. OmniVRP proposes a generic meta-learning framework to train its model on diverse tasks, employing a hierarchical scheduler to adaptively select suitable subsets from the training task set. We adopt the FOMAML version of OmniVRP as the baseline.

However, AMDKD and OmniVRP face severe cross-scale generalization challenges because their policies operate directly on global node features, causing significant performance degradation when the scale exceeds 500 nodes. Therefore, we only report their performance on small-scale synthetic datasets and real-world benchmarks.

LEHD (Luo et al., 2023) enhances solution quality through its heavy-decoder architecture, which calculates dynamic embeddings to better capture inherent relationships among contextual representations. However, as a global policy, it remains fundamentally unable to address cross-scale generalization, incurring prohibitive computational costs and exhibiting shifted prediction patterns on large-scale instances (Fang et al., 2024; Chen et al., 2025c).

SIL (Luo et al., 2025) employs a lightweight cross-attention mechanism with linear complexity, and it is trained via a self-improved training algorithm. We adopt its checkpoints trained on TSP-1K and

CVRP-1K instances in our comparative experiments. However, it sets the capacity to 250 for CVRP-1K, and its performance degrades significantly on our synthetic CVRP instances, where the capacity is set to 50 for all scales. Therefore, for CVRP testing datasets, we only report its performance on synthetic TSP datasets and real-world benchmarks.

TTPL (Chen et al., 2026) extends LEHD by restricting stepwise context to k-NN candidate nodes and applying projection to normalize node coordinates. It addresses the scalability issue and significantly improves cross-scale generalization performance on large-scale instances.

INViT (Fang et al., 2024) adopts multiple local policies with distinct receptive field ranges, representing distinct strategies for strengthening local policies. We adopt INViT-4V as the baseline method because it yields the best solution quality among all its variants.

L2R (Zhou et al., 2026) learns to adaptively prioritize nodes by extracting patterns from problem-specific features to prune the search space at each decoding step, enabling efficient and scalable solution construction.

ELG (Gao et al., 2024) presents an early attempt to design a hybrid policy, which employs an ensemble model comprising both global and local policies. However, the global policy operates over all nodes and introduces cross-scale generalization risk, thus compromising its practical performance.

GELD (Xiao et al., 2025a) partitions nodes into grids and uses a region-average linear attention mechanism to fuse global information into node embeddings. Subsequently, the decoder autoregressively selects the most promising node within a local selection range. Nevertheless, its implementation is limited to TSP and cannot solve CVRP instances.

DGL (Xiao et al., 2025b) incorporates global information into the features of local candidate nodes, providing enriched contextual understanding and thus better capturing the relationship between the current state and the broader problem context. However, its coarse aggregation mechanisms, such as the average distance between the candidate node and all unvisited nodes, lead to loss of critical spatial distribution information, which is crucial for decision-making.

The Fairness of Comparisons

A natural concern is that several baseline methods are trained exclusively on the uniform distribution, whereas LPACR is trained with our proposed MDT strategy, which may raise fairness concerns. We clarify this point as follows.

First, on synthetic datasets sampled from the uniform distribution, LPACR consistently outperforms all neural baseline methods. This is important because the uniform distribution is precisely the training distribution of most baselines, and therefore constitutes the most favorable setting for them.

Second, the MDT strategy is one of the core technical contributions of this work. We identify local topological diversity as a critical factor for cross-distribution generalization and propose the MDT strategy to systematically encompass heterogeneous local patterns in training samples. The comparison against single-distribution baselines is therefore not an unfair advantage, but rather a direct reflection of the methodological difference we aim to evaluate: whether explicitly promoting local topological diversity improves out-of-distribution generalization.

More importantly, our ablation study isolates the effect of MDT. The "w/o MDT" variant is trained only on uniformly distributed instances and uses the same architecture and training pipeline as LPACR. As shown in Tables 3, 6, and 7, removing MDT leads to consistent performance degradation on cluster, mixed, and unseen distributions, while the full LPACR remains stronger across both uniform and non-uniform subsets. This controlled comparison confirms that the gain from MDT is genuine and not confounded by other factors.

Finally, we note that all methods are evaluated under the same inference protocol, and on the same testing datasets covering diverse scales and distributions. The baselines are run with their original checkpoints, and the comparison reflects the performance of each method under its intended training paradigm. We believe this is a meaningful and fair evaluation of cross-scale and cross-distribution generalization, which is precisely the problem this paper targets.

