Published in Computers & Industrial Engineering 208 (2025) 111376
DOI: 10.1016/j.cie.2025.111376

Lightweight population-based policy optimization for pickup and delivery problems

Yizhou Liu [a,b], Li Li [c], Yixin Xu [d], Tang Liu [a], Rong Cheng [e], Die Wu [a], Jilin Yang [a], Jingwen Li [a,b,*]

[a] Department of Computer Science, Sichuan Normal University, Chengdu, 610101, China
[b] Visual Computing and Virtual Reality Key Laboratory of Sichuan Province, Sichuan Normal University, Chengdu, 610068, China
[c] School of Automation, Guangdong University of Technology, Guangzhou, 510006, China
[d] School of Computing and Information Systems, Singapore Management University, 188065, Singapore
[e] College of Transportation Engineering, Dalian Maritime University, Dalian, 116026, China

Keywords:
Deep reinforcement learning
Population-based search strategy
Pickup and delivery problems
Localized attention synthesis

A B S T R A C T

In recent years, applying deep models to automatically learn construction heuristics for vehicle routing problems has achieved remarkable advancements. However, they are less effective in searching solutions due to two primary limitations: relying on deterministic probability distributions and overlooking the strategic advantage of prioritizing nearby unvisited nodes during the route construction process, resulting in suboptimal policies In this paper, we propose a novel lightweight population-based policy optimization (LPPO) framework that learns a diverse population of solution strategies through the utilization of innovative perturbation factors, in order to facilitate search exploration. Moreover, we design a localized attention synthesis (LAS) network to dynamically refine the node selection process by prioritizing effective and informative decision-relevant features. To further ameliorate search efficiency, we leverage a cluster search scheme during inference that rapidly identifies the most effective search strategy from the population. We apply LPPO to address the pickup and delivery traveling salesman problem (PDTSP) and multi-commodity PDTSP (m-PDTSP). Empirical results show that our LPPO achieves lower gaps and better generalization in comparison with the state-of-the-art deep models specialized for PDP variants.

1. Introduction

Vehicle routing problems (VRPs) are of paramount significance in various industrial applications, such as logistics (Büyüközkan, lmaz, Özçelik, & lmaz, 2025), intelligent transportation (Liang, Gao, & Shen, 2023) and supply chain management (Dondo, Méndez, & Cerdá, 2011). Among VRP variants, the pickup and delivery problem (PDP) is particularly significant due to its modeling of paired requests, where each customer order involves a pickup location and a corresponding delivery location. This pairing structure is fundamental to numerous real-world services such as ride-sharing (Guo, Long, Xu, Yu, & Yuan, 2022) and meal delivery (Luo, Gu, Poon, Liu, & Lim, 2022). A notable special case of the PDP is the Pickup and Delivery Traveling Salesman Problem (PDTSP), which aims to determine the shortest route for a single vehicle that starts and ends at a depot while visiting all pickup and delivery nodes exactly once. The PDTSP is characterized by two primary constraints: (1) *visiting constraints*, requiring each node to be visited exactly once; and (2) *precedence constraints*, enforcing that each pickup node must be visited before its paired delivery node. Fig. 1 depicts an example of a PDTSP instance and solution with four requests, where blue spheres represent pickup points and orange squares denote their paired delivery points.

Due to the NP-hard nature (Arora, 2003), the PDP remains intractable for optimal resolution using exact solvers (Pessoa, Sadykov, Uchoa, & Vanderbeck, 2019). As desirable alternatives, heuristic methods (Olgun, Koç, & parmak, 2021) typically rely on hand-crafted rules designed to simplify the search process, and are commonly employed in industry to yield near-optimal solutions with much less computational costs. However, they are often tailored to specific problems and manually crafted through considerable trial and error, requiring redesigns whenever constraints or objectives shift. These limitations may hinder their applicability and accessibility in rapidly evolving industries. The emphasized precedence relationship inherent in PDP complicates the optimization of routes, thereby presenting additional challenges in comparison to canonical VRPs such as traveling salesman problem (TSP), which requires innovative solution strategies.

