# Learning with Foresight: Enhancing Neural Routing Policy via Multi-Node Lookahead Prediction

**Xia Jiang**[1] , **Yaoxin Wu**[1*] , **Yew-Soon Ong**[2,3] , **Yingqian Zhang**[1]
[1]Eindhoven University of Technology
[2]Nanyang Technological University
[3]Agency for Science, Technology and Research (A*STAR)
x.jiang1@tue.nl, wyxacc@hotmail.com, asysong@ntu.edu.sg, yqzhang@tue.nl

## Abstract

Neural policies have shown promise in solving vehicle routing problems due to their reduced reliance on handcrafted heuristics. However, current training paradigms suffer from a fundamental limitation: they primarily focus on next-node prediction for solution construction, resulting in myopic decision-making that undermines long-horizon planning capacity. To this end, we introduce Multi-node Lookahead Prediction (MnLP), a novel training strategy that extends the supervised learning paradigm to predict multiple future nodes simultaneously. We incorporate causal and discardable MnLP modules that operate exclusively during training, facilitating models to anticipate multi-step decisions while preserving inference-time efficiency. By incorporating multi-depth auxiliary supervision into the loss function, MnLP equips neural policies with the ability of long-range contextual understanding. Experimentally, MnLP outperforms existing training methods, improving the generalization capability of neural policies across various problem sizes, distributions, and real-world benchmarks. Moreover, MnLP can be seamlessly integrated into diverse neural architectures without introducing additional inference overhead.

## 1 Introduction

Vehicle routing problems (VRPs), represented by the travelling salesman problem (TSP) and the capacitated vehicle routing problem (CVRP), appear in many real-world industrial scenarios, such as logistics planning [Sar and Ghadimi, 2023], circuit design [Brophy and Voigt, 2014], and robotic systems [Bullo *et al.*, 2011]. Due to their NP-hard nature, solving VRPs (particularly at large scales) requires a trade-off between computational efficiency and solution quality. This challenge is traditionally tackled by heuristics or metaheuristics. However, designing high-quality heuristics typically requires considerable domain expertise and extensive parameter tuning, limiting their practical applicability.

Recently, there has been a growing research interest in developing neural routing policies, which leverage neural networks to learn heuristics directly from data. Specifically, the Transformer architecture [Vaswani *et al.*, 2017] is commonly used in constructive neural routing policies, which select nodes from the VRP instance sequentially and construct solutions (i.e., tours) in a step-by-step manner [Kool *et al.*, 2019; Hua *et al.*, 2025]. They offer high computational efficiency during inference and can match or even surpass the performance of some traditional algorithms [Hottung *et al.*, 2025b].

Training neural routing policies is typically accomplished using either reinforcement learning (RL) [Kool *et al.*, 2019; Kwon *et al.*, 2020; Hottung *et al.*, 2025a] or supervised learning (SL) [Luo *et al.*, 2023; Drakulic *et al.*, 2023]. Recent studies have shown that SL approaches exhibit significantly better generalizability on large-scale VRP instances compared to RL methods [Luo *et al.*, 2023; Drakulic *et al.*, 2023], highlighting the potential of SL in developing scalable and generalizable neural routing policies. Current SL-based methods, such as the Light-Encoder-and-Heavy-Decoder (LEHD) model, are typically trained using cross-entropy loss to predict the next node, conditioned on the previously selected node [Luo *et al.*, 2023]. However, constructing VRP solutions constitutes a long-horizon planning task, often involving hundreds or thousands of nodes. The standard next-node prediction paradigm risks myopic decisions, as the model may fail to capture longer-range context crucial for route construction, thereby leading to a local optimum.

In this paper, we propose a multi-node lookahead prediction (MnLP) strategy for SL routing policies. Specifically, in addition to the standard decoding head that predicts next node, we introduce causal and discardable MnLP modules tasked with predicting future nodes. An auxiliary loss is designed for training these MnLP modules, thereby implementing the lookahead mechanism to enhance the representation learning of the main model. During the training stage, this mechanism encourages the model to account for long-term impact of its decisions by explicit multi-step predictions. At inference time, the MnLP modules are removed, and only the next-node head is used, ensuring no additional computational overhead compared to standard architectures. To demonstrate the effectiveness of MnLP, we conducted extensive experiments and proved its benefits in enhancing neural policies.

Our main contributions are summarized as follows: 1) We

*Yaoxin Wu is the corresponding author

propose a novel architectural design for multi-node lookahead prediction in neural routing policies. The proposed MnLP modules operate exclusively during training, thereby introducing no additional computational overhead during inference; 2) We introduce a multi-depth auxiliary loss that guides the SL process to improve the model's long-horizon planning capability. This enables the learned policy to anticipate future decisions beyond next-node prediction; 3) We conduct extensive experiments demonstrating that our proposed training strategy can effectively enhance neural routing policies in terms of cross-size and cross-distribution generalization, as well as performance on real-world benchmarks.

## 2 Related Work

### 2.1 Neural Routing Policy

Neural routing policies constitute an important area within neural combinatorial optimization (NCO), and can be categorized into improvement and construction methods.

Improvement methods start from an initial solution and iteratively refine it through a neural network, which learns to optimize sub-problems, applies local search operators, or augments the existing solvers [Wu *et al.*, 2021; Cheng *et al.*, 2023; Kim *et al.*, 2023]. In contrast, construction methods, pioneered by the Pointer Networks [Vinyals *et al.*, 2015], directly generate problem solutions from scratch. By leveraging Transformer-like architectures and RL, the Attention Model (AM) proposed by [Kool *et al.*, 2019] significantly advanced neural routing policies' performance. Building on AM, subsequent work has improved construction-based neural routing policies along several axes: test-time adaptation [Hottung *et al.*, 2022; Kim *et al.*, 2025], enhanced learning strategies [Wang *et al.*, 2024], feature refinement [Li *et al.*, 2024; Wang *et al.*, 2025], and multi-task training [Berto *et al.*, 2025; Jiang *et al.*, 2024; Liu *et al.*, 2025].

Beyond the typical Transformer-like modeling, some efforts have also explored alternative architectures. For example, LEHD [Luo *et al.*, 2023] introduced a design with a light encoder and a heavy decoder to improve generalizability. Meanwhile, there are also studies that apply large language models (LLMs) to solve VRPs [Jiang *et al.*, 2025; Yin *et al.*, 2026], leveraging the power of large generative models to tackle routing challenges.