H COMPARATIVE RESULTS ON SMALL-SCALE SYNTHETIC DATASETS

Table 4: Comparative results on small-scale synthetic TSP datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}	T_{avg}
TSP-100	Concorde	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.3s
	LKH3	0.23%±0.04%	0.15%±0.02%	0.22%±0.03%	0.28%±0.05%	0.16%±0.03%	0.22%±0.04%	0.25%±0.04%	0.1s
	AMDKD	2.39%±0.15%	2.23%±0.12%	2.26%±0.12%	3.22%±0.15%	2.27%±0.19%	3.15%±0.23%	2.19%±0.11%	0.1s
	OmniVRP	4.37%±0.21%	4.76%±0.20%	4.95%±0.21%	5.21%±0.29%	4.25%±0.24%	4.55%±0.30%	3.62%±0.17%	0.1s
	SIL	4.83%±0.34%	4.79%±0.39%	6.32%±0.45%	8.10%±0.40%	5.66%±0.59%	7.70%±0.50%	5.30%±0.38%	0.6s
	LEHD	4.57%±0.25%	4.24%±0.26%	4.34%±0.22%	4.43%±0.27%	4.14%±0.21%	4.48%±0.26%	4.37%±0.23%	0.5s
	TTPL	1.98%±0.16%	2.76%±0.25%	2.08%±0.19%	6.88%±0.31%	2.01%±0.21%	3.42%±0.34%	1.85%±0.16%	0.5s
	INViT	4.78%±0.18%	7.05%±0.32%	5.28%±0.19%	7.91%±0.30%	5.35%±0.28%	6.49%±0.31%	4.78%±0.19%	6.5s
	L2R	2.80%±0.14%	4.70%±0.29%	4.03%±0.20%	6.90%±0.29%	3.96%±0.27%	4.11%±0.31%	2.84%±0.15%	0.7s
	ELG	1.78%±0.10%	2.40%±0.11%	1.97%±0.11%	3.21%±0.24%	1.66%±0.10%	2.15%±0.17%	1.67%±0.09%	0.2s
	GELD	0.94%±0.09%	3.26%±0.37%	1.04%±0.11%	6.92%±0.35%	1.68%±0.21%	2.21%±0.25%	0.73%±0.08%	0.8s
	DGL	2.08%±0.15%	4.51%±0.38%	3.17%±0.21%	6.95%±0.32%	3.00%±0.24%	3.66%±0.35%	1.77%±0.13%	0.4s
	LPACR	**0.49%**±0.05%	**0.48%**±0.04%	**0.45%**±0.04%	**0.68%**±0.06%	**0.41%**±0.05%	**0.48%**±0.04%	**0.45%**±0.04%	1.3s
TSP-300	Concorde	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	3.0s
	LKH3	0.76%±0.04%	0.56%±0.04%	0.65%±0.04%	0.80%±0.08%	0.61%±0.04%	0.74%±0.07%	0.62%±0.03%	0.2s
	AMDKD	13.48%±0.20%	15.70%±0.32%	14.17%±0.25%	15.09%±0.26%	14.41%±0.31%	14.01%±0.32%	12.99%±0.20%	0.2s
	OmniVRP	6.48%±0.12%	6.30%±0.16%	6.99%±0.13%	7.29%±0.17%	7.06%±0.18%	6.98%±0.20%	6.28%±0.13%	0.2s
	LEHD	3.41%±0.10%	4.52%±0.16%	3.85%±0.10%	4.78%±0.15%	4.24%±0.13%	4.14%±0.12%	3.70%±0.11%	0.9s
	SIL	2.33%±0.17%	9.92%±0.39%	3.63%±0.28%	5.90%±0.24%	4.29%±0.34%	5.44%±0.35%	1.94%±0.13%	1.7s
	TTPL	2.10%±0.15%	8.60%±0.55%	2.79%±0.23%	4.94%±0.21%	3.82%±0.28%	3.47%±0.27%	1.91%±0.14%	1.8s
	INViT	5.79%±0.13%	14.14%±0.38%	8.02%±0.21%	9.55%±0.30%	8.81%±0.39%	8.90%±0.35%	5.42%±0.12%	19.7s
	L2R	3.91%±0.11%	11.51%±0.37%	5.89%±0.19%	8.17%±0.30%	6.84%±0.39%	6.27%±0.35%	3.48%±0.10%	2.1s
	ELG	8.31%±0.23%	9.23%±0.34%	8.94%±0.28%	10.13%±0.34%	8.33%±0.27%	8.79%±0.31%	7.80%±0.27%	0.6s
	GELD	1.98%±0.13%	11.23%±0.43%	3.10%±0.21%	5.87%±0.22%	3.44%±0.23%	4.77%±0.37%	1.99%±0.18%	1.6s
	DGL	3.84%±0.22%	10.97%±0.38%	5.43%±0.27%	7.43%±0.27%	5.60%±0.34%	5.98%±0.36%	3.29%±0.17%	1.4s
	LPACR	**0.84%**±0.06%	**1.24%**±0.13%	**0.79%**±0.04%	**1.68%**±0.09%	**0.81%**±0.06%	**1.07%**±0.07%	**0.69%**±0.04%	1.5s
TSP-500	Concorde	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	15.4s
	LKH3	0.93%±0.03%	0.99%±0.08%	0.85%±0.04%	0.86%±0.06%	0.83%±0.04%	0.92%±0.04%	0.79%±0.03%	0.4s
	AMDKD	23.62%±0.20%	26.38%±0.36%	24.29%±0.26%	25.73%±0.29%	24.39%±0.41%	24.89%±0.28%	21.85%±0.15%	0.3s
	OmniVRP	11.62%±0.14%	10.65%±0.23%	11.97%±0.21%	12.65%±0.22%	12.49%±0.22%	11.88%±0.24%	10.54%±0.12%	0.3s
	LEHD	3.81%±0.08%	5.00%±0.12%	3.86%±0.09%	5.25%±0.14%	4.40%±0.11%	4.94%±0.45%	3.63%±0.09%	1.5s
	SIL	2.08%±0.14%	10.00%±0.36%	2.81%±0.18%	5.73%±0.20%	3.69%±0.31%	5.22%±0.27%	1.73%±0.10%	3.2s
	TTPL	2.19%±0.12%	14.10%±0.66%	3.45%±0.22%	5.07%±0.22%	4.62%±0.33%	3.55%±0.23%	2.10%±0.14%	2.1s
	INViT	6.16%±0.11%	16.51%±0.43%	9.23%±0.20%	10.54%±0.31%	10.85%±0.49%	8.98%±0.25%	5.54%±0.10%	33.1s
	L2R	4.27%±0.09%	14.91%±0.41%	6.69%±0.17%	8.49%±0.30%	8.25%±0.50%	6.57%±0.25%	3.66%±0.09%	3.5s
	ELG	13.41%±0.31%	12.44%±0.40%	12.87%±0.28%	13.97%±0.37%	13.23%±0.30%	13.23%±0.55%	11.61%±0.29%	0.9s
	GELD	2.70%±0.21%	12.43%±0.39%	3.20%±0.15%	5.86%±0.19%	3.52%±0.23%	5.05%±0.25%	2.29%±0.16%	2.7s
	DGL	4.45%±0.18%	12.63%±0.42%	6.86%±0.26%	8.34%±0.27%	8.01%±0.43%	6.27%±0.28%	4.06%±0.19%	2.1s
	LPACR	**0.95%**±0.04%	**2.75%**±0.18%	**1.07%**±0.04%	**2.32%**±0.13%	**1.10%**±0.07%	**1.45%**±0.09%	**0.90%**±0.04%	2.5s

