Learning to Solve Compositional Geometry Routing Problems

Mingfeng Fan
National University of Singapore
ming.fan@nus.edu.sg

Jianan Zhou*
Nanyang Technological University
jianan004@e.ntu.edu.sg

Jiaqi Cheng*
Central South University
chengjq0705@gmail.com

Yifeng Zhang
National University of Singapore
yifeng@u.nus.edu

Jie Zhang
Nanyang Technological University
zhangj@ntu.edu.sg

Guillaume Adrien Sartoretti
National University of Singapore
guillaume.sartoretti@nus.edu.sg

Abstract

We study the Compositional Geometry Routing Problem (CGRP), a unified superclass of traditional routing problems that covers point-only, line-only, area-only, and arbitrary hybrid task geometries, providing a broad abstraction for real-world routing scenarios. Beyond standard point-based routing, CGRP with non-point tasks can be inherently asymmetric, tightly coupled travel routes with the intrinsic path, and enlarges the action space with numerous feasible yet often irrelevant options, thereby posing significant challenges for both representation learning and decision-making. To address these challenges, we propose DiCon, a differential attention–assisted solver with contrastive learning, as a plug-and-play framework that tackles the problem from two complementary angles. First, we introduce a differential attention mechanism that actively suppresses the probability mass on less competitive candidate actions. Second, we design a double-level contrastive learning objective to promote robust global instance representations and regularize geometry-aware task representations. Extensive experiments demonstrate that DiCon achieves strong performance, broad versatility, and superior generalization across diverse CGRP instances with different compositions.

1 Introduction

Routing problems have long served as fundamental models for spatial decision-making, with broad applications in logistics [47], robotic inspection [2], and disaster response [50]. Classical formulations usually assume a specific service geometry. For example, node-routing problems, such as the Traveling Salesman Problem (TSP), represent each task as a discrete point and optimize the visiting order among these points [15]. Line- or arc-routing problems focus on extended structures, such as roads, cracks, and power lines [42, 1]. Area-routing or coverage-routing problems further require robots to service spatial regions, such as fields, buildings, or search zones [44, 57]. However, many real-world missions cannot be naturally described by a single type of service geometry. Robotic inspection may require covering spatial regions while tracking crack-like linear defects [48]. Similarly, search-and-rescue, infrastructure maintenance, and environmental surveying may involve point tasks,

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

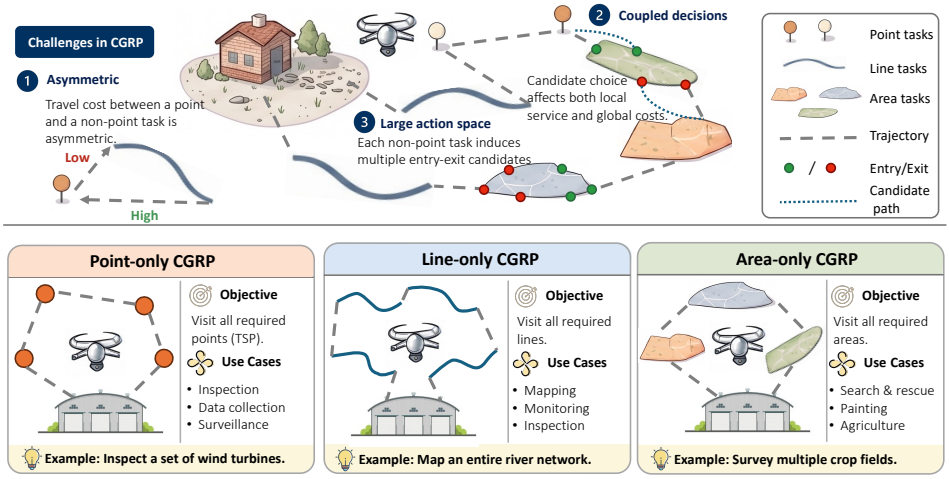


Figure 1: The characteristics of CGRP.

line tasks, area tasks, or arbitrary combinations of them [18, 60]. Treating these cases as separate routing problems limits the flexibility of solver design and often requires geometry-specific adaptation.

To provide a common abstraction for homogeneous and mixed-geometry routing, we study the Compositional Geometry Routing Problem (CGRP), a unified formulation over point, line, and area tasks in a two-dimensional workspace. CGRP subsumes several classical routing settings while allowing arbitrary mixtures of service geometries within a single instance. Unlike point tasks, non-point tasks impose intrinsic service requirements, such as traversing a line segment or covering a region with a feasible pattern, and often admit multiple valid entry–exit pairs. The unified nature of CGRP introduces several challenges beyond classical node-centric routing, as illustrated in Fig. 1. First, the problem with non-point tasks can become asymmetric, since the cost of traveling from a preceding task to a non-point task generally differs from the cost of traveling from that non-point task back to the preceding task. Second, route planning and intrinsic service paths are tightly coupled: selecting an entry–exit pair for a line or area task is both a local servicing decision and a global routing decision, because it changes the connection lengths to preceding and succeeding tasks. Third, the action space expands substantially, as each line or area task gives rise to multiple feasible candidates, many of which may be geometrically valid but strategically irrelevant.

While traditional exact and heuristic methods exist, they often require substantial domain knowledge, and efficiently finding near-optimal solutions remains challenging, particularly for large-scale instances [38, 31, 23, 11]. Recent neural combinatorial optimization (NCO) approaches have shown strong empirical performance on a variety of routing problems [5, 52]. However, most neural solvers are tailored to node-centric formulations with relatively structured action spaces. When applied to CGRP, they encounter two recurring challenges: 1) *Representation:* the model must encode task geometries and their service implications in a manner that generalizes across single-geometry instances, hybrid compositions, and diverse spatial layouts. 2) *Decision-making:* the model needs to assign probability mass over a substantially larger and noisier candidate set, where many options are geometrically feasible but strategically less competitive. These challenges are particularly pronounced under compositional generalization, where test instances present novel combinations of task types, scales, and geometric configurations that were not jointly observed during training.

To address these challenges, we introduce Differential attention-assisted solver with Contrastive learning (DiCon), a plug-and-play framework designed to augment neural routing backbones with enhanced representation learning and decision-making for compositional geometry tasks. DiCon comprises two complementary mechanisms. First, a differential attention (DA) mechanism is introduced to suppress probability mass assigned to less competitive candidate actions, thereby mitigating over-exploration of geometrically feasible yet strategically weak entry–exit choices. Second, we design a double-level contrastive learning (CL) objective that operates at two granularities: an instance level, which promotes robust global representations across diverse task compositions, and an intra-task level, which reinforces semantic coherence within each task while preserving representational diversity across different tasks. Our contributions are summarized as follows:

- We study CGRP, which more faithfully captures the complexity of diverse real-world applications. We formulate CGRP solving as a Markov Decision Process (MDP) and propose a plug-and-play framework that enables neural solvers to tackle CGRP effectively.
- We identify key challenges in neural routing backbones and propose a differential attention mechanism together with a double-level CL objective to enhance both representation learning and decision-making for compositional geometry tasks.
- We conduct experiments on diverse compositional geometry settings and integrate DiCon into different neural routing backbones. Empirical results show that DiCon achieves strong performance, broad versatility, and robust compositional generalization.

2 Related Work

2.1 Neural Routing Solver

Pointer Networks [49] pioneered neural heuristics for TSP by framing solution construction as a sequence-to-sequence prediction task with attention. While effective, this approach relies on costly supervised labels. [4] replaced supervision with reinforcement learning (RL), enabling direct optimization of tour length. The Attention Model (AM) [33] further advanced this line of work through a Transformer-style encoder–decoder that constructs solutions autoregressively. Exploiting the head–tail symmetry of solutions, POMO [34] introduced parallel rollouts with strong data augmentation, substantially outperforming AM and inspiring many subsequent studies [32, 25, 20, 17, 28]. On the other hand, several studies propose an alternative architectural paradigm, the Light Encoder-Heavy Decoder (LEHD) [13, 39], which is typically trained with supervised labels. To reduce the dependence on expensive labels, subsequent works [43, 40] train LEHD models without labels through self-improving learning. Beyond these advances, more recent efforts have increasingly focused on improving generalization to large-scale instances [37, 61, 26, 53, 58], robustness under distribution shifts [7], and broader problem variants [59, 6, 12].

In this paper, we focus on the above construction-based solvers and refer interested readers to a comprehensive survey [52]. Despite recent extensions, most neural routing solvers remain tailored to node-centric formulations, including asymmetric variants with direction-dependent pairwise costs [35, 36], arc routing problems that require servicing edges rather than nodes [29], and generalized TSP settings in which each cluster provides multiple candidate nodes [9]. In contrast, CGRP covers compositional geometry tasks, including point-only, line-only, area-only, and hybrid geometries. By incorporating non-point tasks with intrinsic service paths and multiple entry–exit candidates, CGRP defines a more challenging, general, and practically relevant problem setting.

2.2 Contrastive Learning for NCO

Contrastive learning (CL) is a self-supervised learning paradigm that acquires transferable representations by encouraging agreement between positive samples while pushing apart representations of negative samples, thereby exploiting structural similarity in unlabeled data. It has been extensively studied in computer vision [22, 46, 10] and natural language processing [41, 19], where it consistently improves representation quality and downstream generalization. A representative method is SimSiam [8], which adopts a simple Siamese architecture with asymmetric prediction and a stop-gradient operation to prevent representation collapse, avoiding the need for explicit negative pairs or large memory banks.

In NCO, CL has emerged as an effective self-supervised paradigm for enhancing representation learning and robustness, particularly under limited supervision or distribution shifts. [14] provided an early study, demonstrating that CL benefits combinatorial problems when augmentations preserve feasibility and cost structure, while naive perturbations can be detrimental. Building on this insight, [30] proposed a multi-view graph contrastive framework for vehicle routing, contrasting complementary instance views to promote invariant representations. Beyond routing, [27] introduced a contrastive predict-and-search framework for mixed-integer linear programs, aligning predictions with search trajectories to guide solution discovery. CL has also been integrated with RL: [56] combined contrastive objectives with policy learning to strengthen state representations and accelerate training. More recently, [55] scaled CL to a hierarchical pretraining regime using a multi-view graph Transformer, enabling transfer across diverse combinatorial optimization tasks. In addition, [21]

aligned multiple graph decision problems with their translated Boolean satisfiability forms through a cross-modal contrastive pretraining paradigm. In this paper, we design a double-level CL objective to enhance the representation learning for compositional geometry tasks.

3 Problem Statement

CGRP. The CGRP generalizes classical routing problems by covering point-only, line-only, area-only, and arbitrary hybrid task geometries within a unified formulation. Non-point tasks involve intrinsic service paths and multiple candidate entry-exit configurations. A *point task* is considered served when the agent passes through its center. A *line task* must be physically traversed along its segment, offering two candidate entry-exit pairs corresponding to traversal in either direction. For an *area task*, the agent must execute a coverage path under detection range γ, without assuming a specific coverage pattern. For each area task, we retain the ω shortest feasible coverage paths as candidate service options, each associated with a corresponding entry–exit pair.

Formally, let n_a, n_l, and n_p denote the number of area, line, and point tasks, respectively, where each of them can be zero to allow point-only, line-only, area-only, and hybrid instances. We define the full candidate set as $\mathcal{V} = \{0\} \cup \mathcal{V}_a \cup \mathcal{V}_l \cup \mathcal{V}_p$, where 0 is the depot. The area candidate set is $\mathcal{V}_a = \{1, \ldots, \omega \cdot n_a\}$, with ω candidates per area task. The line candidate set is $\mathcal{V}_l = \{\omega \cdot n_a + 1, \ldots, \omega \cdot n_a + 2n_l\}$, and the point task set is $\mathcal{V}_p = \{\omega \cdot n_a + 2n_l + 1, \ldots, \omega \cdot n_a + 2n_l + n_p\}$. In this paper, we treat candidate nodes within the same task (e.g., the ω entry–exit pairs associated with a single area task) as *semantically equivalent*. Each candidate is associated with an entry–exit coordinate pair. For each line or area candidate $i \in \mathcal{V}_a \cup \mathcal{V}_l$, we represent this pair as $L_i = \{L_i^+, L_i^-\}$, where L_i^+ and L_i^- denote the entry and exit locations, respectively. Additionally, each such candidate $i \in \mathcal{V}_a \cup \mathcal{V}_l$ is assigned a service cost c_i, corresponding to the segment length for line tasks or the coverage path length induced by the geometry of the associated area task. For other candidates $i \in \{0\} \cup \mathcal{V}_p$, the entry and exit locations coincide and the service cost $c_i = 0$.