### 2.2 Training Paradigm

The training paradigms of neural policies broadly fall into two categories: RL and SL. RL-based approaches have achieved strong performance but typically suffer from sparse, delayed rewards and low sample efficiency (particularly on large-scale instances) [Kim *et al.*, 2024]. To alleviate these limitations, prior work refines the RL pipeline via policy optimization with multiple optima (POMO) [Kwon *et al.*, 2020], meta-learning [Zhou *et al.*, 2023], and preference-based optimization [Liao *et al.*, 2025].

In parallel, SL-based methods have gained traction for their strong cross-scale generalization [Drakulic *et al.*, 2023; Luo *et al.*, 2023]. By constructing partial solutions, they restrict training to a reduced solution space and shift from full-solution construction to high-quality partial routes, thereby complementing RL by trading exploration for data efficiency and transfer. Empirically, SL markedly improves the generalizability of neural routing solvers [Luo *et al.*, 2023]. However, despite recent refinements [Yao *et al.*, 2024; Luo *et al.*, 2025], most SL methods still use teacher-forced next-node cross-entropy, which optimizes local decisions rather than sequence-level quality and remains prone to exposure bias and error accumulation. We address this issue by extending standard next-node SL to predict multiple future nodes. This process equips the neural policy with stronger foresight during solution construction, thereby improving both performance and generalizability of SL-based routing policies.

### 2.3 Multi-token Prediction

Multi-token prediction (MTP) is a training framework for LLMs in which the model predicts a block of future tokens at each step rather than only the next token, building on prior work in blockwise decoding and non-autoregressive sequence modeling [Stern *et al.*, 2018]. Increasing evidence suggests that pure next-token prediction underfits objectives requiring multi-step planning and long-horizon credit assignment [Bachmann and Nagarajan, 2024]. By supervising multiple future steps jointly, MTP encourages planful intermediate representations, mitigates myopic yet globally suboptimal behaviors induced by teacher forcing, and serves as a regularizer against overfitting to short-range patterns [Qi *et al.*, ]. Empirically, MTP outperforms next-token prediction for LLM pre-training across diverse scales and tasks [Gloeckle *et al.*, 2024; Gerontopoulos *et al.*, 2025].

DeepSeek-V3 is a prominent real-world deployment of MTP, using it as an auxiliary pre-training objective [DeepSeek-AI *et al.*, 2025]. Inspired by this design, we develop MnLP for SL-based routing policies. While conceptually related, we target VRPs rather than language modeling: MnLP omits heavy transformer blocks in MTP and trains over a feasible set that shrinks as nodes are visited instead of a fixed vocabulary, enabling the policy to anticipate multi-step decisions under an evolving decoding process.

## 3 Preliminaries

### 3.1 VRP Formulation

A VRP instance of size $n$ is defined on a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}_{i=1}^{n}$ denotes node features (including the depot and customers) and $\mathcal{E} = \{e_{i,j} \mid i, j \in \{1, \ldots, n\},\ i \neq j\}$ is the set of edges. Without loss of generality, the depot is $v_1$. Each edge $e_{i,j}$ has cost $c_{i,j} \geq 0$, forming a cost matrix $C = [c_{i,j}]$. A feasible solution is represented as a node sequence $\tau = (x_1, \ldots, x_k)$, where each $x_\ell \in [1, n]$ indexes node $v_{x_\ell} \in \mathcal{V}$. The total travel cost is:

$$C(\tau \mid \mathcal{G}) \ = \ \sum_{\ell=1}^{k-1} c_{x_\ell,\, x_{\ell+1}} + c_{x_k,\, x_1}, \tag{1}$$

where we add the wrap-around term $c_{x_k, x_1}$ because a closed tour is required. Given the feasible set $\Psi$, the objective is:

$$\tau^*(\mathcal{G}) \ = \ \arg\min_{\tau \in \Psi} \ C(\tau \mid \mathcal{G}). \tag{2}$$

The feasible set $\Psi$ encodes all problem-specific constraints. For the TSP, $\Psi$ consists of Hamiltonian cycles that visit each node exactly once. For the CVRP, $\Psi$ comprises a set of routes $\{\tau_r\}_{r=1}^{R}$, each starting and ending at $v_1$, such that every customer $i \in \mathcal{V} \setminus \{v_1\}$ with demand $d_i$ is served exactly once and the capacity constraint holds on every route:

$$\sum_{i \in \tau_r \setminus \{1\}} d_i \;\le\; D \quad \text{for all } r \in [1, R]. \tag{3}$$

We consider Euclidean VRPs, where each node has coordinates $\mathbf{z}_i \in \mathbb{R}^2$ and edge costs are $c_{i,j} = \|\mathbf{z}_i - \mathbf{z}_j\|_2$.

### 3.2 LEHD model with SL

We select the LEHD model [Luo *et al.*, 2023] as the backbone and use the proposed MnLP for its training, since: 1) LEHD is a neural policy that is typically trained by SL rather than RL; 2) unlike other SL models such as BQ-NCO [Drakulic *et al.*, 2023], which learn an unconditional decision process, LEHD generates solutions in an auto-regressive manner, conditioning each decision on previous steps. This causal factorization matches MnLP's multi-step supervision: the node predicted by the $(k-1)$-th MnLP module enters the decoding context of the $k$-th, yielding a coherent chain of future nodes. Thus, MnLP can supervise several future decisions at once, improving credit assignment across consecutive steps and reducing exposure bias compared with next-step SL paradigm.

LEHD consists of an encoder (e.g., the Shared Encoder in Figure 1), a decoder with $L-1$ attention blocks (e.g., the Main Decoder), and an output head. Given node features $\mathcal{V} = (v_1, \ldots, v_n)$ for an instance, a light encoder (a linear layer followed by an attention block) maps them to node embeddings $H^{(E)} = (\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_n^{(0)})$. Each attention block comprises a multi-head self-attention layer ($\mathrm{MultiHeadAttn}$) and a feed-forward network ($\mathrm{FFN}$). Further computational details in the attention block are provided in Appendix A.