The bold value indicates that the method achieves the lowest average reference gap among all evaluated neural solvers.

Table 5: Comparative results on small-scale synthetic CVRP datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}	T_{avg}
CVRP-100	HGS	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	24.0s
	LKH3	2.10%±0.10%	1.68%±0.09%	1.99%±0.11%	1.99%±0.15%	2.06%±0.10%	2.43%±0.16%	2.05%±0.09%	0.5s
	AMDKD	5.06%±0.14%	4.04%±0.15%	4.47%±0.14%	5.01%±0.19%	4.57%±0.13%	5.19%±0.18%	4.78%±0.16%	0.1s
	OmniVRP	6.89%±0.17%	5.30%±0.16%	6.62%±0.19%	6.72%±0.24%	7.14%±0.17%	6.97%±0.19%	6.92%±0.18%	0.1s
	LEHD	4.57%±0.25%	4.24%±0.26%	4.34%±0.22%	4.43%±0.27%	4.14%±0.21%	4.48%±0.26%	4.37%±0.23%	0.9s
	TTPL	3.78%±0.22%	4.29%±0.27%	4.34%±0.21%	4.47%±0.25%	4.10%±0.21%	4.81%±0.30%	4.04%±0.20%	0.5s
	INViT	9.04%±0.20%	8.87%±0.24%	8.80%±0.18%	9.98%±0.31%	10.08%±0.23%	9.58%±0.23%	8.80%±0.17%	1.4s
	L2R	9.13%±0.32%	8.45%±0.29%	9.66%±0.32%	9.15%±0.32%	10.50%±0.39%	8.98%±0.26%	8.49%±0.27%	0.8s
	ELG	5.56%±0.15%	8.16%±0.52%	5.26%±0.15%	6.74%±0.27%	6.09%±0.19%	5.68%±0.19%	5.26%±0.13%	0.6s
	DGL	18.80%±0.48%	14.85%±0.46%	18.49%±0.40%	16.36%±0.50%	19.21%±0.44%	18.04%±0.51%	19.31%±0.46%	0.6s
	LPACR	**1.80%**±0.08%	**1.63%**±0.09%	**1.72%**±0.08%	**2.04%**±0.12%	**1.86%**±0.09%	**1.80%**±0.09%	**1.77%**±0.07%	1.4s
CVRP-300	HGS	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	3.6m
	LKH3	3.18%±0.08%	2.03%±0.11%	3.18%±0.09%	3.55%±0.12%	3.26%±0.09%	3.65%±0.10%	3.22%±0.07%	4.1s
	AMDKD	22.88%±0.31%	14.89%±0.41%	20.55%±0.40%	22.30%±0.41%	22.89%±0.41%	22.04%±0.37%	22.96%±0.32%	0.3s
	OmniVRP	7.19%±0.13%	4.32%±0.11%	6.61%±0.12%	6.45%±0.14%	7.17%±0.15%	7.16%±0.13%	7.00%±0.11%	0.3s
	LEHD	3.41%±0.10%	4.52%±0.16%	3.85%±0.10%	4.78%±0.15%	4.24%±0.13%	4.14%±0.12%	3.70%±0.11%	1.0s
	TTPL	3.81%±0.11%	4.60%±0.16%	4.13%±0.12%	4.53%±0.15%	4.48%±0.16%	4.31%±0.13%	3.73%±0.12%	2.3s
	INViT	10.86%±0.15%	9.79%±0.20%	11.06%±0.18%	10.54%±0.17%	11.35%±0.14%	10.74%±0.15%	10.51%±0.13%	4.3s
	L2R	14.17%±0.24%	8.05%±0.21%	12.73%±0.22%	11.38%±0.21%	13.75%±0.24%	12.56%±0.19%	13.17%±0.25%	2.5s
	ELG	7.50%±0.12%	10.77%±0.86%	7.38%±0.13%	8.28%±0.21%	8.29%±0.18%	7.34%±0.12%	7.35%±0.10%	1.7s
	DGL	16.30%±0.30%	11.36%±0.29%	15.59%±0.30%	13.91%±0.28%	15.53%±0.28%	15.67%±0.31%	16.28%±0.28%	1.9s
	LPACR	**1.75%**±0.05%	**1.86%**±0.09%	**1.99%**±0.06%	**2.13%**±0.07%	**2.36%**±0.10%	**2.04%**±0.07%	**1.81%**±0.04%	1.7s
CVRP-500	HGS	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	0.00%±0.00%	14.0m
	LKH3	3.56%±0.09%	2.68%±0.10%	3.33%±0.10%	3.95%±0.12%	3.71%±0.11%	3.73%±0.10%	3.53%±0.09%	12.9s
	AMDKD	41.36%±0.68%	27.49%±0.72%	38.05%±0.81%	40.03%±0.93%	40.94%±0.78%	39.67%±0.85%	40.78%±0.67%	0.5s
	OmniVRP	8.70%±0.15%	4.74%±0.11%	7.77%±0.13%	7.68%±0.17%	8.34%±0.15%	8.13%±0.13%	8.49%±0.13%	0.5s
	LEHD	3.81%±0.08%	5.00%±0.12%	3.86%±0.09%	5.25%±0.14%	4.40%±0.11%	4.94%±0.45%	3.63%±0.09%	1.7s
	TTPL	3.49%±0.08%	4.89%±0.13%	4.05%±0.09%	4.95%±0.12%	4.51%±0.11%	4.05%±0.10%	3.39%±0.08%	2.4s
	INViT	11.21%±0.13%	9.83%±0.16%	11.05%±0.15%	10.55%±0.15%	10.92%±0.12%	10.78%±0.13%	10.63%±0.13%	7.1s
	L2R	14.26%±0.22%	7.52%±0.17%	12.78%±0.20%	12.10%±0.21%	13.76%±0.22%	12.47%±0.23%	13.61%±0.20%	4.2s
	ELG	8.49%±0.13%	11.02%±0.50%	7.86%±0.14%	9.89%±0.50%	8.97%±0.21%	8.37%±0.14%	8.20%±0.12%	2.5s
	DGL	14.57%±0.25%	9.38%±0.22%	13.35%±0.22%	12.40%±0.25%	13.64%±0.24%	13.30%±0.23%	14.29%±0.23%	3.2s
	LPACR	**1.62%**±0.04%	**1.93%**±0.07%	**1.91%**±0.06%	**2.08%**±0.07%	**2.04%**±0.07%	**1.90%**±0.06%	**1.60%**±0.03%	2.8s