The agent must construct a tour that starts and ends at the depot, selects exactly one entry-exit pair for each non-point task, and visits all tasks exactly once. Let $\pi = (\pi_0, \pi_1, \ldots, \pi_N)$, where $N = n_a + n_l + n_p$, $\pi_i \in \mathcal{V}$, and $\pi_0 = 0$, denote a sequence of selected candidates (with exactly one candidate chosen per task). Let $d_{\pi_i \pi_j}$ denote the travel cost from candidate π_i to candidate π_j. Importantly, $d_{\pi_i \pi_j} \neq d_{\pi_j \pi_i}$ in general due to the asymmetric nature of line and area tasks. For example, if $\pi_i \in \mathcal{V}_p$ and $\pi_j \in \mathcal{V}_l$, then $d_{\pi_i \pi_j} = \|L_i - L_j^+\|_2$ and $d_{\pi_j \pi_i} = \|L_j^- - L_i\|_2$.

The total cost consists of the sum of all inter-task travel distances, including the return to the depot, as well as the intrinsic service costs associated with each line and area task. The objective is given by:

$$\min_{\pi} f(\pi) = \sum_{i=0}^{N-1} d_{\pi_i \pi_{i+1}} + d_{\pi_N \pi_0} + \sum_{j=0}^{N} c_{\pi_j}. \tag{1}$$

This formulation captures the key characteristics of CGRP, including task-specific geometries, asymmetric routing, intrinsic service paths, and combinatorial candidate selection, while retaining classical routing problems as special cases.

MDP Formulation. To solve CGRP with RL-based neural policies, we formulate it as an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ denotes the transition dynamics, and $\mathcal{R}$ is the reward function.

- **State.** A state $s_t \in \mathcal{S}$ encodes the current partial solution together with the input instance $g \in \mathcal{G}$. The initial state s_0 contains only the depot, indicating the agent's starting position. The terminal state s_T corresponds to a complete solution $\pi = (\pi_0, \pi_1, \ldots, \pi_T)$, where $T = N$, and all tasks have been visited exactly once.
- **Action.** The action space $\mathcal{A} = \mathcal{V}$ consists of the depot 0, all point tasks $\mathcal{V}_p$, and all candidate entry-exit pairs of line and area tasks in $\mathcal{V}_l$ and $\mathcal{V}_a$. An action $a_t \in \mathcal{A}$ selects the next task to visit, along with its entry-exit pair if applicable.
- **Transition.** The next state s_{t+1} is derived deterministically from s_t by the chosen action a_t. Once a candidate of a non-point task is selected as a_t, all other candidates corresponding to that task are marked as visited. The depot remains masked until all tasks are visited.

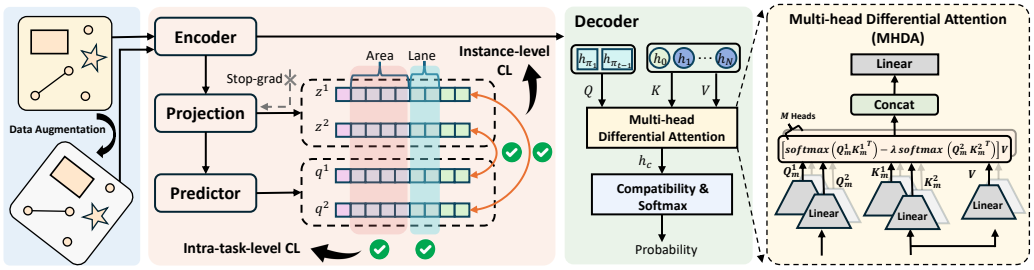


Figure 2: An overview of DiCon. DiCon incorporates two mechanisms: 1) a multi-head differential attention module that replaces the standard MHA in the decoder to facilitate decision-making, and 2) a double-level contrastive objective to enhance representation learning.

- **Reward.** The agent receives zero reward at each intermediate step $t < T$ and a terminal reward at s_T equal to the negative total cost of the constructed tour $\mathcal{R}(s_T, \cdot) = -f(\pi)$.
- **Policy.** The policy p_θ, parameterized by θ, constructs a complete solution autoregressively, and its probability can be factorized via the chain rule as:

$$p_\theta(\pi|g) = \prod_{t=0}^{T} p_\theta(\pi_t|\pi_{<t}, g). \tag{2}$$

With a slight abuse of notation, we use π_t to denote the action selected at step t, and $\pi_{<t}$ to represent the partial solution up to step $t-1$.

4 Methodology

4.1 Overview

Most neural routing solvers adopt a Transformer-style architecture consisting of a multi-layer encoder and a single-layer decoder. To extend such architectures to the CGRP, we propose a unified encoding scheme capable of handling different task geometries. Its effectiveness is validated in Appendix L. Given an instance $g \in \mathcal{G}$, we extract three sets of raw features: 1) the coordinate pair $L_i \in \mathbb{R}^4$ for each $i \in \mathcal{V}$; 2) a geometric anchor $G_i \in \mathbb{R}^2$ for each non-depot candidate $i \in \mathcal{V} \setminus \{0\}$, representing either the task location (for point tasks) or the geometric center (for line and area tasks); and 3) a task-type indicator $I_i \in \{0, 1, 2\}$ representing area, line, and point tasks, respectively, also defined for each $i \in \mathcal{V} \setminus \{0\}$. The initial embedding for each non-depot node is then given by:

$$h_i^0 = W_L L_i + W_G G_i + W_I I_i + b, \tag{3}$$

where W_L, W_G, W_I are learnable projection matrices and b is a shared bias term. The initial embedding of the depot is computed separately as $h_0^0 = W_d L_0 + b_d$, where W_d and b_d are learnable parameters. Finally, we obtain the full input sequence as: $H^0 = [h_0^0, h_1^0, \ldots, h_{|\mathcal{V}|}^0] \in \mathbb{R}^{(|\mathcal{V}|+1)\times d}$, where d is the hidden dimension and $[\cdot]$ denotes the row-wise concatenation. This unified encoding allows the encoder to reason over compositional geometry tasks within a shared representation space while preserving the semantic and geometric diversity essential for CGRP. The $\mathbb{L}$-layer encoder transforms the initial embeddings H^0 into task-aware representations $H^{\mathbb{L}}$. Conditioned on the output of the encoder and the evolving state s_t, the decoder is repeatedly invoked to select actions in an autoregressive manner.

As illustrated in Fig. 2, DiCon introduces two dedicated components to address the intrinsic challenges of CGRP, namely, the abundance of feasible yet less competitive candidate actions and the compositional geometric structure of tasks. First, we incorporate a DA mechanism into the decoder to suppress probability mass assigned to strategically weak candidates, thereby promoting more focused and effective decision-making. Second, to improve representational capacity for compositional geometry reasoning, we introduce a double-level CL objective: an instance-level CL loss that promotes robust global representations across different task compositions, and an intra-task CL loss that reinforces semantic coherence within the same task while preserving inter-task diversity. Notably, DiCon is a generic, plug-and-play framework that can be seamlessly integrated into existing neural routing solvers without architectural redesign.

4.2 Differential Attention Mechanism

Most neural routing solvers employ a single-layer decoder composed of a multi-head attention (MHA) sublayer followed by a compatibility sublayer. Given the encoder output $H^{\mathbb{L}} = \{h_0^{\mathbb{L}}, \ldots, h_{|\mathcal{V}|}^{\mathbb{L}}\}$ and contextual features at step t, such as the embedding of the first selected node h_{π_1} and the most recently visited node $h_{\pi_{t-1}}$, the decoder first constructs query, key, and value representations to compute a context embedding h_c via MHA. This context embedding is then passed through a compatibility sublayer to produce a probability distribution over all available actions. The details of the model forward pass are provided in Appendix A. To enhance robustness in the presence of numerous feasible yet less competitive actions, a common challenge in CGRP, we integrate a DA mechanism into the MHA sublayer of the decoder. This mechanism performs attention denoising by suppressing the influence of low-saliency candidate actions, enabling the decoder to focus more effectively on critical decision points [54]. Through this targeted integration, DiCon improves the decoder's discrimination and decision-making capability in compositional geometry routing settings.

Specifically, we partition the query and key vectors into two groups and compute two separate softmax attention maps for each head in the MHA sublayer. Formally, for head $m \in [1, \cdots, M]$ with total heads number M:

$$Q_m^1 = [h_{\pi_1}, h_{\pi_{t-1}}]W_{q_m}^1,\ Q_m^2 = [h_{\pi_1}, h_{\pi_{t-1}}]W_{q_m}^2, \tag{4}$$

$$K_m^1 = H^{\mathbb{L}}W_{k_m}^1,\ K_m^2 = H^{\mathbb{L}}W_{k_m}^2, \tag{5}$$

where $W_{q_m}^1, W_{q_m}^2, W_{k_m}^1, W_{k_m}^2$ are learnable parameters. Then we obtain the softmax logits of the two groups of query and key as follows:

$$A_m^1 = \text{softmax}(\frac{Q_m^1 {K_m^1}^T}{\sqrt{d/m}}),\ A_m^2 = \text{softmax}(\frac{Q_m^2 {K_m^2}^T}{\sqrt{d/m}}). \tag{6}$$

Then the result of subtracting these two maps is regarded as attention scores: $s_m = (A_m^1 - \lambda_m A_m^2)V$, where λ_m is a head-dependent learnable scalar initialized as λ_0 (see Appendix E) and $V = H^{\mathbb{L}}W_{v_m}$ with the learnable parameter W_{v_m}. The DA mechanism mitigates attention noise, encouraging the model to focus on critical information (see Appendix N). The outputs of all attention heads are projected to the final context embedding: $h_c = [s_1, \cdots, s_M]W_O$, where $W_O \in \mathbb{R}^{d\times d}$ is a learnable projection matrix. The context embedding h_c is used to compute the compatibility scores $\alpha_i = C \cdot \tanh\left(\frac{h_c^\top W^K h_i^{\mathbb{L}}}{\sqrt{d}}\right)$, where $\alpha_i = -\infty$ if the candidate is masked, C is a clipping constant, and W^K is a learnable projection. The final selection probability is given by $p_\theta(\pi_t \mid \pi_{<t}, g) = \mathrm{Softmax}(\alpha)$. During training, parallel solutions are sampled from $p_\theta(\cdot \mid g)$, while inference selects the highest-probability candidate greedily.

4.3 Double-Level Contrastive Learning

Most neural solvers rely on RL to learn a policy p_θ that produces near-optimal solutions. However, the compositional geometry nature of CGRP poses representation challenges. To address this, we augment the standard RL objective with two CL objectives: an *instance level* CL objective and an *intra-task* CL objective. Together, these objectives enhance the representational capacity of the backbone model and enable effective reasoning over compositional geometric tasks.

Instance Level CL Loss. To improve the robustness of global representations for CGRP instances, we adopt a data augmentation strategy that exploits the rotation invariance of routing problems, under which the optimal solution remains unchanged by rigid transformations of the coordinate system. Specifically, given an instance $g \in \mathcal{G}$, we generate $\mathbb{A}$ augmented views by applying random spatial rotations and axis permutations to all geometric entities, including candidate locations L_i and geometric anchors G_i.