Instead of training for constructing the complete tour with $n$ nodes, LEHD learns to sequentially construct the solution in $n_p$ steps for a partial solution $(x_1, ..., x_{n_p})$, which is sampled from the optimal complete solution. In step $t \in \{1, ..., n_p\}$, the decoding process is conditioned on the embeddings of the first selected node $\mathbf{h}_{x_1}^{(0)}$ and the node selected in the previous step $\mathbf{h}_{x_{t-1}}^{(0)}$, which together serve as the decoding context for the current step. More precisely, they interact with $H_a$ through $L-1$ attention blocks:

$$\begin{aligned}
\widetilde{H}^{(0)} &= \mathrm{concat}(W_1 \mathbf{h}_{x_1}^{(0)}, W_2 \mathbf{h}_{x_{t-1}}^{(0)}, H_a), \\
\widetilde{H}^{(1)} &= \mathrm{AttentionBlock}(\widetilde{H}^{(0)}), \\
&\cdots \\
\widetilde{H}^{(L-1)} &= \mathrm{AttentionBlock}(\widetilde{H}^{(L-2)}),
\end{aligned} \tag{4}$$

where $H_a = \{\mathbf{h}_i^{(0)} \mid i \in \{1, \ldots, n_p\} \setminus \{x_1, \ldots, x_{t-1}\}\}$ and $W_1, W_2$ are learnable matrices. The output head of the LEHD model incorporates an attention block, a linear layer with a learnable matrix $W_O$, and a Softmax function to calculate the probabilities $\mathbf{p}_t$ for selecting among all available nodes:

$$\begin{aligned}
\widetilde{H}^{(L)} &= \mathrm{AttentionBlock}(\widetilde{H}^{(L-1)}), \\
u_i &= \begin{cases} W_O \widetilde{\mathbf{h}}_i^{(L)}, & i \notin \{1, 2\} \\ -\infty, & \text{otherwise} \end{cases}, \\
\mathbf{p}_t &= \mathrm{Softmax}(\mathbf{u}),
\end{aligned} \tag{5}$$

The SL paradigm employs a cross-entropy loss: $\mathcal{L}_{lehd} = -\sum_i^m y_i \log(\mathbf{p}_{t,i})$, where $m$ is the number of available nodes; $y_i$ is a binary variable that denotes if $v_i$ is selected. However, if trained only with teacher-forced next-node SL, the model can be locally myopic because auto-regressive decoding weakens credit assignment and accumulates exposure bias, with generalization degrading as $n$ grows.

## 4 Methodology

The MnLP framework (see Figure 1) consists of a shared encoder, a main decoder for next-node prediction, and $K$ causal MnLP modules that predict $K$ future nodes during training.

More specifically, each MnLP module $k \in \{1, \ldots, K\}$ predicts node $x_{t+k}$ at decoding step $t$, leveraging the intermediate node representation produced by the preceding module. This design enforces causality across different prediction depths, meaning that each MnLP module receives contextual information from the previous one to predict the corresponding node. From these predictions, we obtain node-selection probabilities $\mathbf{p}_t^{(k)}$ and corresponding losses $\mathcal{L}_{\mathrm{MnLP}}^{(k)}$, which are combined with the main model loss to promote long-horizon planning. Notably, all MnLP modules are used solely during training and are discarded at inference time.

### 4.1 Model Architecture

The overall architecture of our model is illustrated in Figure 1. After encoding the VRP instance using the shared encoder, the main decoder and the MnLP modules perform depth-wise computation and apply task-specific output heads to select the node at depth $k$. The process is detailed below.

**Encoder.** The model uses a lightweight encoder that linearly projects raw node features $\mathcal{V} = (v_1, ..., v_n)$ (e.g., 2-D coordinates for Euclidean TSP) and then applies a single Transformer-style attention block (MultiHeadAttn + FFN, as detailed in Appendix A) to produce the node-embedding matrix $H^{(E)} = (\mathbf{h}_1^{(0)}, ..., \mathbf{h}_n^{(0)})$, such that,

$$\tilde{H}^{(E)} = \mathcal{V} W_E + \mathrm{MultiHeadAttn}(\mathcal{V} W_E), \tag{6}$$

$$H^{(E)} = \tilde{H}^{(E)} + \mathrm{FFN}(\tilde{H}^{(E)}), \tag{7}$$

where $W_E$ denotes the learnable parameter of the linear layer. The node embeddings $H^{(E)}$ are shared by the decoder and all MnLP modules for depth-wise computation.

**Depth-wise Computation.** The main decoder and all the MnLP modules compute the representations used for node selection. As depicted by Figure 1, the $k$-th MnLP module comprises two LayerNorm (LN) operations, a linear layer with a learnable matrix $W_I^{(k)}$, an FFN layer, and a task-specific output head. At depth $k = 0$, the main model predicts $x_t$ at step $t$ using the embedding of the previously selected node

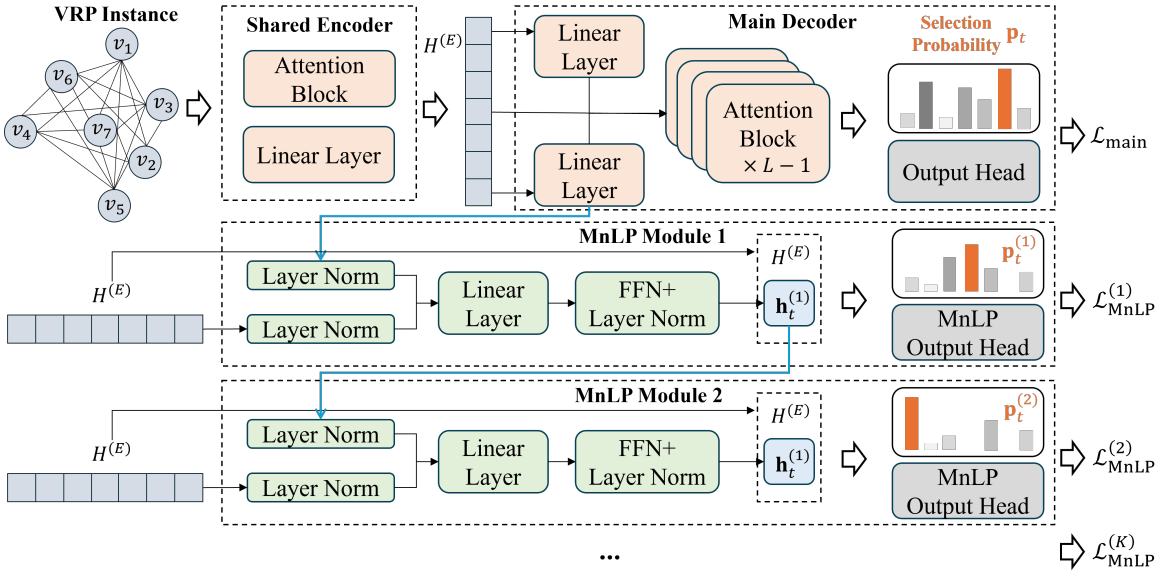


Figure 1: The overall MnLP model architecture.