The bold value indicates that the method achieves the lowest average reference gap among all evaluated neural solvers.

I Ablation Study Results on Synthetic datasets

Table 6: Ablation study results on synthetic TSP datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}	T_{avg}
TSP-100	**LPACR**	0.49%±0.05%	0.48%±0.04%	0.45%±0.04%	**0.68%**±0.06%	0.41%±0.05%	**0.48%**±0.04%	0.45%±0.04%	1.3s
	w/o GDC	0.68%±0.08%	1.00%±0.11%	0.78%±0.08%	2.59%±0.22%	0.84%±0.13%	0.94%±0.15%	0.59%±0.06%	0.7s
	w/o MDT	**0.38%**±0.04%	2.22%±0.24%	0.43%±0.04%	3.31%±0.29%	0.46%±0.08%	0.82%±0.11%	**0.36%**±0.04%	1.3s
	w/ SL+	0.48%±0.08%	**0.41%**±0.04%	**0.33%**±0.04%	**0.68%**±0.09%	**0.40%**±0.06%	0.52%±0.07%	0.41%±0.06%	1.3s
TSP-300	**LPACR**	0.84%±0.06%	1.24%±0.13%	0.79%±0.04%	1.68%±0.09%	**0.81%**±0.06%	**1.07%**±0.07%	**0.69%**±0.04%	1.6s
	w/o GDC	1.30%±0.09%	3.55%±0.27%	1.87%±0.16%	3.67%±0.20%	2.12%±0.16%	3.29%±0.35%	1.38%±0.12%	1.3s
	w/o MDT	0.87%±0.04%	6.92%±0.32%	1.05%±0.05%	4.53%±0.22%	1.23%±0.11%	2.39%±0.25%	0.80%±0.05%	1.6s
	w/ SL+	**0.73%**±0.04%	**0.98%**±0.12%	**0.77%**±0.06%	**1.67%**±0.11%	0.84%±0.08%	1.10%±0.11%	0.70%±0.04%	1.6s
TSP-500	**LPACR**	**0.95%**±0.04%	2.75%±0.18%	**1.07%**±0.04%	2.32%±0.13%	1.10%±0.07%	**1.45%**±0.09%	0.90%±0.04%	2.6s
	w/o GDC	1.89%±0.14%	5.77%±0.33%	2.29%±0.16%	3.73%±0.17%	2.85%±0.19%	2.74%±0.17%	1.83%±0.13%	2.1s
	w/o MDT	1.27%±0.08%	10.94%±0.39%	1.89%±0.07%	5.01%±0.18%	2.14%±0.15%	2.59%±0.14%	1.07%±0.04%	2.5s
	w/ SL+	1.06%±0.06%	**2.66%**±0.21%	1.08%±0.07%	**2.18%**±0.12%	**1.07%**±0.08%	1.51%±0.09%	**0.87%**±0.05%	2.6s
TSP-1K	**LPACR**	**0.20%**±0.06%	**2.99%**±0.31%	**0.46%**±0.06%	**0.93%**±0.11%	0.57%±0.11%	**0.57%**±0.08%	**0.09%**±0.05%	5.1s
	w/o GDC	1.00%±0.10%	5.78%±0.40%	1.51%±0.12%	2.25%±0.15%	2.38%±0.15%	2.50%±0.26%	1.43%±0.14%	4.2s
	w/o MDT	1.03%±0.11%	12.83%±0.46%	2.35%±0.09%	3.60%±0.16%	2.45%±0.17%	2.53%±0.17%	0.93%±0.08%	5.1s
	w/ SL+	0.23%±0.06%	3.31%±0.29%	0.61%±0.07%	1.34%±0.17%	**0.47%**±0.12%	0.95%±0.11%	0.21%±0.06%	5.4s
TSP-5K	**LPACR**	**0.92%**±0.07%	**4.46%**±0.27%	**1.79%**±0.09%	**1.85%**±0.14%	**1.98%**±0.16%	**1.35%**±0.10%	1.08%±0.07%	25.6s
	w/o GDC	1.73%±0.09%	7.25%±0.29%	2.40%±0.12%	2.53%±0.13%	3.28%±0.15%	2.60%±0.15%	1.71%±0.09%	20.9s
	w/o MDT	3.29%±0.10%	15.89%±0.33%	5.63%±0.10%	5.60%±0.16%	5.89%±0.21%	4.64%±0.13%	3.00%±0.08%	25.2s
	w/ SL+	1.21%±0.09%	8.52%±0.43%	2.09%±0.08%	2.48%±0.15%	2.29%±0.16%	1.64%±0.12%	**1.05%**±0.07%	26.4s
TSP-10K	**LPACR**	**1.26%**±0.06%	**7.16%**±0.29%	**2.40%**±0.08%	**2.23%**±0.12%	**2.67%**±0.13%	**1.24%**±0.09%	**1.15%**±0.06%	49.1s
	w/o GDC	1.82%±0.07%	9.09%±0.32%	2.90%±0.12%	2.71%±0.10%	3.65%±0.14%	2.21%±0.11%	1.78%±0.07%	41.7s
	w/o MDT	4.35%±0.09%	17.71%±0.27%	7.70%±0.13%	6.14%±0.15%	7.16%±0.17%	5.35%±0.12%	4.00%±0.08%	51.6s
	w/ SL+	1.57%±0.08%	11.75%±0.32%	3.22%±0.12%	2.87%±0.13%	3.10%±0.14%	1.94%±0.10%	1.41%±0.07%	53.3s