In specific, we first sample a transformation index $\alpha \sim \mathcal{U}(0, 1)$, then apply a continuous rotation by

$$\phi = \begin{cases} 4\pi\alpha, & \alpha < 0.5, \\ 4\pi(\alpha - 0.5), & \alpha \geq 0.5, \end{cases} \tag{7}$$

around the center of a unit square. Given a coordinate (x, y), the rotated coordinates are computed as:

$$\begin{aligned} x' &= \cos(\phi)(x - 1/2) - \sin(\phi)(y - 1/2) + 1/2, \\ y' &= \sin(\phi)(x - 1/2) + \cos(\phi)(y - 1/2) + 1/2. \end{aligned} \tag{8}$$

Additionally, for half of the transformations ($\alpha \geq 0.5$), the coordinate axes are swapped, i.e., $(x', y') \leftarrow (y', x')$, introducing reflectional symmetry augmentation. This transformation is applied consistently to all geometric components (i.e., candidate locations L_i and geometric anchors G_i). Repeating this process $\mathbb{A} - 1$ times yields a set of augmented instances $\{g^1, \ldots, g^{\mathbb{A}}\}$, where g^1 is the original instance. As shown in Fig. 2, the encoder processes each augmented instance to produce all candidate embeddings $H^{\mathbb{L}}(g^\xi), \xi \in \{1, \cdots, \mathbb{A}\}$, which are mapped through a projection head $\varphi(\cdot)$ and a predictor head $\psi(\cdot)$ to obtain latent representations $z^\xi = \varphi(H^{\mathbb{L}}(g^\xi))$ and predictions $q^\xi = \psi(z^\xi)$. Following a SimSiam-style objective [8], we encourage agreement between different augmented views by maximizing the cosine similarity $\mathcal{D}$ between the predictor output and the stop-gradient projection output:

$$\mathcal{D}(q^\xi, z^\tau) = \frac{q^\xi}{\|q^\xi\|_2} \cdot \frac{z^\tau}{\|z^\tau\|_2}, \ \xi \neq \tau. \tag{9}$$

The instance-level contrastive loss is defined as:

$$\mathcal{L}_{\text{CL}}^{\text{ins}} = -\frac{1}{\mathbb{A}(\mathbb{A}-1)} \sum_{\xi=1}^{\mathbb{A}} \sum_{\tau \neq \xi} \mathcal{D}(q^\xi, z^\tau). \tag{10}$$

Intra-Task CL Loss. To enable the model to distinguish between different tasks in the latent space, task representations should be well separated. However, for non-point tasks, multiple geographically distinct entry-exit pairs correspond to the same underlying task. Treating these candidates independently, without accounting for their shared semantic identity, may hinder representation learning. To address this, we introduce an intra-task contrastive objective that encourages the embeddings of entry-exit candidates belonging to the same task to be close in the latent space. Unlike instance-level contrast, which promotes global consistency, this objective refines the task-specific feature subspace by clustering candidate embeddings of the same task while preserving separability across tasks.

Specifically, given a set of augmented views $\{g^1, \cdots, g^{\mathbb{A}}\}$ and their latent embeddings z^ξ and predictions q^ξ, for $\xi \in \{1, \cdots, \mathbb{A}\}$, we compute contrastive losses separately for area and line tasks. For each area task, we randomly select the prediction q_i^ξ of one candidate i from view ξ as the query and treat the remaining $\omega - 1$ semantically equivalent candidates from another view τ as positive samples, denoted by $\{z_{i_1^+}^\tau, z_{i_2^+}^\tau, z_{i_{(\omega-1)}^+}^\tau\}$. The contrastive loss for area tasks is then defined as:

$$\mathcal{L}_{\text{area}}(\xi, \tau) = \frac{1}{(\omega-1)n_a} \sum_{i=1}^{n_a} \sum_{j=1}^{\omega-1} \mathcal{D}(q_i^\xi, z_{i_j^+}^\tau). \tag{11}$$

Similarly, for each line task, we randomly select one candidate's prediction q_i^ξ and treat its directionally opposite candidate from view τ, denoted $z_{i^+}^\tau$, as the positive sample. The contrastive loss for line tasks is computed as:

$$\mathcal{L}_{\text{line}}(\xi, \tau) = \frac{1}{n_l} \sum_{i=1}^{n_l} \mathcal{D}(q_i^\xi, z_{i^+}^\tau). \tag{12}$$

Combining all view pairs, the total intra-task contrastive loss is expressed as:

$$\mathcal{L}_{\text{CL}}^{\text{intra}} = -\frac{1}{2\mathbb{A}(\mathbb{A}-1)} \sum_{\xi=1}^{\mathbb{A}} \sum_{\tau \neq \xi} \left[\mathcal{L}_{\text{area}}(\xi, \tau) + \mathcal{L}_{\text{line}}(\xi, \tau)\right]. \tag{13}$$

This loss encourages semantically equivalent candidates of the same task to lie closer in the latent space, enhancing intra-task feature compactness. By incorporating this structured contrastive signal, the encoder becomes more sensitive to the geometric coherence of non-point tasks, an essential property in CGRP, where multiple candidate representations may correspond to the same task.

RL Loss. To optimize the CGRP policy, we use the REINFORCE algorithm [51], a classical Monte Carlo policy gradient method, to train the backbone model. The expected loss is formulated as:

$$\mathcal{L}_{\text{RL}}^\theta = -\mathbb{E}_{g \sim \mathcal{G}} \mathbb{E}_{\pi \sim p_\theta(\pi|g)}[\mathcal{R}(\pi)]. \tag{14}$$

To reduce variance, we apply a shared baseline $\tilde{\mathcal{R}}(g)$ and approximate the gradient via Monte Carlo sampling. Given a batch of $\mathbb{A}$ augmented views $\{g^\xi\}_{\xi=1}^{\mathbb{A}}$ and N sampled solutions per instance g, the

empirical gradient is calculated as:

$$\nabla_\theta \mathcal{L}_{\text{RL}}^\theta \approx -\frac{1}{N\mathbb{A}} \sum_{\xi=1}^{\mathbb{A}} \sum_{j=1}^{N} \left(\mathcal{R}(\pi^{\xi,j}) - \tilde{\mathcal{R}}(g) \right) \nabla_\theta \log p_\theta \left(\pi^{\xi,j} | g^\xi \right), \tag{15}$$

with the baseline $\tilde{\mathcal{R}}(g) = \frac{1}{N\mathbb{A}} \sum_{\xi=1}^{\mathbb{A}} \sum_{j=1}^{N} \mathcal{R}(\pi^{\xi,j})$.

Global Objective. To enhance the backbone's robustness and geometry-awareness in solving CGRP, we design a triplet loss incorporating auxiliary contrastive objectives at both the instance and intra-task levels. The instance level CL loss $\mathcal{L}_{\text{CL}}^{\text{ins}}$ promotes consistent global representations across augmented views, while the intra-task CL loss $\mathcal{L}_{\text{CL}}^{\text{intra}}$ encourages semantically coherent representations among candidates belonging to the same task. The overall training objective is a weighted combination of the reinforcement and contrastive components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RL}}^\theta + \lambda_{\text{ins}} \mathcal{L}_{\text{CL}}^{\text{ins}} + \lambda_{\text{intra}} \mathcal{L}_{\text{CL}}^{\text{intra}}, \tag{16}$$

where λ_{ins} and λ_{intra} are tunable hyperparameters that balance the auxiliary losses. The encoder, projection head, and predictor parameters are optimized jointly by the triplet loss, while the decoder is only optimized by the RL loss.

5 Experiment

5.1 Experimental Setting

Training Data. During training, we use a mixed-composition CGRP distribution. Specifically, the problem size is sampled uniformly from $[20, 100)$, the number of area tasks is sampled from $[0, 5)$, and the number of line tasks is sampled from $[0, 20)$, with the remaining tasks treated as point tasks. All task coordinates are uniformly sampled from the unit square $[0, 1]^2$, subject to geometric constraints that prevent overlap. For training-time parameterization, area tasks use zigzag coverage paths aligned with the longer side to reduce traversal cost, with $\omega = 4$ candidate service options; details are provided in Appendix C. We further evaluate irregular area tasks with alternative coverage methods and different values of ω in Appendix F. This training distribution mainly contains point-dominant hybrid instances, while single-geometry and more extreme compositions are reserved as held-out settings for evaluating compositional generalization.

Evaluation Data. For evaluation, we first consider in-distribution instances with problem sizes 20 and 100, where the numbers of line and area tasks follow the same sampling ranges as in training. We then assess scalability by testing on larger instances with 200 and 300 tasks, using the same sampling strategy. To further examine compositional generalization across diverse geometry compositions, we construct specialized test sets with different task configurations. `Point20` and `Point100` contain only point tasks, with 20 and 100 tasks, respectively. `Line20` and `Line100` contain only line tasks, while `Area20` and `Area100` contain only area tasks. In addition, `Area20+Line30` and `Area20+Line80` mix area and line tasks with moderate and high line densities, respectively. For each problem setting, we generate 200 test instances to form the test dataset.

Hyperparameter. We configure the neural network with an embedding dimension of 128, 6 encoder layers, and 8 attention heads with a query-key-value dimension of 16 per head. The feed-forward network hidden dimension is set to 512, and logit clipping C is set to 10. For the double-level CL, we generate $\mathcal{A} = 4$ augmented views per instance through rotation and reflection transformations. The weights of the instance-level and intra-task contrastive losses are set to $\lambda_{ins} = 0.1$ and $\lambda_{intra} = 0.02$, respectively. The hyperparameter study for double-level CL is provided in Appendix P.2. The model parameters are optimized using Adam with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-6}. The learning rate is decayed by a factor of 0.1 at epoch 180. The model is trained for 200 epochs with each epoch processing 100,000 instances and a batch size of 64. During inference, we use greedy rollout for evaluation. Methods with the "-Aug" suffix apply instance augmentation [34] to further improve solution quality.

Baseline. We compare our approach against several state-of-the-art algorithms, including both traditional and RL-based neural solvers. For traditional solvers, we utilize LKH3 [24], a heuristic typically producing near-optimal solutions for TSP variants, Google OR-Tools [16], an efficient optimization toolkit with specialized routing capabilities, and Adaptive Large Neighborhood Search (ALNS) [45],

Table 1: Performance comparison. Left: in-distribution testing. Right: generalization to larger sizes.

	20			100			200			300		
	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
Gurobi	**6.065**[*]	**-0.02**	8.49m	8.423	0.82	2.86h	14.451	5.66	5.03h	16.253	7.17	10.08h
LKH3	<u>6.066</u>	<u>0.00</u>	7.53m	**8.354**	**0.00**	50.30m	**13.677**	**0.00**	3.32h	**15.165**	**0.00**	7.96h
OR-Tools	6.079	0.20	6.67m	8.593	2.85	33.35m	14.073	2.89	1.67h	15.721	3.67	2.78h
ALNS	6.127	0.99	1.03m	8.525	2.05	33.57m	14.466	5.77	1.69h	16.642	9.74	2.83h
POMO	6.076	0.15	0.09s	8.465	1.32	0.19s	14.115	3.20	0.94s	16.344	7.78	2.58s
DiCon-P	6.075	0.14	0.07s	8.406	0.63	0.16s	13.949	1.99	0.80s	16.021	5.65	2.14s
RELD	6.074	0.13	0.09s	8.406	0.63	0.26s	13.922	1.79	1.17s	15.839	4.45	2.99s
DiCon-R	6.072	0.10	0.08s	8.388	0.41	0.31s	13.845	1.23	1.83s	15.702	3.54	4.55s
POMO-Aug	<u>6.066</u>	<u>0.00</u>	1.43s	8.402	0.57	2.40s	13.978	2.20	15.07s	16.129	6.36	31.88s
DiCon-P-Aug	<u>6.066</u>	<u>0.00</u>	1.42s	8.376	0.26	2.69s	13.854	1.29	17.05s	15.853	4.54	38.96s
RELD-Aug	**6.065**	**-0.02**	1.36s	8.373	0.23	3.42s	13.824	1.07	17.65s	15.724	3.69	35.67s
DiCon-R-Aug	**6.065**	**-0.02**	1.36s	<u>8.363</u>	<u>0.10</u>	4.01s	<u>13.755</u>	<u>0.57</u>	20.95s	<u>15.595</u>	<u>2.84</u>	43.44s

[*] The result is obtained by Gurobi without a time limit and thus represents the exact optimum.

a destroy-and-repair metaheuristic. For neural solvers, we compare wth POMO [34], an RL-based method exploiting problem symmetry for routing problems, and RELD [28], which enhances decoder capacity through identity mapping and feed-forward layers to improve generalization. We also compare with Gurobi, a commercial exact solver that provides optimal solutions for size 20 instances. For larger scales, Gurobi is given time limits of $60s$, $90s$, and $180s$ for sizes 100, 200, and 300, respectively. In the compositional generalization experiments, Gurobi provides optimal solutions for `Point20` and `Line20` without time limits. For the remaining settings, we impose time limits as follows: 10s for `Area20`, 20s for `Area20+Line30`, and 60s for `Point100`, `Line100`, `Area100`, and `Area20+Line80`. We apply DiCon to two neural routing backbones, POMO and RELD, resulting in DiCon-P and DiCon-R, respectively. All methods are executed on a machine equipped with an NVIDIA A100-SXM4-80GB GPU and an AMD EPYC 7742 64-Core Processor. The code and dataset are publicly released for reproducibility.[2]

Metric. We evaluate algorithm performance using three metrics: the average objective value (Obj.) representing the total tour length; the average gap (Gap) calculated as (Obj - Baseline)/Baseline × 100%; and total run time (Time) per test dataset measured in CPU seconds. We select LKH3 as the primary Baseline for computing gaps due to its strong and consistent performance among solvers. We highlight the best Obj and Gap in **bold** and <u>underline</u> the second-best Obj and Gap.