$x_{t-1}$ as part of the decoding context. We generalize this process to a multi-node prediction setting: for any $k > 0$, the $k$-th MnLP module predicts node $x_{t+k}$ using an intermediate context representation $\mathbf{h}^{(k)}t$, obtained by combining 1) the representation from the $(k-1)$-th module, $\mathbf{h}^{(k-1)}t$, and 2) the embedding of the ground-truth node $x_{t+k-1}$, $\mathbf{h}^{(0)}_{t+k-1}$ (for $k = 1$, $\mathbf{h}^{(k-1)}_t$ reduces to the embedding of $x_{t-1}$). Following the normalization-then-projection design of DeepSeek-V3 [DeepSeek-AI *et al.*, 2025], we compute:

$$\mathbf{h}_t^{'(k)} = W_I^{(k)} \,\mathrm{concat}(\mathrm{LN}(\mathbf{h}_t^{(k-1)}), \mathrm{LN}(\mathbf{h}^{(0)}_{t+k-1})) \tag{8}$$

Note that Equation (8) is typically for TSP. In the case of other VRPs, we need to incorporate additional constraint information. For example, we introduce the updated vehicle capacity $C^{(k)}$ for CVRP, as the available vehicle capacity must be dynamically adjusted for each prediction depth $k > 0$. This is specified by: $\mathbf{h}_t^{'(k)} = W_I^{(k)} \,\mathrm{concat}(\mathrm{LN}(\mathbf{h}_t^{(k-1)}), \mathrm{LN}(\mathbf{h}^{(0)}_{t+k-1}), W_c C^{(k)})$, where $W_c$ is also a learnable matrix that projects $C^{(k)}$ to the embedding dimension. After that, we apply a FFN layer with LN to $\mathbf{h}_t^{'(k)}$ for enriched representation learning, such that,

$$\mathbf{h}_t^{(k)} = \mathrm{LN}(\mathrm{FFN}(\mathbf{h}_t^{'(k)})) \tag{9}$$

As shown in Figure 1, $\mathrm{OutputHead}_k$ takes input as $\mathbf{h}_t^{(k)}$ and generates probability $\mathbf{p}_t^{(k)}$, which is introduced below.

**Task-specific output head.** The output head in $k$-th MnLP module predicts the $(t+k)$-th node (i.e., $x_{t+k}$). Specifically, we use a task-specific output head $\mathrm{OutputHead}_k$, which is also a parameterized and learnable component, to allow for an interaction between $\mathbf{h}_t^{(k)}$ and the available node embeddings $H^{(E)}$. As a result, the probabilities of selecting each node at prediction depth $k$ are calculated as below,

$$\mathbf{p}_t^{(k)} = \mathrm{OutputHead}_k(\mathbf{h}_t^{(k)}, H^{(E)}) \tag{10}$$

Considering the unique problem characteristics, we design task-specific output heads for different VRP variants. For the TSP, we compute the compatibility between the intermediate MnLP representation $\mathbf{h}_t^{(k)}$ at prediction depth $k$ and the node embeddings $H^{(E)}$ simply via a dot-product operation:

$$\mathrm{logits} = W_M^{(k)}\mathbf{h}_t^{(k)} \cdot (W_E^{(k)}H^{(E)})^\top, \tag{11}$$

where $W_M^{(k)}$ and $W_E^{(k)}$ are learnable matrices. This is followed by masking (as remarked below) and a Softmax operation imposed on the feasible set $\mathcal{A}_t^{(k)}$ at each depth $k$.

**Remark 1** (Feasible set $\mathcal{A}_t^{(k)}$ at depth $k$)**.** *Let $S_p$ be the node set of a sampled partial tour of length $n_p$. At each decoding step $t$, the available nodes (i.e., feasible set) for the depth-$k$ prediction are $\mathcal{A}_t^{(k)} = \{\, i \in S_p \setminus \{x_1, \ldots, x_{t+k-1}\} \,\}$.*

We ensure feasibility by assigning $-\infty$ to the logits of all nodes not in $\mathcal{A}_t^{(k)}$ (same masking as in prior work [Kool *et al.*, 2019]), including nodes visited by earlier MnLP depths. Consequently, the size of the feasible set $|\mathcal{A}_t^{(k)}| = n_p - (t + k - 1)$ decreases with the increase of $k$. If $t+k > n_p$, we skip the computation of depth-$k$ loss. Therefore, the final selection distribution is then computed only over $\mathcal{A}_t^{(k)}$:

$$\mathbf{p}_t^{(k)}(i) = \begin{cases} \dfrac{\exp(\mathrm{logits}(i))}{\sum_{j \in \mathcal{A}_t^{(k)}} \exp(\mathrm{logits}(j))}, & \text{if } i \in \mathcal{A}_t^{(k)}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

In contrast, the output head of CVRP requires a more elaborate mechanism due to the need to predict both the next customer and the possibility of returning to the depot [Luo *et al.*, 2023]. To address this, we adopt an architecture similar to the decoder layer of the LEHD. Specifically, we first concatenate the representation $\mathbf{h}_t^{(k)}$ with $H^{(E)}$ (the embeddings of

**Algorithm 1** Training with MnLP

**Input**: episodes per epoch $E$, batch size $B$
**Param**: MnLP weight $\gamma$, warm-up epochs $W$, ratio $\alpha$
**Output**: trained model $\theta$

1: **for** epoch $e = 1, 2, \dots$ **do**
2: $\quad\gamma_e \leftarrow \gamma \cdot \min\big(1, e/(\alpha W)\big)$
3: $\quad$**for** $i = 1$ to $E/B$ **do**
4: $\quad\quad$Sample a batch of $B$ instances and reset env
5: $\quad\quad$**for** decoding step $t = 0, 1, \dots$ until termination **do**
6: $\quad\quad\quad$**if** $t \leq 1$ **then**
7: $\quad\quad\quad\quad$Select fixed nodes (e.g., depot/start)
8: $\quad\quad\quad$**else**
9: $\quad\quad\quad\quad$$(\mathcal{L}_{\text{main}}, \mathcal{L}_{\text{MnLP}}) \leftarrow \textsc{Forward}(\theta)$
10: $\quad\quad\quad\quad$$\mathcal{L} \leftarrow \mathcal{L}_{\text{main}} + \gamma\mathcal{L}_{\text{MnLP}}$
11: $\quad\quad\quad\quad$Update $\theta$ by backprop on $\mathcal{L}$
12: $\quad\quad\quad$**end if**
13: $\quad\quad\quad$StepEnv()
14: $\quad\quad$**end for**
15: $\quad$**end for**
16: **end for**
17: **return** $\theta$

available nodes): $\widetilde{H}^{(k)} = \text{concat}(\mathbf{h}_t^{(k)}, H^{(E)})$, which is then processed by an attention block to make the decoding context (i.e., $\mathbf{h}_t^{(k)}$) interact with the node embeddings. After that, the output $\widetilde{H}^{(k)}$ is used to compute the node selection probabilities at depth $k$, as formulated similarly by Equation (5).