The bold value indicates that the method achieves the lowest average relative gap among all evaluated configurations.

Table 7: Ablation study results on synthetic CVRP datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}	T_{avg}
CVRP-100	**LPACR**	**1.80%**±0.08%	**1.63%**±0.09%	**1.72%**±0.08%	**2.04%**±0.12%	**1.86%**±0.09%	**1.80%**±0.09%	**1.77%**±0.07%	1.4s
	w/o GDC	2.52%±0.11%	2.36%±0.13%	2.30%±0.10%	2.44%±0.13%	2.54%±0.11%	2.67%±0.11%	2.50%±0.11%	1.1s
	w/o MDT	1.87%±0.08%	1.97%±0.12%	1.97%±0.09%	2.30%±0.12%	1.98%±0.09%	2.08%±0.11%	1.83%±0.08%	1.4s
	w/ SL+	3.35%±0.22%	3.77%±0.25%	3.55%±0.22%	4.96%±0.37%	3.96%±0.26%	3.50%±0.25%	3.10%±0.17%	1.4s
CVRP-300	**LPACR**	**1.75%**±0.05%	**1.86%**±0.09%	**1.99%**±0.06%	**2.13%**±0.07%	**2.36%**±0.10%	**2.04%**±0.07%	**1.81%**±0.04%	1.7s
	w/o GDC	2.47%±0.07%	2.65%±0.11%	2.68%±0.08%	2.73%±0.10%	2.99%±0.10%	2.67%±0.09%	2.40%±0.05%	1.4s
	w/o MDT	2.12%±0.06%	3.19%±0.14%	2.52%±0.08%	3.53%±0.13%	2.89%±0.12%	2.54%±0.09%	2.06%±0.06%	1.8s
	w/ SL+	3.64%±0.15%	5.63%±0.27%	3.85%±0.15%	5.29%±0.28%	3.99%±0.17%	4.31%±0.37%	3.45%±0.13%	1.7s
CVRP-500	**LPACR**	**1.62%**±0.04%	**1.93%**±0.07%	**1.91%**±0.06%	**2.08%**±0.07%	**2.04%**±0.07%	**1.90%**±0.06%	**1.60%**±0.03%	2.8s
	w/o GDC	2.29%±0.05%	2.81%±0.09%	2.57%±0.06%	2.59%±0.08%	2.73%±0.08%	2.48%±0.06%	2.12%±0.05%	2.4s
	w/o MDT	2.08%±0.05%	3.74%±0.11%	2.50%±0.07%	3.86%±0.15%	2.91%±0.12%	2.64%±0.10%	1.97%±0.04%	3.0s
	w/ SL+	3.17%±0.10%	5.42%±0.23%	3.32%±0.09%	4.24%±0.15%	3.93%±0.15%	4.00%±0.18%	3.11%±0.10%	2.8s
CVRP-1K	**LPACR**	**-0.08%**±0.05%	**1.09%**±0.05%	**0.61%**±0.07%	**0.48%**±0.06%	**0.50%**±0.06%	**0.26%**±0.06%	**0.02%**±0.05%	5.7s
	w/o GDC	0.50%±0.06%	1.97%±0.07%	1.26%±0.08%	0.93%±0.06%	1.16%±0.08%	0.87%±0.06%	0.64%±0.06%	4.7s
	w/o MDT	0.36%±0.07%	2.54%±0.09%	1.34%±0.07%	1.86%±0.12%	1.56%±0.11%	0.97%±0.08%	0.51%±0.06%	6.0s
	w/ SL+	1.09%±0.08%	4.35%±0.17%	1.96%±0.09%	2.23%±0.11%	1.96%±0.10%	1.86%±0.12%	1.16%±0.08%	5.7s
CVRP-5K	**LPACR**	**-1.45%**±0.01%	**-0.74%**±0.03%	**-1.03%**±0.02%	**-1.33%**±0.03%	**-1.16%**±0.03%	**-1.42%**±0.04%	**-1.39%**±0.01%	28.2s
	w/o GDC	-0.70%±0.04%	0.04%±0.04%	-0.41%±0.04%	-0.81%±0.04%	-0.56%±0.04%	-0.70%±0.05%	-0.70%±0.04%	24.0s
	w/o MDT	-0.87%±0.06%	0.05%±0.05%	-0.44%±0.05%	-0.44%±0.06%	-0.33%±0.06%	-0.87%±0.05%	-0.87%±0.04%	29.2s
	w/ SL+	-0.76%±0.02%	1.60%±0.08%	-0.35%±0.03%	-0.15%±0.07%	-0.28%±0.04%	-0.14%±0.08%	-0.71%±0.02%	28.5s
CVRP-10K	**LPACR**	**-1.51%**±0.01%	**-0.97%**±0.03%	**-1.12%**±0.01%	**-1.52%**±0.03%	**-1.31%**±0.02%	**-1.49%**±0.03%	**-1.46%**±0.01%	55.2s
	w/o GDC	-0.68%±0.03%	-0.07%±0.04%	-0.37%±0.04%	-0.90%±0.03%	-0.69%±0.04%	-0.67%±0.05%	-0.71%±0.03%	47.8s
	w/o MDT	-1.01%±0.04%	-0.35%±0.04%	-0.62%±0.04%	-0.80%±0.04%	-0.61%±0.05%	-1.00%±0.04%	-0.95%±0.04%	60.1s
	w/ SL+	-0.93%±0.01%	0.98%±0.07%	-0.52%±0.02%	-0.68%±0.05%	-0.53%±0.04%	-0.55%±0.06%	-0.89%±0.02%	55.9s