5.2 Comparison Analysis

The results are reported in Table 1, covering both in-distribution instances and larger-size generalization settings. DiCon consistently improves different neural routing backbones. On in-distribution instances, DiCon-P and DiCon-R achieve competitive solution quality with substantially lower runtime than Gurobi and LKH3. On larger 200-task and 300-task instances, DiCon-P and DiCon-R clearly outperform their original backbones, POMO and RELD, with the performance gap becoming more pronounced as the problem size increases. With data augmentation, DiCon achieves further improvements in solution quality. In terms of efficiency, DiCon variants require much less runtime than exact and traditional heuristic solvers. These results demonstrate that DiCon provides a strong quality-efficiency trade-off and robust generalization to problem sizes beyond the training range.

5.3 Performance Across Diverse Geometry Compositions

We consider two types of geometry composition settings: single-geometry and extreme hybrid-geometry compositions. The experimental results are reported in Table 2.

Performance on Single-Geometry Settings. DiCon achieves consistent improvements across single-geometry settings, with the advantage becoming more pronounced as the problem size increases. On `Point20` and `Line20`, DiCon-R-Aug reaches optimal solution quality. Notably, line-only and area-only instances are not included in the training distribution, suggesting that DiCon can generalize to held-out geometry compositions. DiCon-R-Aug remains competitive with LKH3 while requiring much less runtime, and even achieves better solution quality on `Line20`, `Line100`, and `Area20`.

[2] https://github.com/mingfan321/Learning-to-Solve-Compositional-Geometry-Routing-Problems

Table 2: Performance comparison across different compositional geometry instances.

	Point20 (TSP)			Point100 (TSP)			Line20			Line100		
	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
Gurobi	**3.918**[*]	**0.00**	5.27m	8.011	2.53	3.23h	**4.339**[*]	**-0.02**	7.10h	**12.184**	**-3.17**	2.82h
LKH3	**3.918**	**0.00**	4.93m	**7.813**	**0.00**	20.98m	4.340	0.00	8.57m	12.582	0.00	29.77m
OR-Tools	3.918	0.00	6.68m	8.006	2.47	33.34m	4.356	0.36	6.68m	13.398	6.48	33.38m
ALNS	3.930	0.30	33.03s	7.911	1.25	44.30m	4.382	0.97	1.08m	12.697	0.91	43.61m
POMO	3.923	0.11	0.03s	7.903	1.16	0.21s	4.347	0.16	0.03s	12.623	0.32	0.71s
DiCon-P	3.921	0.06	0.02s	7.857	0.56	0.25s	4.343	0.07	0.04s	12.487	-0.76	0.76s
RELD	3.920	0.05	0.03s	7.855	0.55	0.25s	4.345	0.12	0.03s	12.471	-0.88	0.68s
DiCon-R	3.920	0.04	0.03s	7.844	0.40	0.33s	4.341	0.02	0.05s	12.394	-1.50	0.92s
POMO-Aug	3.919	0.00	0.20s	7.850	0.47	1.61s	4.340	0.00	0.26s	12.494	-0.71	4.36s
DiCon-P-Aug	3.919	0.01	0.19s	7.831	0.24	1.95s	**4.339**	**-0.02**	0.27s	12.389	-1.54	6.05s
RELD-Aug	**3.918**	**0.00**	0.22s	7.829	0.21	2.03s	4.340	0.00	0.26s	12.382	-1.59	5.17s
DiCon-R-Aug	**3.918**	**0.00**	0.22s	7.822	0.12	2.61s	**4.339**	**-0.02**	0.31s	12.319	-2.10	7.71s

	Area20			Area100			Area20+Line30			Area20+Line80		
	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
Gurobi	**9.387**	**-0.26**	22.2m	24.250	0.20	3.33h	12.394	-1.04	1.02h	**16.078**	**-2.62**	1.86h
LKH3	9.411	0.00	8.17m	**24.200**	**0.00**	27.67m	12.524	0.00	16.00m	16.510	0.00	43.63m
OR-Tools	9.443	0.35	6.70m	24.841	2.65	33.55m	12.637	0.91	16.71m	16.791	1.70	33.41m
ALNS	9.523	1.20	1.07m	24.359	0.66	43.16m	12.665	1.12	11.85m	16.628	0.71	33.82m
POMO	9.469	0.62	0.04s	24.725	2.17	2.57s	12.504	-0.16	0.16s	16.516	0.03	0.53s
DiCon-P	9.459	0.52	0.06s	24.483	1.17	3.39s	12.470	-0.43	0.23s	16.390	-0.73	0.80s
RELD	9.466	0.59	0.05s	24.472	1.12	3.11s	12.461	-0.50	0.19s	16.370	-0.85	1.01s
DiCon-R	9.436	0.27	0.07s	24.303	0.42	3.41s	12.437	-0.70	0.27s	16.303	-1.25	1.37s
POMO-Aug	9.417	0.06	0.44s	24.543	1.42	14.78s	12.426	-0.78	2.35s	16.372	-0.84	7.45s
DiCon-P-Aug	9.411	0.00	0.45s	24.369	0.70	20.19s	12.408	-0.93	3.57s	16.287	-1.35	8.39s
RELD-Aug	9.419	0.09	0.51s	24.354	0.63	17.04s	12.402	-0.98	2.41s	16.277	-1.41	8.97s
DiCon-R-Aug	9.400	-0.11	0.57s	24.217	0.07	23.71s	**12.385**	**-1.11**	3.66s	16.217	-1.78	10.87s

[*] The result is obtained by Gurobi without a time limit and thus represents the exact optimum.

Performance on Extreme hybrid-geometry. For unseen extreme hybrid compositions, DiCon-R-Aug achieves best performance on `Area20+Line30` and outperforms LKH3 on `Area20+Line30` and `Area20+Line80`, suggesting that DiCon is effective in handling extreme hybrid-geometry.

5.4 Ablation Study

To assess the contribution of key components in DiCon, we progressively remove the intra-task (IT) CL, instance-level (IL) CL, and DA modules from DiCon-P, yielding DiCon-P w/o IT, DiCon-P w/o IT+IL, and the backbone POMO, respectively. The results on CGRP100 and `Area20+Line30` are visualized in Fig. 3. Performance consistently degrades as components are removed, demonstrating the complementary and positive contributions of each module.

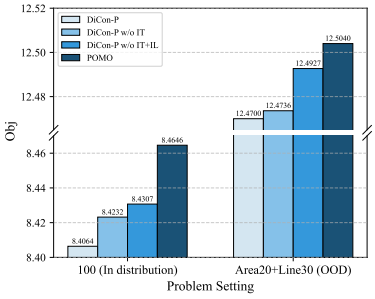


Figure 3: Ablation study.

6 Conclusion

In this paper, we study the Compositional Geometry Routing Problem (CGRP), which generalizes conventional routing by covering point, line, area, and hybrid task geometries that commonly arise in real-world applications. We formulate CGRP solving as an MDP and identify its key characteristics and challenges. Our primary contribution is DiCon, a plug-and-play framework that combines a differential attention mechanism with a double-level contrastive objective to enhance both representation learning and decision-making. Empirically, DiCon consistently demonstrates strong performance, broad versatility, and robust compositional generalization across diverse problem settings and backbone models. The present formulation focuses on single-agent routing without complex operational constraints, such as time windows and capacity limits. Future work will extend DiCon with constraint-aware modules and investigate multi-agent CGRP, where hierarchical or decentralized RL may be required for effective coordination.

Acknowledgments and Disclosure of Funding

This research is partly supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-031).

References

[1] Saurav Agarwal and Srinivas Akella. Line coverage with multiple robots: algorithms and experiments. *IEEE Transactions on Robotics*, 40:1664–1683, 2024.

[2] Ahmad Bilal Asghar, Shreyas Sundaram, and Stephen L Smith. Multi-robot persistent monitoring: Minimizing latency and number of robots with recharging constraints. *IEEE Transactions on Robotics*, 2024.

[3] Rik Bähnemann, Nicholas Lawrance, Jen Jen Chung, Michael Pantic, Roland Siegwart, and Juan Nieto. Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem. In *Field and Service Robotics: Results of the 12th International Conference*, pages 277–290. Springer, 2021.

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

[5] Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, et al. Rl4co: an extensive reinforcement learning for combinatorial optimization benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5278–5289, 2025.

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

[7] 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.

[8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[9] Jiaqi Cheng, Mingfeng Fan, Xuefeng Zhang, Jingsong Liang, Yuhong Cao, Guohua Wu, and Guillaume Adrien Sartoretti. Multimodal fused learning for solving the generalized traveling salesman problem in robotic task planning. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 1562–1575. PMLR, 27–30 Sep 2025.

[10] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *Advances in Neural Information Processing systems*, 33:8765–8775, 2020.

[11] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 2002.

[12] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] 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.

[14] Haonan Duan, Pashootan Vaezipoor, Max B Paulus, Yangjun Ruan, and Chris Maddison. Augment with care: Contrastive learning for combinatorial problems. In *International Conference on Machine Learning*, pages 5627–5642. PMLR, 2022.

[15] Merrill M Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[16] Vincent Furnon and Laurent Perron. OR-Tools Routing Library version v9.15. `https://developers.google.com/optimization/routing/`, 2026.

[17] 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-First International Joint Conference on Artificial Intelligence*, 2024.

[18] Sheng Gao, Jiazheng Wu, and Jianliang Ai. Multi-UAV reconnaissance task allocation for heterogeneous targets using grouping ant colony optimization algorithm. *Soft Computing*, 25(10):7155–7167, 2021.

[19] John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. DeCLUTR: Deep contrastive learning for unsupervised textual representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 879–895, 2021.

[20] Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Thomas D Barrett. Winner takes it all: Training performant RL populations for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2023.

[21] Ziao Guo, Yang Li, Shiyue Wang, and Junchi Yan. ConRep4CO: Contrastive representation learning of combinatorial optimization instances across types. In *The Fourteenth International Conference on Learning Representations*, 2026.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[23] Keld Helsgaun. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[24] Keld Helsgaun. LKH-3 version 3.0.7. `http://webhotel4.ruc.dk/~keld/research/LKH-3/`, 2017.

[25] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations, 2022, Virtual Event, April 25-29, 2022*, 2022.

[26] Qingchun Hou, Jingwei Yang, Yiqiang Su, Xiaoqing Wang, and Yuming Deng. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *International Conference on Learning Representations*, 2023.

[27] Taoan Huang, Aaron M Ferber, Arman Zharmagambetov, Yuandong Tian, and Bistra Dilkina. Contrastive predict-and-search for mixed integer linear programs. In *International Conference on Machine Learning*, 2024.

[28] 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.

[29] Ya-Hui Jia, Qiquan Zheng, Yang Wang, Yi Mei, Wei-Neng Chen, and Zhenhong Lin. A neural solver with traversal-based feature representation and adjacent attention for capacitated arc routing problem. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[30] Yuan Jiang, Zhiguang Cao, Yaoxin Wu, and Jie Zhang. Multi-view graph contrastive learning for solving vehicle routing problems. In *Uncertainty in Artificial Intelligence*, pages 984–994. PMLR, 2023.

[31] Daniel Karapetyan and Gregory Gutin. Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem. *European Journal of Operational Research*, 208(3):221–232, 2011.

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

[33] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems. In *International Conference on Learning Representations*, 2018.

[34] 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.