### 4.2 MnLP Training

MnLP enhances policy learning with auxiliary supervision. As shown in Algorithm 1, each forward pass produces both the main logits for next-node prediction and lookahead logits for MnLP. The main loss $\mathcal{L}_{\text{main}}$ and auxiliary loss $\mathcal{L}_{\text{MnLP}}$ are computed in parallel (Figure 1), and parameters are updated using their weighted sum. Only the main decoder is used at inference, so test-time complexity remains unchanged.

More precisely, we extend the standard cross-entropy used in SL training to supervise predictions at multiple depths. Let $m^k = |\mathcal{A}_t^{(k)}|$ be the size of the feasible set, $\mathbf{p}_t^{(k)} \in \mathbb{R}^{m^k}$ be the predicted distribution over $\mathcal{A}_t^{(k)}$, and $y_i^k \in \{0, 1\}$ be a one-hot indicator of the node selected at depth $k$ (such that $\sum_i y_i^k = 1$). For a given $k$, the depth-$k$ MnLP loss is:

$$\mathcal{L}_{\text{MnLP}}^{(k)} = \begin{cases} -\sum_{i=1}^{m^k} y_i^k \log\big(\mathbf{p}_{t,i}^{(k)}\big), & \text{if } t{+}k \leq n_p, \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

where $n_p$ is the length of the teacher-forced partial solution. Note that when the lookahead would exceed the partial trajectory, the loss is masked out. To obtain the overall signal, we average the depth-wise losses across the $K$ MnLP modules:

$$\mathcal{L}_{\text{MnLP}} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{\text{MnLP}}^{(k)} \tag{14}$$

This auxiliary loss is then integrated into the training objective via a weighted sum with the main task loss:

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \gamma\mathcal{L}_{\text{MnLP}}, \tag{15}$$

Here, $\gamma$ is the parameter that controls the contribution of the MnLP auxiliary supervision, and $\mathcal{L}_{\text{main}}$ denotes the cross-entropy loss of the main model. To prevent MnLP from dominating early training, we linearly ramp up $\gamma$ over a warm-up period of $W$ epochs (line 2 in Algorithm 1).

We train with $\mathcal{L}$ that augments the standard next-node head with $K$ lookahead heads ($k$=1, …, $K$). At depth $k$, supervision targets the node that would appear at step $t{+}k$ under teacher forcing, but is applied already at $t$: the $k$-th MnLP module is conditioned on $x_{t+k-1}$ and its cross-entropy is computed over $\mathcal{A}_t^{(k)}$. This design shortens credit-assignment paths, strengthens long-range representations in the encoder, and provides multi-task regularization during training.

## 5 Experiments

We empirically evaluate our proposed MnLP training strategy on TSP and CVRP of various sizes and distributions, comparing it against the existing neural routing policies with different architectures and training algorithm designs.

**Problem setting** We follow standard data-generation protocols for TSP and CVRP in prior work [Kool *et al.*, 2019]. In line with the settings used by LEHD [Luo *et al.*, 2023], we use one million instances each for TSP100 (i.e., TSP instance with 100 nodes) and CVRP100 as the training sets. For evaluation, the TSP test set comprises 10,000 instances with 100 nodes, along with 128 instances each for problem sizes of 200, 500, and 1000 nodes. The CVRP test set mirrors this structure, containing the same number of instances across corresponding problem sizes. To obtain ground-truth labels, the optimal solutions for the TSP training set are computed using the Concorde solver [Cook *et al.*, 2011], while optimal solutions for the CVRP training set are generated by the Hybrid Genetic Search (HGS) solver [Vidal, 2022].

**Model and training setting** We adopt the LEHD configuration from [Luo *et al.*, 2023], which has a one-layer encoder and a decoder with 6 attention blocks. The node embedding dimension is set to 128. Each $\mathrm{MultiHeadAttn}$ layer uses 8 attention heads, while the dimension of $\mathrm{FFN}$ is 512. We set $K = 4$ for MnLP, while a sensitivity analysis of $K$ is also conducted. The MnLP weight $\gamma$ is 0.2 for TSP and 0.1 for CVRP, with $W = 5$ warm-up epochs and warm-up ratio $\alpha = 3$. TSP and CVRP models are trained for 150 and 15 epochs, respectively, with a batch size of 1024. We use Adam with an initial learning rate $10^{-4}$ and decay rates 0.97 for TSP and 0.9 for CVRP. All experiments are run on a server with an AMD EPYC 7F72 CPU and an NVIDIA H100 GPU.

**Baselines** We compare with: **(1) Classical solvers:** Concorde [Cook *et al.*, 2011], LKH3 [Helsgaun, 2017], HGS [Vidal, 2022], and OR-Tools [Furnon and Perron, 2024]; **(2) Neural routing policies with architectural designs:** ELG [Gao *et al.*, 2024], LEHD [Luo *et al.*, 2023], INViT [Fang *et al.*, 2024], DGL [Xiao *et al.*, 2025], and RELD [Huang *et al.*, 2025]; **(3) Neural routing policies with training algorithm designs:** POMO [Kwon *et al.*, 2020], which train using parallel solution trajectories, SADABL [Yao *et al.*, 2024], which implements SL with data augmentation and bidirectional loss, and BOPO [Liao *et al.*,