The bold value indicates that the method achieves the lowest average relative gap among all evaluated configurations.

J SENSITIVITY ANALYSIS OF GRID RESOLUTION

In this section, we conduct a sensitivity analysis on the key hyperparameter, grid resolution G, to understand its impact on LPACR's performance. In the inference phase, we compare the default $G = 6 \times 6$ with varying grid resolutions including $G = 4 \times 4$, $G = 8 \times 8$, and $G = 10 \times 10$. These experiments use the same model trained using $G = 6 \times 6$. In addition, we train a model using dynamic grid resolutions uniformly sampled from $[4, 10]$. This dynamic variant uses the fixed $G = 6 \times 6$ setting during inference for a fair comparison with other variants.

As presented in Tables 8, 9, and 10, LPACR achieves robust performance across different grid resolution configurations on extensive TSP instances, and the default $G = 6 \times 6$ yields the best performance on most evaluated subsets. The $G = 10 \times 10$ configuration exhibits larger relative gaps since this setting deviates more from the training configuration.

As for CVRP instances, the default $G = 6 \times 6$ configuration delivers competitive results, though there exist greater performance variations when using different grid resolutions. The dynamic variant achieves the best performance on several real-world benchmarks, but its performance on synthetic datasets is inferior.

Overall, LPACR is robust across different grid resolutions, and the default $G = 6 \times 6$ configuration yields the best results on most evaluated datasets.