[35] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

[36] Jiayi Li, Guohua Wu, Mingfeng Fan, Zhiguang Cao, and Yalin Wang. Heterogeneous attention-based graph convolutional network for solving asymmetric pickup and delivery problem. *IEEE Transactions on Automation Science and Engineering*, 2025.

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

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

[39] 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.

[40] 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.

[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26, 2013.

[42] František Nekovář, Jan Faigl, and Martin Saska. Multi-tour set traveling salesman problem in planning power transmission line inspection. *IEEE Robotics and Automation Letters*, 6(4):6196–6203, 2021.

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

[44] Mogens Plessen. Path planning for spot spraying with UAVs combining TSP and area coverages. *Smart Agricultural Technology*, page 100965, 2025.

[45] Stephen L Smith and Frank Imeson. GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, 87:1–19, 2017.

[46] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European Conference on Computer Vision*, pages 776–794. Springer, 2020.

[47] Marjolein Veenstra, Kees Jan Roodbergen, Iris FA Vis, and Leandro C Coelho. The pickup and delivery traveling salesman problem with handling costs. *European Journal of Operational Research*, 257(1):118–132, 2017.

[48] Vishnu Veeraraghavan, Kyle Hunte, Jingang Yi, and Kaiyan Yu. Complete and near-optimal robotic crack coverage and filling in civil infrastructure. *IEEE Transactions on Robotics*, 40:2850–2867, 2024.

[49] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700, 2015.

[50] Xinyu Wang, Tsan-Ming Choi, Haikuo Liu, and Xiaohang Yue. A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(4):545–556, 2016.

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

[52] Xuan Wu, Di Wang, Lijie Wen, Yubin Xiao, Chunguo Wu, Yuesong Wu, Chaoyu Yu, Douglas L Maskell, and You Zhou. Neural combinatorial optimization algorithms for solving vehicle routing problems: A comprehensive survey with perspectives. *arXiv preprint arXiv:2406.00415*, 2024.

[53] Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[54] Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer. In *International Conference on Learning Representations*, 2025.

[55] Hao Yuan, Wenli Ouyang, Changwen Zhang, Congrui Li, and Yong Sun. OPTFM: A scalable multi-view graph transformer for hierarchical pre-training in combinatorial optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[56] Zhongju Yuan, Genghui Li, Zhenkun Wang, Jianyong Sun, and Ran Cheng. RL-CSL: A combinatorial optimization method using reinforcement learning and contrastive self-supervised learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4):1010–1024, 2022.

[57] Changming Zhang, Caiyue Xu, Xu Cheng, Xin Li, Gang Li, and Bin He. Multi-region joint coverage for environmental monitoring using energy-constrained UAVs. *IEEE Transactions on Instrumentation and Measurement*, 2025.

[58] Zhi Zheng, Changliang Zhou, Tong Xialiang, Mingxuan Yuan, and Zhenkun Wang. UDC: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 37:6081–6125, 2024.

[59] Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. MVMoE: Multi-task vehicle routing solver with mixture-of-experts. In *International Conference on Machine Learning*, 2024.

[60] Wang Zhu, Liu Li, Long Teng, and Wen Yonglu. Multi-UAV reconnaissance task allocation for heterogeneous targets using an opposition-based genetic algorithm with double-chromosome encoding. *Chinese Journal of Aeronautics*, 31(2):339–350, 2018.

[61] Zefang Zong, Hansen Wang, Jingwei Wang, Meng Zheng, and Yong Li. Rbg: Hierarchically solving large-scale routing problems in logistic systems via reinforcement learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4648–4658, 2022.

A DiCon Backbones

Most neural routing solvers adopt a transformer encoder-decoder architecture. Given an instance $g \in \mathcal{G}$, the encoder produces node embeddings that represent the instance, and the decoder then iteratively selects actions conditioned on these embeddings. To demonstrate the plug-and-play nature of DiCon, we integrate it into two strong neural solvers, POMO [34] and RELD [28], resulting in *DiCon-P* and *DiCon-R*, respectively.

A.1 POMO

Given an instance $g \in \mathcal{G}$, we first obtain initial embeddings $H^0 = \{h_0^0, \dots, h_{|\mathcal{V}|}^0\}$ using the unified encoding scheme in Section 4.1. These embeddings are then processed by $\mathbb{L}$ encoder layers to yield the final embeddings $H^{\mathbb{L}} = \{h_0^{\mathbb{L}}, \dots, h_{|\mathcal{V}|}^{\mathbb{L}}\}$. Each encoder layer consists of an MHA sublayer, a residual Add & Norm (skip connection with instance normalization), a feed-forward (FF) sublayer, and a second Add & Norm.

Specifically, at layer $l \in \{1, \dots, \mathbb{L}\}$, the embeddings are updated by 8-head MHA followed by Add & Norm:

$$\hat{h}_i = \mathrm{IN}\big(h_i^{l-1} + \mathrm{MHA}\big(h_0^{l-1}, \dots, h_{|\mathcal{V}|}^{l-1}\big)\big), \quad \forall\, i \in \{0, \dots, |\mathcal{V}|\}. \tag{17}$$

Then, an FF sublayer and another Add & Norm produce $\{h_i^l\}_{i=0}^{|\mathcal{V}|}$:

$$h_i^l = \mathrm{IN}\big(\hat{h}_i + \mathrm{FF}(\hat{h}_i)\big). \tag{18}$$

Given the final embeddings $\{h_i^{\mathbb{L}}\}_{i=0}^{|\mathcal{V}|}$, the decoder autoregressively computes the candidate-selection distribution over T steps. At decoding step $t \in \{1, \dots, T\}$, the context embedding h_c is produced by an 8-head MHDA layer based on contextual features $v_c = [h_{\pi_1},\, h_{\pi_{t-1}}]$ and the encoder embeddings:

$$h_c \;=\; \mathrm{MHDA}\Big(v_c,\; \{\, h_0^{\mathbb{L}}, \dots, h_{|\mathcal{V}|}^{\mathbb{L}}\}\Big). \tag{19}$$

The context embedding h_c is then used to compute unnormalized compatibility scores α:

$$\alpha_i = \begin{cases} -\infty, & \text{if candidate } i \text{ is masked,} \\ C \cdot \tanh\Big(\frac{h_c^\top (W^K h_i^{\mathbb{L}})}{\sqrt{d}}\Big), & \text{otherwise,} \end{cases} \tag{20}$$

where $C = 10$ and W^K is a learnable projection. Finally, the selection probability is

$$p_\theta(\pi_t \mid \pi_{<t}, g) \;=\; \mathrm{Softmax}(\alpha). \tag{21}$$

POMO employs a multi-trajectory strategy during both training and inference by sampling multiple decoding trajectories for each instance, each initialized from a different start node. During training, the baseline is set to the mean reward (negative tour length) across sampled trajectories. At inference, trajectories are generated in parallel and the one with the minimum cost is selected, improving solution-space exploration with modest overhead.

A.2 RELD

RELD follows the same encoder as POMO but modifies the decoder to inject context information directly into the embedding space, rather than relying solely on attention weights. Concretely, it adds a residual connection from the most recently visited node $h_{\pi_{t-1}}$ to the context embedding:

$$h_c = \mathrm{MHDA}(v_c, \{h_0^{\mathbb{L}}, \dots, h_{|\mathcal{V}|}^{\mathbb{L}}\}) + h_{\pi_{t-1}}. \tag{22}$$

In addition, RELD augments the decoder with a lightweight feed-forward network and a residual connection:

$$q_c = h_c + \mathrm{FF}(h_c) = h_c + \sigma(W_2\, \sigma(W_1 h_c + b_1) + b_2)\,, \tag{23}$$

where W_1, W_2, b_1, and b_2 are learnable parameters and σ denotes the ReLU activation. Together, these modifications form a transformer-style update with a single query, enabling nonlinear processing of the query representation and allowing richer context transformations within the decoder.

Algorithm 1 Training Algorithm for DiCon

Input: Instance distribution $\mathcal{G}$, number of augmentations $\mathbb{A}$, number of training steps E, batch size B, number of tours N per subproblem;
Output: The trained policy network θ;

1: Initialize policy network θ.
2: **for** $e = 1$ to E **do**
3: $g_b \sim \text{SAMPLEINSTANCE}\ (\mathcal{G}), \quad \forall b \in \{1, \cdots, B\}$
4: $\{g_b^1, \cdots, g_b^{\mathbb{A}}\} \sim \text{DATAAUGMENTATION}\ (g_b), \quad \forall b \in \{1, \cdots, B\}$
5: $\{z_b^1, \cdots, z_b^{\mathbb{A}}\} \sim \text{PROJECTIONHEAD}\ (g_b), \quad \forall b \in \{1, \cdots, B\}$
6: $q_b^\xi \sim \text{PREDICTIONHEAD}\ (z_b^\xi), \quad \forall \xi \in \{1, \cdots, \mathbb{A}\}, \quad \forall b \in \{1, \cdots, B\}$
7: $\pi^{b,\xi,j} \sim \text{SAMPLESOLUTIONS}\ (p_\theta(\cdot|g_b^\xi)), \forall j \in \{1, \cdots, N\}, \quad \forall \xi \in \{1, \cdots, \mathbb{A}\}, \quad \forall b \in \{1, \cdots, B\}$
8: Calculate instance-level CL loss and intra-level CL loss according to Eq. (10) and Eq. (13)
9: Calculate gradient $\nabla_\theta \mathcal{L}_{total}$ according to Eq. (16)
10: $\theta \leftarrow \text{ADAM}(\theta, \nabla_\theta \mathcal{L}_{total})$
11: **end for**

B Training Algorithm

The training process is outlined in Algorithm 1. For each training iteration, we sample a batch of base instances $\{g_b\}_{b=1}^B$ (Line 3) and generate $\mathbb{A}$ augmented views for each, resulting in $\mathbb{A} \cdot B$ instances in total (Line 4). The model then computes N solutions for each view g_b^ξ using the learned policy $p_\theta(\pi^{b,\xi,j}|g_b^\xi)$ (Line 7), where $\xi \in \{1, \ldots, \mathbb{A}\}$ and $j \in \{1, \ldots, N\}$. Finally, model parameters are updated jointly using the triplet loss derived from the contrastive objectives and reinforcement loss (Lines 8-10).

C Zigzag Path Generation for Area Tasks

For training-time parameterization, we define a zigzag scanning path within each area task region to simulate a sensor-sweeping pattern, such as a UAV camera or ultrasonic probe. The length of this path is used as the coverage cost in the CGRP instance.

Rotated Rectangle Construction. Each area task is represented as a rotated rectangle with center anchor, length $\tilde{L}$, width $\tilde{W}$, and orientation β. The length $\tilde{L}$ is set as the minimal anchor distance for area tasks, while the width $\tilde{W}$ is sampled from a uniform range $[3\gamma, \tilde{L})$. The rectangle corners are computed by rotating the axis-aligned corners via the matrix $\tilde{R}(\beta)$ and translating them to the anchor point.

Zigzag Path Length Estimation. To perform coverage, we align the sweeping path along the rectangle's longer side. The number of parallel sweeps is computed as:

$$n_{\text{sweep}} = \left\lfloor \frac{\tilde{W}}{\gamma} \right\rfloor + 1, \tag{24}$$

and the total zigzag path length is estimated as:

$$L_{\text{zigzag}} = n_{\text{sweep}} \cdot \tilde{L}. \tag{25}$$

Entry–Exit Pair Sampling. For each area task, we define 4 entry-exit pairs associated with the two short edges of the rectangular region. Specifically, entry points are sampled at the endpoints of each short edge. Given the parity of n_{sweep}, the exit point is offset from the entry either on the same side or the opposite side, following the zigzag path displacement.

D Inter-task Level Contrastive Learning

We further investigate whether inter-task-level contrastive learning (CL) can help models better distinguish between heterogeneous task types. Specifically, tasks of the same type (e.g., all areas) are

Table 3: Results under different problem sizes.

Method	20	50	100
POMO	6.08	6.23	8.46
POMO_inter_task_CL	6.08	6.23	8.48

treated as positive pairs, while tasks of different types serve as negatives. We implement this strategy on top of POMO, yielding a variant termed POMO_inter_task_CL. Results on CGRP instances with 20–100 tasks are reported in Table 3. Empirically, inter-task CL brings no gains on CGRP-20 and -50, and degrades performance on CGRP-100. We hypothesize that this is due to over-clustering: forcing all latent representations of the same type together may hinder the model's ability to discriminate fine-grained variations within each task class.