| | | $n = 100$ | | | $n = 200$ | | | $n = 500$ | | | $n = 1000$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Obj. | Gap | Time | Obj. | Gap | Time | Obj. | Gap | Time | Obj. | Gap | Time |
| *TSP* | Concorde | 7.763 | - | 34m | 10.704 | - | 3m | 16.522 | - | 32m | 23.120 | - | 7.8m |
| | OR-Tools | 7.947 | 2.368% | 11h | 11.091 | 3.618% | 17m | 17.296 | 4.682% | 50m | 24.249 | 4.885% | 10h |
| | POMO augx8 | 7.773 | 0.134% | 0.5m | 10.868 | 1.533% | 2.5s | 20.188 | 22.187% | 0.4m | 32.500 | 40.570% | 0.7m |
| | BOPO augx8 | 7.771 | 0.103% | 0.5m | 10.880 | 1.644% | 2.5s | 19.109 | 15.658% | 0.4m | 29.571 | 27.902% | 0.7m |
| | SA-DABL augx8 | 7.767 | 0.053% | 0.5m | 10.853 | 1.392% | 2.5s | 19.034 | 15.204% | 0.4m | 29.681 | 28.378% | 0.7m |
| | ELG augx8 | 7.781 | 0.232% | 1.3m | 10.854 | 1.401% | 3.1s | 17.714 | 7.215% | 0.2m | 25.763 | 11.432% | 0.7m |
| | INViT greedy | 7.907 | 1.855% | 1.6m | 11.079 | 3.503% | 4.7s | 17.392 | 5.266% | 13s | 24.578 | 6.306% | 0.7m |
| | DGL greedy | 7.805 | 0.531% | 1.9m | 10.839 | 1.261% | 7.7s | 16.899 | 2.282% | 19.6s | 23.739 | 2.678% | 0.6m |
| | LEHD greedy | 7.807 | 0.558% | 9.6s | 10.792 | 0.824% | 1.2s | 16.810 | 1.748% | 6.6s | 23.944 | 3.563% | 1.3m |
| | Ours (MnLP) greedy | 7.805 | 0.531% | 9.6s | 10.789 | 0.795% | 1.2s | 16.804 | 1.711% | 6.6s | 23.779 | 2.852% | 1.3m |
| | RRC 100 | 7.764 | 0.011% | 4.6m | 10.710 | 0.059% | 0.7m | 16.576 | 0.333% | 4.0m | 23.375 | 1.105% | 19m |
| | RRC 500 | 7.763 | 0.002% | 22m | 10.707 | 0.026% | 3.4m | 16.556 | 0.207% | 16m | 23.290 | 0.736% | 1.5h |
| | RRC 1000 | 7.763 | 0.001% | 42m | 10.705 | 0.017% | 6.5m | 16.550 | 0.174% | 31m | 23.268 | 0.639% | 2.7h |
| *CVRP* | LKH3 | 15.647 | - | 12h | 20.173 | - | 2.1h | 37.229 | - | 5.5h | 37.091 | - | 7.1h |
| | HGS | 15.564 | -0.533% | 4.5h | 19.946 | -1.126% | 1.4h | 36.561 | -1.794% | 4.0h | 36.289 | -2.162% | 5.3h |
| | OR-Tools | 16.616 | 6.193% | 2h | 21.564 | 6.894% | 1h | 40.698 | 9.112% | 2.2h | 41.417 | 11.662% | 3h |
| | POMO augx8 | 15.755 | 0.689% | 0.5m | 21.155 | 4.866% | 2.9s | 44.638 | 19.901% | 0.5m | 84.896 | 128.885% | 0.9m |
| | SA-DABL augx8 | 15.906 | 1.655% | 0.5m | 20.980 | 3.999% | 2.9s | 41.869 | 12.463% | 0.5m | 51.478 | 38.788% | 0.9m |
| | ELG augx8 | 15.839 | 1.227% | 1.0m | 20.699 | 2.608% | 5.0s | 39.388 | 5.799% | 0.3m | 41.548 | 12.016% | 1.0m |
| | RELD augx8 | 15.797 | 0.960% | 0.7m | 20.506 | 1.654% | 2.4s | 38.337 | 2.975% | 7.8s | 39.597 | 6.757% | 0.5m |
| | INViT greedy | 17.206 | 9.964% | 3.3m | 22.626 | 12.160% | 8.1s | 42.356 | 13.772% | 19s | 42.858 | 15.548% | 1.3m |
| | DGL greedy | 16.654 | 6.441% | 14m | 21.861 | 8.368% | 0.5m | 40.712 | 9.356% | 1.4m | 43.295 | 16.727% | 2.9m |
| | LEHD greedy | 16.233 | 3.748% | 0.4m | 20.809 | 3.156% | 2.4s | 38.359 | 3.035% | 7.2s | 39.877 | 7.513% | 1.3m |
| | Ours (MnLP) greedy | 16.196 | 3.509% | 0.4m | 20.819 | 3.206% | 2.4s | 38.319 | 2.928% | 7.2s | 39.388 | 6.195% | 1.3m |
| | RRC 100 | 15.720 | 0.471% | 5.7m | 20.245 | 0.363% | 3.2m | 37.462 | 0.627% | 6.6s | 38.151 | 2.859% | 16m |
| | RRC 500 | 15.661 | 0.094% | 26m | 20.146 | -0.136% | 4.0m | 37.231 | 0.007% | 17m | 37.693 | 1.624% | 1.3h |
| | RRC 1000 | 15.648 | 0.012% | 51m | 20.121 | -0.255% | 7.9m | 37.133 | -0.257% | 34m | 37.516 | 1.148% | 2.7h |

Table 1: Experimental results on TSP and CVRP instances, with optimality gaps computed using Concorde for TSP and LKH3 for CVRP.

2025], which uses best-anchored and objective-guided preference optimization. We retrain LEHD by following [Luo *et al.*, 2023], and use the provided open-sourced models for other baselines. During evaluation, we measure: 1) average objective values (Obj.), 2) performance gap relative to baseline solvers (Gap), and 3) total inference time (Time).

## 5.1 Main Results

Table 1 reports results on uniformly distributed routing instances of various sizes. MnLP policies consistently outperform the original greedy LEHD, with gains increasing at larger scales. For example, on TSP1000, MnLP reduces the optimality gap from 3.563% to 2.852%—a 20% relative improvement, showing that multi-step supervision enhances long-horizon decision quality. These gains arise purely from training (MnLP modules are not used at inference), isolating the benefit to representation learning. Overall, the results support our claim that supervising multiple future steps yields richer long-range context and smaller gaps, especially on large instances where myopic errors accumulate.

More broadly, MnLP can also outperform other training strategies in the domain of NCO, including POMO, preference-based optimization (i.e., BOPO), and augmentation-driven SL (i.e., SA-DABL), especially when the instance size exceeds 100. The performance gain is more pronounced on $n = 1000$ instances, suggesting that MnLP training is particularly effective for large-scale problems. Meanwhile, compared to methods that introduce additional components and computationally intensive processes (e.g., INViT and DGL), MnLP achieves better efficiency while producing more optimal solutions across most task settings.

| | Method | $n = 500$ | | $n = 1000$ | |
|---|---|---|---|---|---|
| | | $R$ | $E$ | $R$ | $E$ |
| TSP | LEHD | 2.888 | 2.821 | 5.937 | 6.397 |
| | MnLP | 2.304 | 2.447 | 4.458 | 5.189 |
| CVRP | LEHD | 4.383 | 4.558 | 8.188 | 8.972 |
| | MnLP | 4.294 | 4.518 | 7.831 | 8.843 |

Table 2: Generalization performance comparison: Average gaps (%) across 1,000 instances for rotation and explosion distributions.