Table 8: Sensitivity analysis results on synthetic TSP datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}
TSP-100	4×4	0.54%±0.05%	0.59%±0.06%	0.46%±0.04%	1.16%±0.13%	0.51%±0.08%	0.61%±0.07%	0.48%±0.05%
	6×6	**0.49%**±0.05%	**0.48%**±0.04%	**0.45%**±0.04%	0.68%±0.06%	**0.41%**±0.05%	**0.48%**±0.04%	**0.45%**±0.04%
	8×8	0.57%±0.05%	0.50%±0.05%	0.55%±0.05%	**0.67%**±0.06%	0.48%±0.05%	0.65%±0.10%	0.54%±0.05%
	10×10	0.80%±0.06%	0.63%±0.05%	0.70%±0.06%	0.87%±0.09%	0.64%±0.06%	0.97%±0.12%	0.75%±0.06%
	dynamic	0.62%±0.06%	0.65%±0.05%	0.52%±0.04%	0.94%±0.08%	0.56%±0.07%	0.74%±0.09%	0.47%±0.04%
TSP-300	4×4	0.85%±0.07%	1.86%±0.16%	1.06%±0.10%	2.78%±0.20%	**0.80%**±0.06%	1.39%±0.12%	0.76%±0.06%
	6×6	**0.84%**±0.06%	1.24%±0.13%	**0.79%**±0.04%	1.68%±0.09%	0.81%±0.06%	**1.07%**±0.07%	**0.69%**±0.04%
	8×8	0.85%±0.04%	**1.23%**±0.10%	0.91%±0.04%	**1.64%**±0.10%	0.89%±0.07%	1.18%±0.07%	0.70%±0.04%
	10×10	1.15%±0.04%	1.77%±0.15%	1.28%±0.07%	1.90%±0.11%	1.37%±0.10%	1.49%±0.09%	0.99%±0.06%
	dynamic	1.24%±0.06%	1.69%±0.14%	1.42%±0.09%	2.45%±0.13%	1.26%±0.08%	1.92%±0.16%	1.07%±0.06%
TSP-500	4×4	1.03%±0.05%	3.73%±0.22%	1.24%±0.07%	3.18%±0.16%	1.28%±0.11%	1.75%±0.11%	0.96%±0.05%
	6×6	**0.95%**±0.04%	**2.75%**±0.18%	**1.07%**±0.04%	2.32%±0.13%	**1.10%**±0.07%	1.45%±0.09%	**0.90%**±0.04%
	8×8	1.06%±0.04%	3.32%±0.22%	1.18%±0.04%	**2.25%**±0.12%	1.32%±0.08%	**1.41%**±0.07%	0.91%±0.04%
	10×10	1.33%±0.04%	3.98%±0.23%	1.61%±0.06%	2.69%±0.13%	1.66%±0.10%	1.69%±0.08%	1.17%±0.05%
	dynamic	1.84%±0.08%	4.15%±0.27%	2.19%±0.13%	3.56%±0.15%	2.19%±0.11%	2.92%±0.18%	1.68%±0.08%
TSP-1K	4×4	0.36%±0.10%	3.64%±0.35%	0.65%±0.08%	1.82%±0.17%	0.70%±0.13%	0.80%±0.10%	0.10%±0.05%
	6×6	**0.20%**±0.06%	**2.99%**±0.31%	**0.46%**±0.06%	**0.93%**±0.11%	**0.57%**±0.11%	**0.57%**±0.08%	**0.09%**±0.05%
	8×8	0.28%±0.05%	3.12%±0.27%	0.57%±0.06%	1.03%±0.15%	0.94%±0.13%	0.74%±0.09%	0.19%±0.05%
	10×10	0.56%±0.06%	4.36%±0.33%	1.05%±0.06%	1.34%±0.13%	1.23%±0.13%	1.15%±0.11%	0.56%±0.07%
	dynamic	1.53%±0.11%	6.40%±0.40%	2.28%±0.13%	3.17%±0.19%	2.22%±0.23%	2.67%±0.16%	1.34%±0.10%
TSP-5K	4×4	1.01%±0.08%	5.50%±0.29%	**1.78%**±0.09%	**1.72%**±0.14%	2.12%±0.16%	1.42%±0.10%	**0.92%**±0.07%
	6×6	**0.92%**±0.07%	**4.46%**±0.27%	1.79%±0.09%	1.85%±0.14%	**1.98%**±0.16%	**1.35%**±0.10%	1.08%±0.07%
	8×8	1.11%±0.08%	4.82%±0.29%	1.87%±0.08%	2.33%±0.16%	2.32%±0.16%	**1.35%**±0.10%	1.05%±0.07%
	10×10	1.42%±0.07%	5.40%±0.28%	2.47%±0.09%	2.58%±0.15%	2.76%±0.17%	1.56%±0.09%	1.14%±0.05%
	dynamic	2.52%±0.13%	12.10%±0.30%	5.97%±0.25%	5.46%±0.19%	5.71%±0.23%	3.81%±0.17%	2.22%±0.11%
TSP-10K	4×4	**1.05%**±0.06%	8.65%±0.29%	**2.34%**±0.08%	**1.97%**±0.11%	**2.66%**±0.14%	**1.17%**±0.07%	**1.02%**±0.05%
	6×6	1.27%±0.06%	**7.16%**±0.29%	2.40%±0.08%	2.23%±0.12%	2.67%±0.13%	1.24%±0.09%	1.15%±0.06%
	8×8	1.41%±0.07%	7.73%±0.31%	2.39%±0.07%	2.66%±0.14%	2.89%±0.14%	1.48%±0.09%	1.40%±0.07%
	10×10	1.76%±0.07%	8.48%±0.30%	3.02%±0.09%	3.23%±0.14%	3.65%±0.14%	1.64%±0.08%	1.65%±0.06%
	dynamic	2.55%±0.10%	13.24%±0.34%	7.53%±0.28%	5.18%±0.15%	6.42%±0.21%	3.80%±0.12%	2.49%±0.09%

The bold value indicates that the method achieves the lowest average reference gap among all evaluated configurations.

Table 9: Sensitivity analysis results on synthetic CVRP datasets.