* Corresponding author.
E-mail addresses: 20231302008@stu.sicnu.edu.cn (Y. Liu), Leely_1002@hotmail.com (L. Li), yixinxu@smu.edu.sg (Y. Xu), liutang@sicnu.edu.cn (T. Liu), r.cheng@tue.nl (R. Cheng), wd@sicnu.edu.cn (D. Wu), jilinyang@sicnu.edu.cn (J. Yang), lijingwen@sicnu.edu.cn (J. Li).

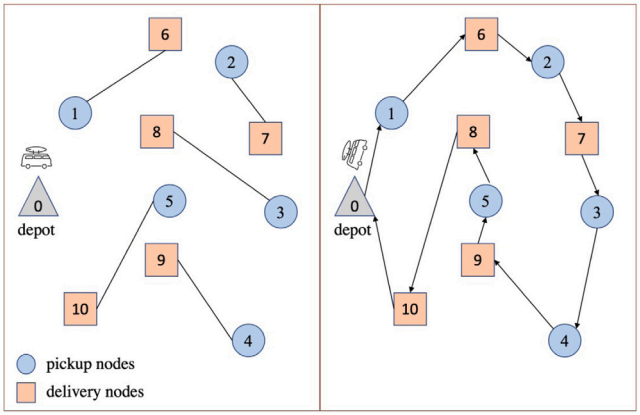

Fig. 1. Example of pickup and delivery Problem.

Recently, there has been increasing attention on applying deep reinforcement learning (DRL) (Qin, He, Zhang, & Li, 2021; Zhang, Li, Zhu, Shi, & Hwang, 2022) to explore neural heuristic methods for VRPs by automatically learning the rules in conventional heuristics rather than using the hand-crafted ones. The DRL models could be generally categorized to two classes, i.e., neural *construction* and *improvement* methods, respectively. The former sequentially decides next customer to visit in the solution, which excel at engendering high-quality solutions with rapid computation. In contrast, the latter iteratively improves a complete initial solution through neighborhood search (Chang, Shi, Luo, & Liu, 2023), while powerful, they usually require problem-specific operator selection and involve extensive local updates, leading to a large number of iterations and increased computational overhead. Thus, this study focuses on neural construction models owing to their efficiency and flexibility. Trained in the fashion of advanced reinforcement learning (Nazari, Oroojlooy, Snyder, & Takác, 2018a; Wu, Zuo, Chen, Ai, & Wan, 2024) or supervised learning (Min, Bai, & Gomes, 2023), the DRL models, particularly neural construction ones, demonstrate promising performance with significantly enhanced computational efficiency for solving simple VRPs such as TSP and Capacitated VRP (CVRP) (Wu, Song, Cao, Zhang, & Lim, 2021). Furthermore, by incorporating more complex and realistic constraints while adapting the model architecture to address them, those DRL models could be flexibly extended to tackle more challenging routing problems such as PDP variants (Ma et al., 2022).

Despite some notable successes, existing neural construction methods suffer from two major limitations for solving PDP. First, they often struggle to generate diverse solutions due to the constraints imposed by deterministic node embeddings and static probability distributions. Although multiple solutions can be sampled, most of them might be intrinsically identical in light of the unchanging distribution, which seriously impairs the search diversity. To mitigate this issue, some works adopt the policy optimization with multiple optima (POMO) (Kwon et al., 2020) scheme to promote exploration by enforcing diverse first actions during solution construction. In other words, each customer is designated to be served first so as to diversify the generated solutions. However, for PDP variants, the initial actions are restricted to customers requiring pickup due to the precedence constraint, rendering those methods less effective at generating diverse solutions. The second limitation lies in the design of decoders. The solution construction process could be deemed as a sequentially decision-making task that involves selecting the next (unvisited) customer after visiting the current customer. Represented on a graph, the task underscores the need for learning more informative node representations of customers. Moreover, good selection of next node to visit typically lies within the vicinity of the current node instead of across the entire pool of candidate nodes. To this end, efficiently extracting features from unvisited nearby nodes is essential for enhancing representation learning and, consequently, improving solution quality. However, most existing neural construction models solely rely on the current node embedding to reflect the context during solution construction, failing to emphasize critical decision-aware information (i.e., the vicinity) related to the current node, which may deteriorate efficiency and performance.