Although some models (e.g., ELG) perform better on in-distribution $n = 100$ cases, MnLP remains competitive and can be strengthened via the Random Re-Construct (RRC) process from LEHD [Luo *et al.*, 2023]. With only 100 RRC steps, MnLP surpasses these compared policies. Meanwhile, we also present the comparative results between our method and LEHD with the same RRC steps in Appendix C, which shows that our model with RRC can also outperform the original model, particularly on large-scale instances, demonstrating compatibility with test-time refinement techniques.

## 5.2 Cross-distribution Generalization

Generalizability is essential for neural routing policies: models that overfit a single distribution have limited practical value [Zhou *et al.*, 2023]. Beyond the performance on uniformly distributed instances reported in Table 1, we further

| | Size | POMO aug×8 | LEHD greedy | Ours (MnLP) greedy |
|---|---|---|---|---|
| TSPLib | <100 | 0.792% | 1.064% | 0.673% |
| | 100-200 | 2.423% | 2.309% | 2.127% |
| | 200-500 | 13.413% | 4.371% | 2.414% |
| | 500-1k | 31.678% | 8.729% | 15.597% |
| | >1K | 63.810% | 13.913% | 12.252% |
| | All | 26.439% | 6.841% | 6.401% |
| | Set (size) | | | |
| CVRPLib | A (31-79) | 4.970% | 6.595% | 7.114% |
| | B (30-77) | 4.747% | 7.493% | 7.137% |
| | E (12-100) | 11.402% | 5.478% | 4.791% |
| | F (44-134) | 15.973% | 8.491% | 7.391% |
| | M (100-199) | 4.861% | 8.407% | 8.197% |
| | P (15-100) | 15.525% | 6.061% | 6.564% |
| | X (100-1k) | 21.684% | 11.178% | 9.024% |
| | All | 15.450% | 9.038% | 7.945% |

Table 3: Experimental results on TSPLib and CVRPLib.

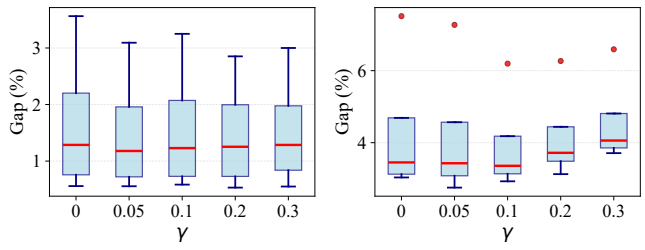


Figure 2: The distribution of optimality gap across different problem sizes for varying values of $\gamma$. (a) TSP; (b) CVRP.

evaluate the cross-distribution generalization performance of our model on two typical benchmark distributions of VRP: the *rotation distribution* ($R$) and the *explosion distribution* ($E$), both of which have been used extensively in prior work [Zhou *et al.*, 2023; Jiang *et al.*, 2023]. The precise generation protocols are given in Appendix B.

Specifically, we randomly generate 1,000 instances for each distribution with different sizes: $n = 500$ and 1000. The comparative results are summarized in Table 2. It shows that the model trained with MnLP consistently improves upon the LEHD baseline across distributions and problem sizes. Taking TSP as an example, MnLP yields substantial relative reductions in average gap: at $n = 1000$, $R$ decreases from $5.937\%$ to $4.458\%$ ($24.9\%$ improvement) and $E$ from $6.397\%$ to $5.189\%$ ($18.9\%$ improvement). These results indicate that MnLP can effectively enhance cross-distribution generalizability and robustness to distributional shift.

Furthermore, we evaluate the MnLP-trained model on the widely used real-world benchmark datasets, including TSPLib [Reinelt, 1991] and CVRPLib [Uchoa *et al.*, 2017], which reflect real-world problem distributions. The performance is evaluated on instances with Euclidean distance, and is compared against POMO and the original LEHD model in Table 3. Empirically, we observe that the MnLP improves overall performance across the benchmark datasets, showing its effectiveness on instances with various distributions.

| Size ($n$) | | 100 | 200 | 500 | 1000 |
|---|---|---|---|---|---|
| TSP | MnLP-E | 0.531% | 0.795% | 1.711% | 2.852% |
| | MnLP-D | 0.572% | 0.804% | 1.823% | 3.442% |
| CVRP | MnLP-E | 3.509% | 3.206% | 2.928% | 6.195% |
| | MnLP-D | 4.008% | 3.332% | 3.215% | 7.286% |

Table 4: The ablation study of the MnLP implementations.

### 5.3 Ablation and Sensitivity Analyses

We conduct ablation and sensitivity analyses to identify the optimal MnLP training configuration.

**Effect of the value of $\gamma$.** The impact of different values of $\gamma$ on model performance is shown in Figure 2. By varying $\gamma$ from $\{0.05, 0.1, 0.2, 0.3\}$, we observe that appropriate values of $\gamma$ lead to improved training outcomes. For instance, setting $\gamma = 0.2$ for TSP and $\gamma = 0.1$ for CVRP yields the best average performance across various problem sizes. In contrast, excessively large values (e.g., $\gamma = 0.3$) may degrade performance, as the training process becomes increasingly biased toward optimizing the MnLP modules, thereby hindering sufficient learning of the main model for next node prediction.

**Effect of different MnLP implementations.** To evaluate our design, we compare different architectural implementations. In our method, MnLP modules share the encoder with the main model. Alternatively, they can share high-level node representations by reusing the network components before the output head, where the first MnLP module takes input from the $(L-1)$-th decoder layer. We denote our encoder-sharing design as MnLP-E and the decoder-based variant as MnLP-D, and report results in Table 4. MnLP-E consistently outperforms MnLP-D, while MnLP-D sometimes even degrades performance compared with the original LEHD. This is likely because decoder representations already contain the current decoding context (e.g., the node selected at the previous step), which introduces noise for future prediction.

**Effect of other factors.** We further analyze the computational cost of MnLP, the impact of using different numbers of MnLP modules, and the role of the warm-up phase. Detailed results and discussions are provided in Appendix C.

### 5.4 Versatility Study

We evaluate the effectiveness of MnLP on another SL-based policy (i.e., SIL [Luo *et al.*, 2025]) in Appendix C, where we demonstrate its versatility on different neural architectures.

## 6 Conclusion

We propose MnLP, a training-only multi-depth supervision scheme for SL-based neural routing policies that adds a lookahead mechanism to predict future nodes, injecting long-horizon signals without changing inference or adding test-time cost. Experiments show consistent gains across cross-size, cross-distribution, and real-world benchmarks, highlighting the value of multi-step supervision for NCO.

Future work includes extending MnLP to RL for long-horizon credit assignment, applying it to other CO tasks (e.g., scheduling), and designing test-time retained multi-step predictors to improve efficiency and solution quality.

## References


[Bachmann and Nagarajan, 2024] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *41st International Conference on Machine Learning*, 2024.