	Method	G_{uni}	G_{cls}	G_{mix}	G_{epn}	G_{exp}	G_{imp}	G_{grd}
CVRP-100	4×4	3.86%±0.12%	3.22%±0.14%	3.53%±0.11%	3.96%±0.18%	3.81%±0.13%	4.09%±0.16%	3.81%±0.11%
	6×6	**1.80%**±0.08%	1.63%±0.09%	1.72%±0.08%	**2.04%**±0.12%	**1.86%**±0.09%	**1.80%**±0.09%	1.77%±0.07%
	8×8	4.53%±0.15%	3.61%±0.16%	4.60%±0.18%	4.87%±0.25%	4.71%±0.17%	4.81%±0.18%	4.73%±0.15%
	10×10	7.50%±0.24%	5.61%±0.21%	6.73%±0.18%	7.72%±0.30%	7.65%±0.25%	7.95%±0.25%	7.73%±0.21%
	dynamic	1.83%±0.08%	**1.55%**±0.09%	**1.71%**±0.08%	2.07%±0.13%	1.87%±0.08%	1.87%±0.10%	**1.75%**±0.07%
CVRP-300	4×4	2.66%±0.05%	2.46%±0.09%	2.82%±0.06%	3.09%±0.10%	3.01%±0.09%	3.07%±0.11%	2.72%±0.05%
	6×6	**1.75%**±0.05%	**1.86%**±0.09%	**1.99%**±0.06%	**2.13%**±0.07%	**2.36%**±0.10%	**2.04%**±0.07%	**1.81%**±0.04%
	8×8	3.54%±0.07%	3.08%±0.12%	3.88%±0.10%	3.77%±0.11%	4.19%±0.14%	3.97%±0.10%	3.54%±0.08%
	10×10	5.39%±0.10%	4.66%±0.16%	5.87%±0.13%	5.49%±0.14%	5.97%±0.14%	5.78%±0.12%	5.48%±0.09%
	dynamic	2.09%±0.05%	2.13%±0.09%	2.34%±0.07%	2.31%±0.08%	2.53%±0.10%	2.28%±0.07%	2.16%±0.06%
CVRP-500	4×4	2.27%±0.05%	2.20%±0.07%	2.38%±0.06%	2.69%±0.07%	2.63%±0.07%	2.56%±0.07%	2.13%±0.04%
	6×6	**1.62%**±0.04%	**1.93%**±0.07%	**1.91%**±0.06%	**2.08%**±0.07%	**2.04%**±0.07%	**1.90%**±0.06%	**1.60%**±0.03%
	8×8	3.11%±0.07%	2.98%±0.09%	3.45%±0.08%	3.30%±0.10%	3.52%±0.11%	3.22%±0.07%	3.06%±0.06%
	10×10	4.77%±0.09%	4.16%±0.11%	5.27%±0.11%	4.85%±0.11%	5.12%±0.11%	5.04%±0.10%	4.62%±0.09%
	dynamic	1.88%±0.04%	2.33%±0.10%	2.22%±0.06%	2.40%±0.08%	2.39%±0.08%	2.12%±0.06%	1.89%±0.05%
CVRP-1K	4×4	0.26%±0.06%	1.47%±0.05%	0.92%±0.07%	0.92%±0.06%	0.91%±0.07%	0.76%±0.10%	0.39%±0.05%
	6×6	**-0.08%**±0.05%	**1.09%**±0.05%	**0.61%**±0.07%	**0.48%**±0.06%	**0.50%**±0.06%	**0.26%**±0.06%	**0.02%**±0.05%
	8×8	0.87%±0.06%	1.82%±0.08%	1.90%±0.09%	1.32%±0.06%	1.60%±0.09%	1.26%±0.07%	0.96%±0.06%
	10×10	2.15%±0.09%	2.80%±0.08%	3.36%±0.10%	2.59%±0.08%	2.75%±0.09%	2.48%±0.10%	2.34%±0.09%
	dynamic	0.19%±0.06%	1.42%±0.06%	1.01%±0.07%	0.80%±0.06%	0.79%±0.07%	0.51%±0.06%	0.31%±0.06%
CVRP-5K	4×4	-1.27%±0.02%	-0.44%±0.04%	-0.90%±0.02%	-1.10%±0.03%	-0.99%±0.03%	-1.13%±0.05%	-1.23%±0.01%
	6×6	**-1.45%**±0.01%	**-0.74%**±0.03%	**-1.03%**±0.02%	**-1.33%**±0.03%	**-1.16%**±0.03%	**-1.42%**±0.04%	**-1.39%**±0.01%
	8×8	-1.01%±0.02%	-0.47%±0.03%	-0.47%±0.03%	-0.93%±0.04%	-0.67%±0.04%	-1.01%±0.04%	-0.95%±0.02%
	10×10	-0.28%±0.03%	0.01%±0.04%	0.37%±0.05%	-0.27%±0.05%	0.06%±0.05%	-0.33%±0.04%	-0.22%±0.03%
	dynamic	-1.17%±0.03%	-0.60%±0.03%	-0.68%±0.03%	-1.13%±0.03%	-0.94%±0.03%	-1.24%±0.04%	-1.12%±0.04%
CVRP-10K	4×4	-1.37%±0.01%	-0.65%±0.03%	-1.02%±0.01%	-1.33%±0.03%	-1.16%±0.02%	-1.30%±0.03%	-1.33%±0.01%
	6×6	**-1.51%**±0.01%	**-0.97%**±0.03%	**-1.12%**±0.01%	**-1.52%**±0.03%	**-1.31%**±0.02%	**-1.49%**±0.03%	**-1.46%**±0.01%
	8×8	-1.20%±0.01%	-0.84%±0.03%	-0.73%±0.02%	-1.24%±0.03%	-0.97%±0.03%	-1.20%±0.03%	-1.16%±0.01%
	10×10	-0.65%±0.02%	-0.53%±0.03%	-0.14%±0.02%	-0.76%±0.04%	-0.43%±0.03%	-0.67%±0.03%	-0.58%±0.02%
	dynamic	-1.28%±0.02%	-0.90%±0.02%	-0.86%±0.02%	-1.36%±0.03%	-1.15%±0.03%	-1.33%±0.03%	-1.24%±0.02%

The bold value indicates that the method achieves the lowest average reference gap among all evaluated configurations.

Table 10: Sensitivity analysis results on real-world benchmarks.

Method	TSPLIB	VLSI	National	set-X	set-XL	set-CMT	set-Golden	set-Li	set-AGS
4×4	2.3%±0.3%	**3.4%**±0.2%	**3.5%**±0.4%	8.5%±0.8%	10.7%±1.1%	2.0%±2.4%	5.6%±0.4%	9.0%±1.8%	25.2%±11.4%
6×6	**2.1%**±0.3%	3.5%±0.2%	3.7%±0.5%	7.5%±0.7%	11.0%±1.2%	-1.6%±2.5%	**4.7%**±0.5%	**6.9%**±1.2%	9.9%±2.0%
8×8	2.2%±0.3%	3.5%±0.2%	3.6%±0.4%	8.4%±0.7%	10.8%±0.8%	1.7%±2.4%	6.4%±0.6%	7.2%±.08%	14.4%±3.8%
10×10	2.3%±0.3%	3.9%±0.2%	4.1%±0.4%	9.5%±0.6%	12.1%±0.8%	7.0%±2.6%	11.1%±0.9%	12.6%±1.2%	13.6%±2.2%
random	2.9%±0.3%	4.5%±0.2%	7.1%±1.0%	**6.1%**±0.6%	**8.1%**±0.6%	**-2.4%**±2.5%	6.0%±0.8%	11.4%±2.0%	**9.5%**±1.9%

The bold value indicates that the method achieves the lowest average relative gap among all evaluated configurations.