E Implementation Details of the Code

To facilitate a deeper understanding of DiCon, we provide the implementation details of its core components, namely the DA mechanism and the double-level CL objectives.

E.1 Differential Attention

λ controls the strength of the subtractive branch in DA. The attention output is computed as $s_m = \left(A_m^1 - \lambda_m A_m^2\right) V$, so a larger λ_m means stronger suppression from the second attention map, while a smaller λ_m makes DA closer to standard attention. The purpose is therefore not to create a new attention map arbitrarily, but to modulate how aggressively potentially low-saliency candidates are discounted. This is consistent with the paper's description of DA as an attention-denoising mechanism for suppressing probability mass on strategically weak actions.

In our implementation, λ_m is constrained to be non-negative. Therefore, it cannot become negative during training. It is initialized to 0.5 and remains in a moderate positive range throughout optimization. Based on the logged values, the final trained values at epoch 200 are $[1.0424,\ 0.3838,\ 0.2412,\ 0.3726,\ 0.2228,\ 0.4419,\ 0.3850,\ 0.4772]$, with a mean of 0.4459. This indicates that the model does not drive λ_m to an extreme regime; rather, it learns head-dependent suppression strengths, with most heads staying below 0.5 and one head learning substantially stronger subtraction.

```
import torch
import torch.nn.functional as F

def differential_multi_head_attention(
    q_a: torch.Tensor,
    k_a: torch.Tensor,
    q_b: torch.Tensor,
    k_b: torch.Tensor,
    v: torch.Tensor,
    ninf_mask: torch.Tensor,
    lambda_broadcast: torch.Tensor,
    key_dim: int,
):
    """
    q_a, q_b: (B,H,n,D)
    k_a, k_b: (B,H,problem,D)
    v:        (B,H,problem,D)
    ninf_mask:(B,n,problem)  (here n = pomo)
    lambda_broadcast: (1,H,1,1) or (1,1,1,1)
    returns: (B,n,H*D)
    """
    B, H, n, D = q_a.shape
    problem = k_a.size(2)
```

```
    # logits: (B,H,n,problem)
    logits_a = torch.matmul(q_a, k_a.transpose(2, 3)) /
        math.sqrt(key_dim)
    logits_b = torch.matmul(q_b, k_b.transpose(2, 3)) /
        math.sqrt(key_dim)

    # apply mask to both branches
    # ninf_mask: (B,n,problem) -> (B,1,n,problem)
    mask = ninf_mask[:, None, :, :].expand(B, H, n, problem)
    logits_a = logits_a + mask
    logits_b = logits_b + mask

    w_a = F.softmax(logits_a, dim=3)  # (B,H,n,problem)
    w_b = F.softmax(logits_b, dim=3)  # (B,H,n,problem)

    weights = w_a - lambda_broadcast * w_b  # (B,H,n,problem)
    out = torch.matmul(weights, v)  # (B,H,n,D)
    out = out.transpose(1, 2).contiguous().view(B, n, H * D)  #
        (B,n,H*D)
    return out
```

E.2 Contrastive Learning

```
z, q = self.model.pre_forward(reset_state)
z = z.reshape(N_aug, -1, z.shape[1], z.shape[2])
q = q.reshape(N_aug, -1, q.shape[1], q.shape[2])

z = F.normalize(z, dim=-1)
q = F.normalize(q, dim=-1)

sim_loss = 0.0
struct_loss = 0.0
cnt = 0
for i in range(N_aug):
    for j in range(N_aug):
        if i == j:
            continue
        q_view_i = q[i]  # (B, N+1, D) predictor output
        z_view_j = z[j]  # (B, N+1, D) projector output

        # Instance-level contrastive loss
        sim_loss += -(q_view_i *
            z_view_j.detach()).sum(dim=-1).mean()

        # Intra-task-level contrastive loss
        struct_loss += self.task_structure_loss(
            q_view_i, n_areas, n_lines, target_output=z_view_j
        )
        cnt += 1

loss_siam = sim_loss / cnt
loss_struct = struct_loss / cnt
```

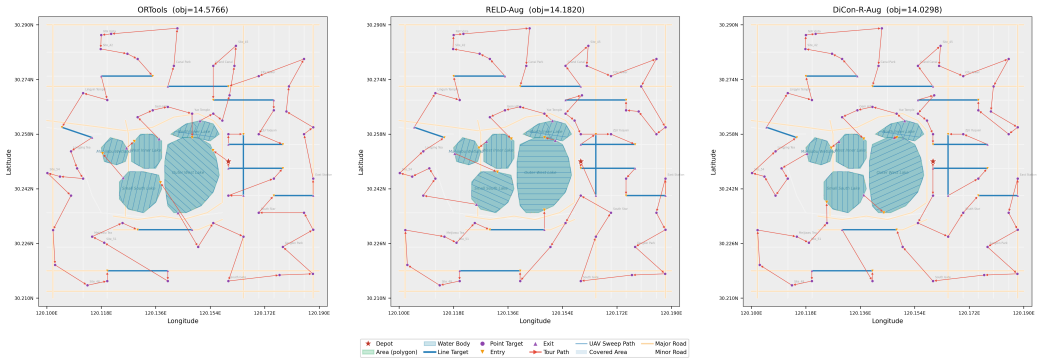


Figure 4: Visualization results on a real-world urban monitoring instance in Hangzhou, where area tasks represent water bodies (e.g., lakes), line tasks correspond to road networks, and point tasks denote key landmarks or facilities requiring localized inspection.



Figure 5: Visualization results on a real-world electric power inspection instance in Jiaxing, where the area task represents the Fenhu substation (approximately 337 m × 400 m), the line tasks correspond to seven transmission lines, and the point tasks represent 20 high-voltage transmission towers.

F Real-World Scenarios

We include a *heterogeneous urban monitoring scenario*, shown in Fig. 4, where area tasks correspond to water bodies requiring surface inspection, line tasks represent road networks for infrastructure monitoring, and point tasks denote critical facilities or landmarks requiring localized inspection. All locations were obtained from a real-world map of Hangzhou, China. The evaluation is conducted in a zero-shot manner, the area tasks are irregular polygons rather than rectangles, and the coverage path is generated by a coverage path planning (CPP) routine [3]. The candidate set is expanded to include the eight shortest entry–exit pairs (i.e., $\omega = 8$) induced by the CPP path. In addition, we include a real-world experiment on *electric power line inspection*, and the results are shown in Fig. 5. We selected the area surrounding the Fenhu 500 kV / 220 kV Substation in Jiaxing, China, where we constructed a real-world CGRP instance. All geographic data were obtained from real annotations in OpenStreetMap.

On both real-world instances, DiCon-R achieves better performance than its backbone model, RELD, and also outperforms OR-Tools. Notably, this test setting differs from the synthetic training setup in region shape, coverage pattern, and candidate cardinality. The two instances therefore exhibit substantial distribution shifts from the training data, further demonstrating the practical applicability and robustness of DiCon.

G Visualization Results

We provide qualitative visualizations of randomly selected CGRP instances in Fig. 6 to highlight the effectiveness of DiCon in handling diverse task geometries.

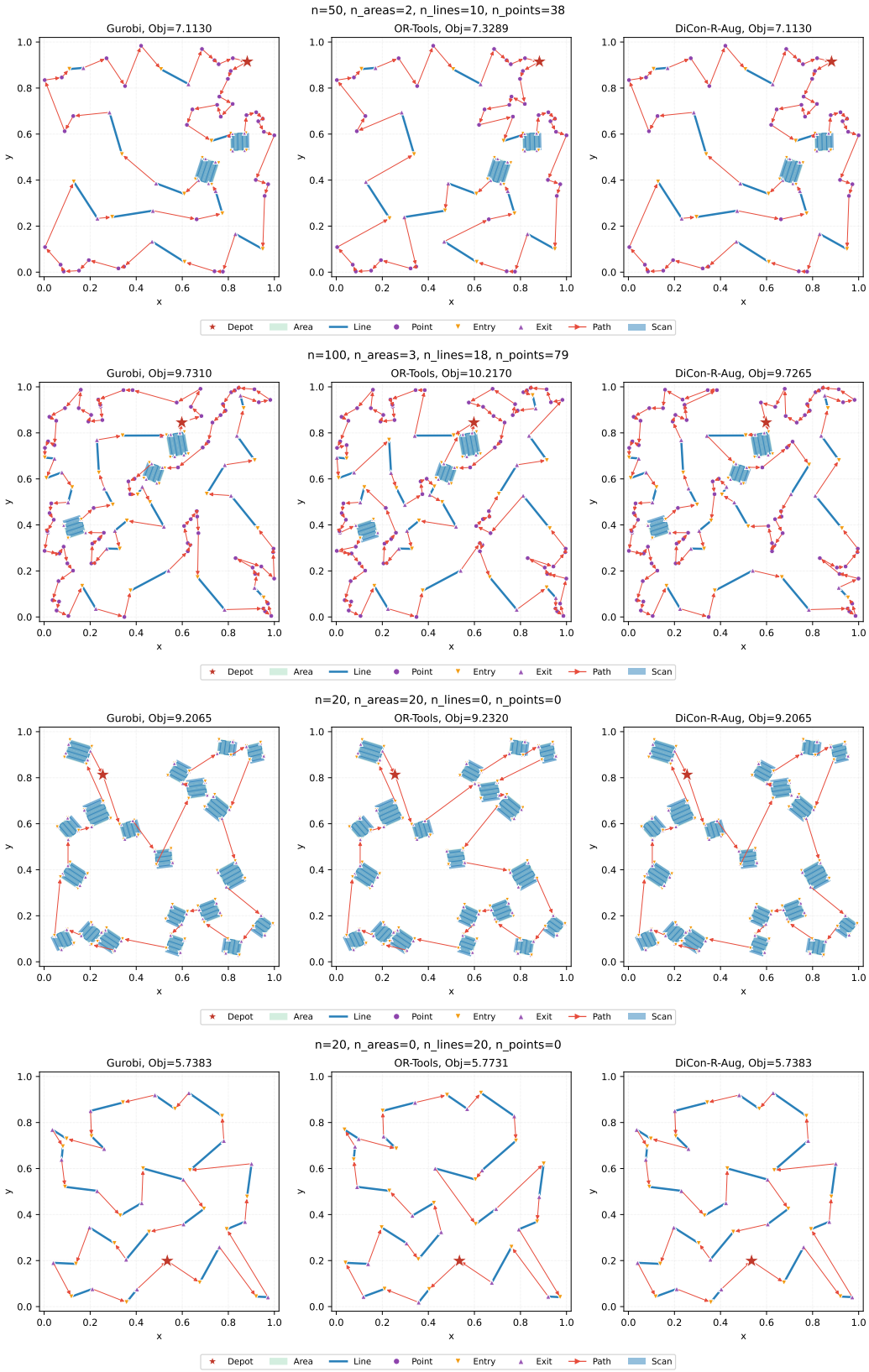


Figure 6: The visualizations of randomly selected CGRP instances.

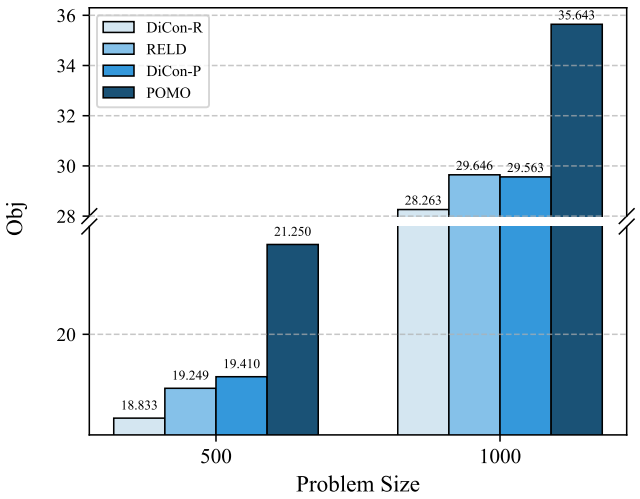


Figure 7: Comparative results on large problem sizes.