[Berto *et al.*, 2025] Federico Berto, Chuanbo Hua, Nayeli 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.

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

[Brophy and Voigt, 2014] Jennifer AN Brophy and Christopher A Voigt. Principles of genetic circuit design. *Nature methods*, 11(5):508–520, 2014.

[Bullo *et al.*, 2011] Francesco Bullo, Emilio Frazzoli, Marco Pavone, Ketan Savla, and Stephen L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[Cheng *et al.*, 2023] Hanni Cheng, Haosi Zheng, Ya Cong, Weihao Jiang, and Shiliang Pu. Select and optimize: Learning to solve large-scale tsp instances. In *International Conference on Artificial Intelligence and Statistics*, pages 1219–1231, 2023.

[Cook *et al.*, 2011] William J Cook, David L Applegate, Robert E Bixby, and Vasek Chvátal. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[DeepSeek-AI *et al.*, 2025] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report, 2025.

[Drakulic *et al.*, 2023] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[Fang *et al.*, 2024] Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. INViT: A generalizable routing problem solver with invariant nested view transformer. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 12973–12992, July 2024.

[Furnon and Perron, 2024] Vincent Furnon and Laurent Perron. Or-tools routing library, 2024.

[Gao *et al.*, 2024] Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, 2024.

[Gerontopoulos *et al.*, 2025] Anastasios Gerontopoulos, Spyros Gidaris, and Nikos Komodakis. Multi-token prediction needs registers, 2025.

[Gloeckle *et al.*, 2024] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[Helsgaun, 2017] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980, 2017.

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

[Hottung *et al.*, 2025a] André Hottung, Mridul Mahajan, and Kevin Tierney. Polynet: Learning diverse solution strategies for neural combinatorial optimization. In *13th International Conference on Learning Representations*, 2025.

[Hottung *et al.*, 2025b] André Hottung, Paula Wong-Chung, and Kevin Tierney. Neural deconstruction search for vehicle routing problems. *Transactions on Machine Learning Research*, 2025.

[Hua *et al.*, 2025] Chuanbo Hua, Federico Berto, Jiwoo Son, Seunghyun Kang, Changhyun Kwon, and Jinkyoo Park. CAMP: Collaborative Attention Model with Profiles for Vehicle Routing Problems. In *Proceedings of the 2025 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2025.

[Huang *et al.*, 2025] Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. In *13th International Conference on Learning Representations*, 2025.

[Jiang *et al.*, 2023] Yuan Jiang, Zhiguang Cao, Yaoxin Wu, Wen Song, and Jie Zhang. Ensemble-based deep reinforcement learning for vehicle routing problems under distribution shift. In *Advances in Neural Information Processing Systems*, volume 36, pages 53112–53125, 2023.

[Jiang *et al.*, 2024] Xia Jiang, Yaoxin Wu, Yuan Wang, and Yingqian Zhang. Bridging large language models and optimization: A unified framework for text-attributed combinatorial optimization. *arXiv:2408.12214*, 2024.

[Jiang *et al.*, 2025] Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[Kim *et al.*, 2023] Minjun Kim, Junyoung Park, and Jinkyoo Park. Learning to CROSS exchange to solve min-max vehicle routing problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[Kim *et al.*, 2024] Hyeonah Kim, Minsu Kim, Sungsoo Ahn, and Jinkyoo Park. Symmetric replay training: Enhancing

sample efficiency in deep reinforcement learning for combinatorial optimization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[Kim *et al.*, 2025] Hyeonah Kim, Sanghyeok Choi, Jiwoo Son, Jinkyoo Park, and Changhyun Kwon. Neural genetic search in discrete spaces. In *Forty-second International Conference on Machine Learning*, 2025.

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

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

[Li *et al.*, 2024] Jingwen Li, Yining Ma, Zhiguang Cao, Yaoxin Wu, Wen Song, Jie Zhang, and Yeow Meng Chee. Learning feature embedding refiner for solving vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):15279–15291, 2024.

[Liao *et al.*, 2025] Zijun Liao, Jinbiao Chen, Debing Wang, Zizhen Zhang, and Jiahai Wang. Bopo: Neural combinatorial optimization via best-anchored and objective-guided preference optimization. In *Forty-second International Conference on Machine Learning*, 2025.

[Liu *et al.*, 2025] Suyu Liu, Zhiguang Cao, Shanshan Feng, and Yew-Soon Ong. A mixed-curvature based pre-training paradigm for multi-task vehicle routing solver. In *42nd International Conference on Machine Learning*, 2025.

[Luo *et al.*, 2023] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *The 37th Annual Conference on Neural Information Processing Systems*, 2023.

[Luo *et al.*, 2025] 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.

[Qi *et al.*, ] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

[Reinelt, 1991] Gerhard Reinelt. TSPLIB—a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[Sar and Ghadimi, 2023] Kubra Sar and Pezhman Ghadimi. A systematic literature review of the vehicle routing problem in reverse logistics operations. *Computers & Industrial Engineering*, 177:109011, 2023.

[Stern *et al.*, 2018] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

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

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

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

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

[Wang *et al.*, 2024] Chenguang Wang, Zhouliang Yu, Stephen McAleer, Tianshu Yu, and Yaodong Yang. Asp: Learn a universal neural solver! *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4102–4114, 2024.

[Wang *et al.*, 2025] Yang Wang, Ya-Hui Jia, Wei-Neng Chen, and Yi Mei. Distance-aware attention reshaping for enhancing generalization of neural solvers. *IEEE Transactions on Neural Networks and Learning Systems*, 36(10):18900–18914, 2025.

[Wu *et al.*, 2021] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5057–5069, 2021.

[Xiao *et al.*, 2025] Yubin Xiao, Yuesong Wu, Rui Cao, Di Wang, Zhiguang Cao, Peng Zhao, Yuanshu Li, You Zhou, and Yuan Jiang. DGL: Dynamic global-local information aggregation for scalable vrp generalization with self-improvement learning. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2025.

[Yao *et al.*, 2024] Shunyu Yao, Xi Lin, Jiashu Wang, Qingfu Zhang, and Zhenkun Wang. Rethinking supervised learning based neural combinatorial optimization for routing problem. *ACM Transactions on Evolutionary Learning and Optimization*, 2024.

[Yin *et al.*, 2026] Zhuoli Yin, Yi Ding, Reem Khir, and Hua Cai. ViTSP: A vision language models guided framework for solving large-scale traveling salesman problems. In *The Fourteenth International Conference on Learning Representations*, 2026.

[Zhou *et al.*, 2023] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *40th International Conference on Machine Learning*, 2023.