To address above two limitations, we propose the light-weight population-based policy optimization (LPPO), which not only learn a population of diverse solution strategies characterized by distinct probability distributions to facilitate search exploration, but also flexibly emphasize unvisited nearby nodes in relation to the current node during the decoding process to enhance search efficiency. Specifically, we first introduce the population-based search strategy that aims to integrate perturbation factors into a neural construction decoder via a lightweight network. By doing so, a population of multiple strategies are learned, which allows for the rapid generation of diverse solutions for a problem instance. Unlike POMO-based methods that impose diversity by enforcing varied initial nodes, the proposed diversity mechanism in LPPO enhances both solution performance and adaptability to new problems, especially for PDP variants studied in this paper. Then, we introduce a localized attention synthesis (LAS) network to ameliorate node selection precision by harnessing more relevant and informative contextual information related to the current node. It also incorporates a hypernetwork that leverages the embeddings of the current node and its nearby neighbors to intelligently adapt the attention weights of the decoder during solution construction. Additionally, we propose a cluster search scheme that organizes perturbation factors into distinct groups, enabling the rapid selection of the most effective factors to identify the promising solution strategies from the population for each problem instance. This scheme significantly enhances search efficiency during inference, thereby facilitating the swift identification of high-quality solutions. We evaluated our LPPO on two canonical PDP variants, i.e., the pickup and delivery traveling salesman problem (PDTSP) and its variant with multi-commodity (m-PDTSP) to verify our design. Empirical results show that our LPPO achieves lower gaps and better generalization than the state-of-the-art methods while maintaining comparable computational efficiency.

Our contributions are summarized as follows: (1) We propose a novel lightweight population-based policy optimization (LPPO) to promote the search diversity of neural construction solvers for PDP variants. By leveraging designed perturbation factors, LPPO learns a population of diverse search strategies through a single decoder, without significantly increasing computation time; (2) We design a LAS network that prioritizes effective decision-relevant information regarding the current node within the contextual representation, facilitating more precise node selection. It incorporates a hypernetwork to adaptively highlight salient features and refine the solution construction process; (3) We develop a cluster search scheme during inference that can efficiently identify the most effective perturbation factors, enabling the selection of promising search strategies from the population for a specific problem instance, which enhances the search efficiency and solution quality. Extensive experimental results on both PDTSP and m-PDTSP verify the superiority of our LPPO to the state-of-the-art neural models that are specialized for PDP variants.

2. Related work

In this section, we review recent advancements in solving combinatorial routing problems through *neural construction* and *neural improvement* paradigms based on deep reinforcement learning (DRL).

2.1. Neural construction methods

Starting from an empty solution, neural construction methods learn to sequentially select the next nodes to visit so as to construct a

complete solution by using deep neural networks to produce probability distributions over node arrangements. Vinyals, Fortunato, and Jaitly (2015) made the first attempt to propose a Pointer Network for solving TSP via supervised learning (Ammon, Phillipson, & Almeida, 2024). It was subsequently extended to reinforcement learning (Bello, Pham, Le, Norouzi, & Bengio, 2016; Du, Zhu, Wang, Han, & Qiao, 2024) and CVRP (Nazari, Oroojlooy, Takáč, & Snyder, 2018b). Other than RNNs used in Pointer Network, Khalil, Dai, Zhang, Dilkina, and Song (2017) introduced S2V-DQN based on a graph embedding network (Joshi, Laurent, & Bresson, 2019) for solving TSP, which outperformed handcrafted heuristic algorithms (Qiu, Sun, & Yang, 2022). With recent developments of the self-attention mechanism (Shaw, Uszkoreit, & Vaswani, 2018), the attention model (AM) (Kool, van Hoof, & Welling, 2019) introduced the Transformer-style model for solving multiple VRP variants, achieving strong performance and being recognized as a milestone in this field. In follow-up work, the AM is enhanced by the policy optimization with multiple optima (POMO) (Kwon et al., 2020), which enforces diverse rollouts and leveraged data augment techniques, and is recognized as the state-of-the-art neural construction method. Based on POMO, a large number of studies have emerged to address routing problems in recent years (Alqahtani, Scott, & Hu, 2022; Drakulic, Michel, Mai, Sors, & Andreoli, 2023; Sun & Yang, 2023). Among them, some works (Gao, Shang, Xue, Li, & Qian, 2024) partition unvisited nodes into smaller instances to mitigate the complexity of large-scale problems. Symmetric neural optimization for pick-up and delivery problems (PD-SNO) (Li, Niu, Zhu, & Xiao, 2024) effectively captures the symmetries of studied problems to enhance the solution performance, representing the state-of-the-art in solving PDP variants.