Table 4: Generalization results on extreme and highly imbalanced task mixtures.

Algorithm	Cluster Node		Long Line		Large Area	
	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)
OR-Tools	4.9525	9.39	8.8544	30.35	20.9451	33.15
RELD	4.9091	0.72	8.6683	0.55	20.8315	0.56
DiCon-R	4.8927	0.52	8.6503	0.55	20.8038	0.55

H Performance on Large-Size Instances

Figure 7 shows the results on large-size instances, with 20 test instances for each setting. DiCon consistently improves both backbone models. On 500-task instances, DiCon-P reduces the objective of POMO from 21.250 to 19.410, while DiCon-R improves RELD from 19.249 to 18.833. On 1000-task instances, the gains are more substantial: DiCon-P reduces the objective of POMO from 35.643 to 29.563, and DiCon-R improves RELD from 29.646 to 28.263. These results suggest that DiCon generalizes effectively beyond the training range and provides stronger scalability on large CGRP instances.

I Results on Imbalanced CGRP

We further assess DiCon on three types of extreme instances: clustered-node instances, long-line instances, and large-area instances. The results, reported in Table 4 and visualized in Fig. 8, show that DiCon generally maintains a performance advantage under these challenging geometric patterns.

J Performance on Generalized TSP

To further evaluate the applicability of DiCon beyond CGRP, we conduct experiments on Generalized TSP (GTSP), where each cluster contains multiple candidate nodes and the solver must select one node from each cluster while optimizing the tour. The results are reported in Table 5, with each setting evaluated on 200 instances. DiCon-R achieves the best objective values on GTSP50 and GTSP100, with objectives of 1.6494 and 2.1174, respectively. On GTSP20, DiCon-R also remains highly competitive, achieving an objective of 1.1422, close to the Gurobi result of 1.1372. Compared with RELD, DiCon-R consistently improves solution quality across all tested sizes. Meanwhile, DiCon-R maintains fast inference, requiring only 0.09s, 0.10s, and 0.15s for GTSP20, GTSP50, and GTSP100, respectively. These results indicate that DiCon-R can generalize effectively to GTSP and provides a favorable balance between solution quality and computational efficiency.

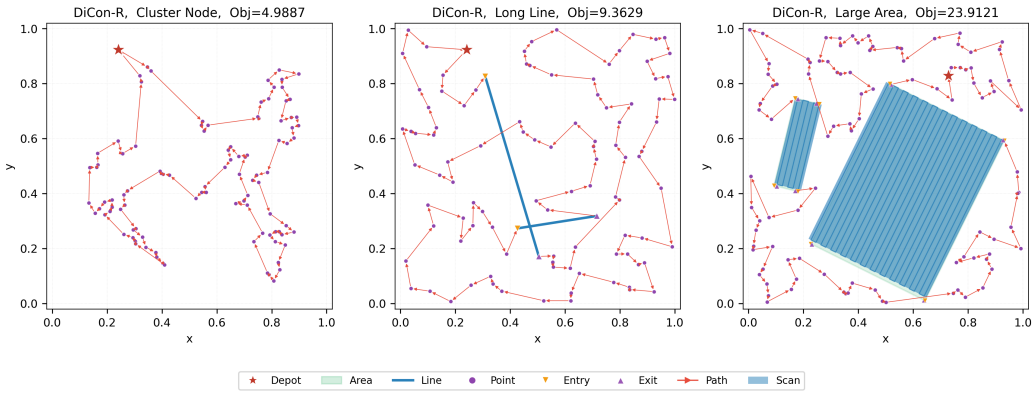


Figure 8: The visualization results on extreme and highly imbalanced task mixtures. (1) Clustered-node instances, where point tasks follow a clustered distribution that differs substantially from the training distribution; (2) Long-line instances, which contain two significantly longer line tasks; and (3) Large-area instances, which contain two substantially larger area tasks.

Table 5: Experimental results on Generalized TSP.

Method	GTSP20		GTSP50		GTSP100	
	Obj	Time	Obj	Time	Obj	Time
Gurobi	1.1372	49.42s	1.6500	8.11h	2.3147	33.33h
OR-tools	1.1427	0.60s	1.7403	4.00s	2.4722	23.80s
ALNS	1.2026	44.79s	1.8733	1.03m	2.5760	2.41m
RELD	1.1433	0.08s	1.6828	0.08s	2.1806	0.13s
DiCon-R	1.1422	0.09s	1.6494	0.10s	2.1174	0.15s

K Performance on Distance Constrained CGRP

In principle, DiCon can be extended to CGRP with more complex constraints by augmenting the encoder input with additional resource-related dimensions or introducing constraint-specific masking schemes. Since distance constraints capture a practically important deployment limitation, namely that inspection robots are often constrained by battery capacity, we focus on a distance-constrained variant of CGRP. The experimental results on distance-constrained CGRP (DCCGRP), reported in Table 6 and visualized in Fig. 9, show that our methodological contributions remain effective under this constraint setting.

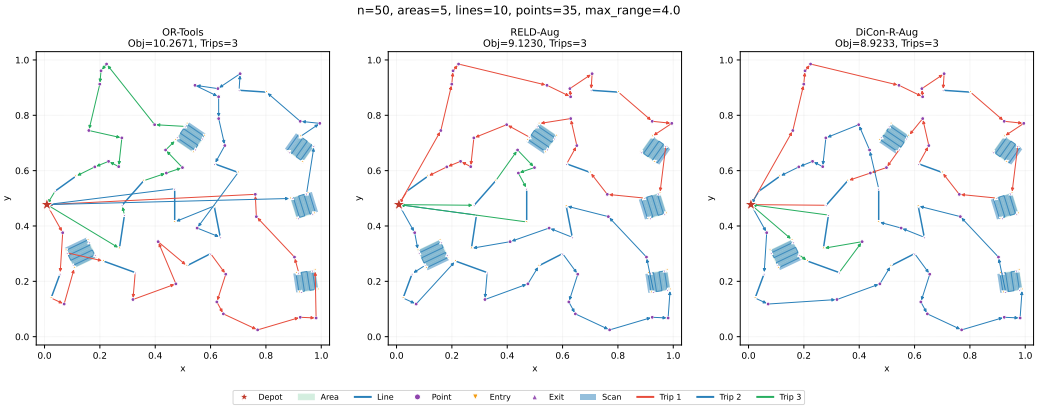


Figure 9: Performance comparison on DCCGRP

.

Table 6: Performance comparison on DCCGRP.

Algorithm	$n = 20$		$n = 50$		$n = 100$	
	Obj	Time	Obj	Time	Obj	Time
OR-Tools	6.7647	6.00s	7.1064	1.9m	9.7524	23.2m
RELD	6.5809	0.09s	6.6688	0.12s	9.0431	0.41s
DiCon-R	6.5780	0.09s	6.6470	0.13s	9.0197	0.49s

L Comparison to TSP-Trained POMO

In principle, CGRP with non-point tasks can be reformulated as a node-centric problem by flattening each entry–exit pair into 2D nodes, allowing a TSP-trained solver such as POMO to be applied in a zero-shot manner. However, this reformulation requires a modified masking scheme to enforce feasibility. To examine this issue, we consider two node-centric variants. POMO-FZ applies a TSP-trained POMO in a zero-shot setting after flattening entry–exit pairs into nodes, while POMO-FR is trained on the same flattened reformulation. The results in Table 7 show that both variants are clearly inferior to POMO trained directly on CGRP under our Unified Encoding Scheme and further behind DiCon-P. This evidence suggests that zero-shot transfer from TSP is possible only after an ad hoc reformulation, but it does not resolve the core modeling mismatch. Instead, the results support the necessity of developing neural solvers for CGRP, which is a superclass of classical routing problems such as TSP, and confirm the importance of the Unified Encoding Scheme for representing compositional geometry tasks.

Table 7: Performance comparison among zero-shot TSP-trained POMO (POMO-FZ), CGRP-trained POMO (POMO-FR), CGRP-adapted and trained POMO (POMO), and DiCon-P.

Method	100			200			Line20			A20L30			A20L80		
	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
POMO-FZ	8.544	2.27	0.26s	14.577	6.58	1.41s	4.426	1.98	0.04s	13.017	3.94	0.33s	17.387	5.31	1.16s
POMO-FR	8.557	2.43	0.26s	14.519	6.16	1.39s	4.355	0.33	0.05s	13.000	3.80	0.34s	17.459	5.75	1.17s
POMO	8.465	1.32	0.19s	14.115	3.20	0.94s	4.347	0.16	0.03s	12.504	-0.16	0.16s	16.516	0.03	0.53s
DiCon-P	8.406	0.63	0.16s	13.949	1.99	0.80s	4.343	0.07	0.04s	12.470	-0.43	0.23s	16.390	-0.73	0.80s

Table 8: Comparison of CGRP instances with varying non-Point task proportions.

Methods	0% (TSP)			25%			50%			75%			100%		
	Obj	Gap(%)	Time	Obj	Gap(%)	Time	Obj	Gap(%)	Time	Obj	Gap(%)	Time	Obj	Gap(%)	Time
LKH3	10.629	0.00	7.43m	15.815	0.00	18.08m	18.131	0.00	13.76m	19.542	0.00	14.79m	20.961	0.00	14.07m
OR-Tools	11.054	4.00	10.00m	16.350	3.39	10.01m	18.751	3.42	10.01m	20.299	3.87	10.01m	21.721	3.62	10.03m
ALNS	10.974	3.25	10.06m	16.260	2.82	10.05m	18.435	1.68	10.05m	19.845	1.55	10.08m	21.323	1.73	10.05m
POMO	11.018	3.66	0.62s	16.297	3.05	0.28s	18.766	3.50	0.39s	20.504	4.92	0.47s	22.296	6.37	0.52s
Dicon-P	10.912	2.66	0.46s	16.064	1.58	0.33s	18.387	1.41	0.41s	19.979	2.23	0.54s	21.578	2.94	0.61s
RELD	10.849	2.07	0.54s	16.022	1.31	0.28s	18.327	1.08	0.31s	19.864	1.64	0.39s	21.384	2.02	0.46s
Dicon-R	10.819	1.79	0.59s	15.904	0.56	0.36s	18.195	0.35	0.44s	19.661	0.61	0.57s	21.180	1.04	0.66s
POMO-Aug	10.910	2.64	4.97s	16.088	1.73	1.86s	18.514	2.12	3.01s	20.266	3.70	3.50s	22.032	5.11	3.63s
Dicon-P-Aug	10.823	1.83	3.99s	15.947	0.84	2.56s	18.241	0.61	3.83s	19.792	1.28	4.72s	21.384	2.02	5.33s
RELD-Aug	10.784	1.46	3.00s	15.906	0.58	2.20s	18.159	0.16	2.41s	19.700	0.81	3.87s	21.198	1.13	4.24s
Dicon-R-Aug	10.750	1.14	4.03s	15.839	0.15	2.90s	18.057	-0.40	4.12s	19.558	0.08	5.04s	21.007	0.22	5.67s

M Impact of Non-Point Task Proportion on CGRP Solvers

To examine solver robustness under varying task geometries, we generate CGRP instances with 200 tasks and vary the proportion of non-point tasks (i.e., lines or areas) from 0% (pure TSP) to 100%, with increments of 25%. For each setting, we sample 20 instances and report results in Table 8. DiCon-R and DiCon-P consistently outperform the neural baselines RELD and POMO across all settings, and the performance gap widens as the proportion of non-point tasks increases—highlighting the geometric adaptability of our model. OR-Tools remains stable, while ALNS improves with higher non-point proportions and surpasses RELD when the ratio exceeds 75%, demonstrating its flexibility in handling non-point tasks. Overall, DiCon shows strong compositional generalization and is especially effective in non-standard routing scenarios.

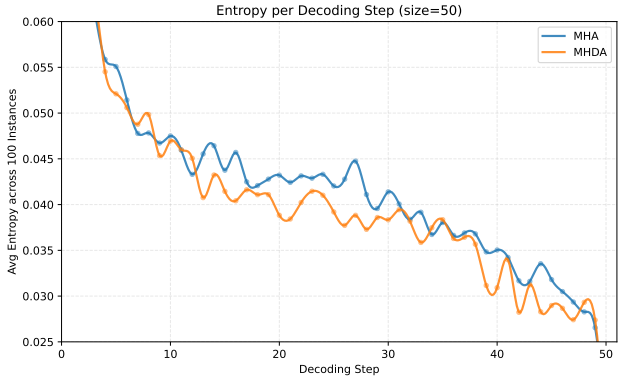


Figure 10: Entropy comparison between MHA and MHDA.

N Entropy of MHA and MHDA

To further demonstrate the effectiveness of differential attention (DA), we compare standard MHA with the proposed multi-head differential attention (MHDA) integrated into the RELD backbone. Specifically, we compute the output entropy at each decoding step on 100 randomly sampled CGRP50 instances for both RELD (with MHA) and RELD with MHDA. As illustrated in Fig. 10, MHDA consistently yields lower entropy across the decision steps compared to MHA, indicating more confident and focused policy outputs. The reduction is generally consistent, with an absolute entropy drop of about 0.001–0.004 for most steps and occasionally close to 0.005 in later decoding stages, relative to a baseline entropy level of approximately 0.03–0.06. This observation aligns with the intended effect of DA, which adaptively amplifies semantically critical ones, thereby enhancing decision consistency across different geometry-task compositions.

Table 9: The decoder parameter amount in RELD and DiCon-R.

Component	RELD	DiCon-R	Difference
Decoder Total Params	213,760	262,920	+49,160

O Efficiency Analysis of DA

We analyzed the parameter overhead of DA and reported the statistics in Table 9. DA adds about 23% more parameters to the decoder relative to RELD. However, since RELD itself uses a lightweight decoder, this corresponds to only 3.5% additional parameters for the entire model. Moreover, Table 10 shows that DA introduces only a slight runtime increase.

More importantly, the key issue is not only how many parameters are added, but also where and how they are introduced. To examine whether the gain simply comes from a larger decoder, we introduce a control baseline, RELD_double, whose decoder parameter budget is matched to RELD_DA. Specifically, RELD_double adds an adapter consisting of a three-layer MLP with a hidden dimension of 128, together with a head-wise gate after the feed-forward layer and a residual connection. As reported in Table 10, RELD_double consistently underperforms RELD_DA and is even worse than RELD on CGRP50.

P Experimental Details

P.1 Hyperparameter of Conventional Baselines

LKH3. The LKH-3 heuristic leverages the open-source LKH-3 solver, a state-of-the-art implementation for solving large-scale TSP variants using Lin-Kernighan style k-opt moves with candidate edge pruning and sequential search. The algorithm employs a multiple restart strategy (we set it to 20 in the main experiments) that initiates independent alternating optimization cycles from different

Table 10: Results of POMO, RELD, and their DA versions.

	100			200			Line20			A20L30			A20L80		
	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
POMO	8.465	1.32	0.19s	14.115	3.20	0.94s	4.347	0.16	0.03s	12.504	-0.16	0.16s	16.516	0.03	0.53s
POMO_DA	8.431	0.92	0.16s	14.024	2.54	0.79s	4.345	0.12	0.03s	12.493	-0.25	0.23s	16.463	-0.28	0.43s
RELD	8.406	0.63	0.26s	13.922	1.79	1.17s	4.345	0.12	0.03s	12.461	-0.50	0.19s	16.370	-0.85	1.01s
RELD_DA	8.403	0.59	0.36s	13.892	1.57	1.74s	4.342	0.05	0.05s	12.456	-0.54	0.27s	16.360	-0.91	1.32s
RELD_double	8.405	0.62	0.27s	13.898	1.61	1.25s	4.344	0.08	0.04s	12.458	-0.53	0.21s	16.371	-0.84	0.68s

Table 11: Results of LKH3-20, LKH3-100, LKH3-1000, DiCon-R, and DiCon-R-Aug. Here, 20 denotes the number of restarts.

Methods	100			200			Line20			A20L30			A20L80		
	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time	Obj	Gap (%)	Time
LKH3-20	8.354	0.00	50.30m	13.677	0.00	3.32h	4.340	0.00	8.57m	12.524	0.00	16.00m	16.510	0.00	43.63m
LKH3-100	8.346	-0.10	3.51h	13.631	-0.34	13.45h	4.339	-0.02	18.17m	12.449	-0.60	40.53m	16.401	-0.66	2.57h
LKH3-1000	8.345	-0.11	6.67h	13.627	-0.37	16.69h	4.339	-0.02	1.66h	12.385	-1.11	6.62h	16.318	-1.16	16.67h
DiCon-R	8.388	0.41	0.31s	13.845	1.23	1.83s	4.341	0.02	0.05s	12.437	-0.70	0.27s	16.303	-1.25	1.37s
DiCon-R-Aug	8.363	0.10	4.01s	13.755	0.57	20.95s	4.339	-0.02	0.31s	12.385	-1.11	3.66s	16.217	-1.78	10.87s

starting configurations. Each restart consists of three stages: node selection, route optimization, and node choice refinement. The first restart uses a deterministic greedy nearest-neighbor strategy for node selection to ensure a high-quality baseline, while subsequent restarts introduce increasing randomization to explore diverse regions of the solution space. Given the selected nodes, the algorithm constructs an ATSP subproblem with explicit distance matrices (exit-to-entry Euclidean distances) and invokes LKH-3 to solve the routing order, exploiting its powerful 5-opt sequential moves and sensitivity-based candidate edge filtering. Following route optimization, a node choice refinement pass fixes the visiting sequence and re-evaluates all candidate nodes for each task to minimize local travel cost, iterating for up to 5 passes. The alternating optimization between LKH-3 routing and node choice refinement repeats for 3 iterations within each restart, allowing the two decision components to mutually improve. The algorithm retains the solution with the minimum objective value across all independent restarts. We further evaluate the effect of the number of restarts. The results are reported in Table 11. As the number of restarts increases, the solution quality improves, but the runtime also increases substantially.

OR-Tools. We employ Google's OR-Tools Routing Library (v9.15) to solve the HTTSP. The problem is modeled as a single-vehicle routing problem over all $M + 1$ candidate nodes (including the depot). To handle the heterogeneous node selection, we formulate *disjunction constraints*: for each task i with candidate node set $\mathcal{V}_i$, we add a disjunction group with maximum cardinality 1, requiring exactly one node from $\mathcal{V}_i$ to be visited. A sufficiently large penalty ($2M \times 10^5$) is assigned to each disjunction to ensure full task coverage. The asymmetric travel cost matrix is defined by the Euclidean distance from each node's exit point to the next node's entry point, scaled to integer precision with a factor of 10^5 as required by the solver's integer arithmetic. The initial solution is constructed using the `PATH_CHEAPEST_ARC` strategy, which greedily extends the route by selecting the minimum-cost arc from the current node. The solution is then improved by the `Guided_Local_Search (GLS)` metaheuristic, which augments the objective with penalties on arcs that frequently appear in local optima, enabling the solver to escape plateaus. The local search phase internally combines multiple neighborhood operators including 2-opt, Or-opt, Relocate, and Cross-exchange moves, managed by the solver's C++ engine.

ALNS Algorithm. The ALNS operates for a maximum of 500 iterations to balance exploration and exploitation within the given time budget. The simulated annealing acceptance criterion begins with an initial temperature set to $\mathbb{T}_0 = 0.05 \times f(\hat{s}_0)$, where $f(\hat{s}_0)$ denotes the objective value of the initial solution, providing moderate diversification at the start, and employs a geometric cooling schedule with rate $\tau = 0.98$ such that $\mathbb{T}_{k+1} = \tau \cdot \mathbb{T}_k$ to gradually intensify the search toward high-quality regions. The destroy ratio is configured at $\rho = 0.3$, removing approximately 30% of nodes in each iteration to maintain a balance between preserving solution structure and enabling substantial modifications. The algorithm employs three destroy operators: random removal for unbiased exploration, worst removal targeting high-cost nodes, and related removal, which selects a seed node and removes its spatially proximate neighbors to enable regional route reconfiguration. Two repair operators are implemented: greedy insertion for quick feasible reconstruction and regret-based insertion for quality-conscious restoration. Operator selection follows a roulette wheel mechanism based on adaptive weights that are recalibrated every 100 iterations using a reaction factor of $\phi = 0.1$

according to $w_i^{new} = (1-\phi)w_i^{old} + \phi\bar{s}_i$, where $\bar{s}_i$ represents the average score of operator i, gradually favoring operators that consistently discover improved solutions. To further exploit the flexibility of heterogeneous task structures, the algorithm invokes node choice optimization every 10 iterations to refine entry-exit point selections for line and area tasks while maintaining computational efficiency throughout the search process.

Table 12: Results with different augmentation settings.

	20	50	100	150	200	300
$\mathbb{A}$=2	6.0726	**6.2073**	8.3921	10.6693	13.8602	15.7547
$\mathbb{A}$=3	6.0728	6.2079	8.3923	10.6633	13.8512	15.7267
$\mathbb{A}$=4 (Ours)	**6.0722**	6.2090	**8.3879**	**10.6594**	**13.8449**	**15.7022**

Table 13: Results with different λ_{ins} settings.

	20	50	100	150	200	300
λ_{ins}=0.05	6.0744	**6.2088**	8.3889	10.6581	13.8396	15.7209
λ_{ins}=0.10 (Ours)	**6.0722**	6.2090	**8.3879**	**10.6594**	**13.8449**	**15.7022**
λ_{ins}=0.20	6.0733	6.2091	8.3898	10.6552	13.8380	15.7391

P.2 Hyperparameter Study

We evaluate the sensitivity of DiCon to key hyperparameters, including the number of augmentations $\mathbb{A}$, the instance-level contrastive loss weight λ_{ins}, and the intra-task contrastive loss weight λ_{intra}.

Effect of Augmentation Number $\mathbb{A}$. Table 12 shows that increasing $\mathbb{A}$ improves performance consistently. Using $\mathbb{A}$ achieves the best overall results across all instance sizes, supporting our claim that strong data augmentations help the model learn geometry-invariant representations.

Effect of Instance-Level Contrastive Loss Weight λ_{ins}. As shown in Table 13, setting λ_{ins}=0.10 yields the best performance. A smaller value (e.g., 0.05) leads to insufficient contrastive pressure, while a larger one (e.g., 0.20) may distort the optimization landscape.

Effect of Intra-Task Contrastive Loss Weight λ_{intra}. Table 14 indicates that λ_{intra}=0.02 performs best overall. Similar to λ_{ins}, both smaller and larger values degrade performance, suggesting that balancing intra-task discrimination is key to generalizable representations.

Table 14: Results with different λ_{intra} settings.

	20	50	100	150	200	300
λ_{intra}=0.01	6.0724	6.2080	8.3905	**10.6498**	**13.8433**	15.7511
λ_{intra}=0.02 (Ours)	**6.0722**	6.2090	**8.3879**	10.6594	13.8449	**15.7022**
λ_{intra}=0.03	6.0734	**6.2066**	8.3886	10.6578	13.8473	15.7137

Q Broader Impacts

DiCon may provide positive societal value by improving the efficiency and scalability of routing systems under the unified CGRP formulation, which covers point-only, line-only, area-only, and hybrid task geometries. These settings arise in logistics, robotic inspection, disaster response, infrastructure monitoring, and agricultural surveying. By handling these routing scenarios within a unified neural solver, DiCon can reduce problem-specific redesign and support faster planning in large or compositionally novel environments. Its differential attention and double-level contrastive learning components further improve decision-making and representation learning, helping produce high-quality routes with lower computational cost than exact or traditional heuristic solvers.

DiCon should be used as a decision-support tool rather than a replacement for human oversight. In safety-critical applications, generated routes should be validated against operational constraints, environmental uncertainty, and domain-specific safety requirements before deployment.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Main contributions and scope are accurately reflect in Abstract and Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Justification: The limitations are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not include new theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Justification: The paper fully discloses all the information needed to reproduce the main experimental results in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [No]

Justification: We promise that the source code and data will be publicly released with the MIT License upon publication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The detailed experimental setups are presented in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Error bars are not reported because it would be too computationally expensive.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Experiments compute resources are indicated in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: This paper respects the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The potential positive societal impacts and negative societal impacts of this work are discussed in Appendix Q.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Licenses for existing assets are detailed in Appendix **??**.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.