Post-processing search strategy. However, existing neural construction methods typically sample solutions based on fixed probability distributions, which often necessitate post-processing procedures to enhance the diversity of solution search, such as sampling (Li et al., 2023), active search (Li, Guo, Wang, & Yan, 2023a) and beam search (Choo et al., 2022). To bolster search diversity, Xin, Song, Cao, and Zhang (2021) proposed a multi-decoder AM (MDAM) to learn multiple decoding policies to improve solution quality. Similarly, Poppy (Grinsztajn, Furelos-Blanco, Surana, Bonnet, & Barrett, 2023) was introduced to employ a population of agents to learn complementary decoding strategies for more effective exploration of the solution space in VRP. While effective, Poppy requires a separate decoder for each agent of population and is computationally intensive, thus limiting the number of learnable policies per problem. Furthermore, Ma et al. (2022) proposed a data augmentation method that converts a problem instance into a set of its symmetrical counterparts for enhancing search diversity. However, the method is constrained by the symmetry of specific problems (e.g. TSP) and is less applicable to asymmetric variants, such as the PDP variants studied in this work.

2.2. Neural improvement methods

Neural improvement methods learn to search high-quality solutions by iteratively improving a complete initial solution until reaching a step limit. NeuRewriter proposed in Chen and Tian (2019) learned to select local search operations and components of the current solution to perform rewriting. Building on this concept, Lu, Zhang, and Yang (2019) proposed the L2I framework, which integrates learning with traditional heuristics by employing a deep neural network to select local search operators from a diverse candidate pool that contains multiple local search operators. A neural large neighborhood search (NLNS) method was proposed in Hottung and Tierney (2020), which learns to repair solutions through a neighborhood search paradigm. Wu et al. (2021) proposed a Transformer-style improvement heuristic that selects node pairs for local operators. It was extended to dual-aspect collaborative transformer (DACT) (Ma et al., 2021) that enhances the combination of node and positional embeddings. The DACT has achieved the state-of-the-art performance among neural improvement methods. Afterwards, Ma et al. (2021) further advanced the DACT by introducing N2S for solving PDP variants with the assistance of neural neighborhood search (Ma et al., 2022). By incorporating a specially designed data augmentation technique, N2S outperforms the LKH3 solver (Helsgaun, 2017) on synthetic datasets and becomes the state-of-the-art neural improvement solver for PDP variants, though it was subsequently surpassed by PD-SNO. While the neural improvement methods could consistently reduce solution costs, they typically rely on problem-specific operators and primarily focus on local updates, which inevitably requires a large number of iterations.

In summary, existing DRL-based solvers face challenges in balancing exploration diversity and computational efficiency. Table 1 provides a comparative analysis of related neural routing literature, highlighting the unique contributions of our study. By integrating the population-based search strategy based on simple perturbation factors, our LPPO efficiently sample diverse solutions from distinct probability distributions via a single decoder, enhancing search diversity and improving overall solution quality.

3. Preliminaries

We first introduce the studied PDP variants, i.e., PDTSP and m-PDTSP, and then present the prevalent encoder–decoder structure employed in deep learning models to construct VRP solutions in an autoregressive manner.

3.1. Mathematical formulation

We define a PDP instance over a graph $G = (V, E)$, where V signifies all nodes and E denotes edges between any two different nodes. The problem incorporates n customer requests, represented by a set of pickup nodes $P = \{x_1, x_2, \dots, x_n\}$ and a corresponding set of delivery nodes $D = \{x_{n+1}, x_{n+2}, \dots, x_{2n}\}$. Each pickup node x_i is paired with its corresponding delivery node x_{n+i}, with a precedence constraint requiring that the pickup node must be visited before the delivery node. With x_0 corresponding to the depot, the complete node set is defined as $V = \{x_0 \cup P \cup D\}$ with a total of $N = 2n + 1$ nodes. Let $V' = \{V \cup x_{2n+1}\}$, where the node x_{2n+1} refers to the copy of depot node. For PDTSP, each node $x_i \in V$ is associated with 2-dimensional location coordinates c_i. With above settings, PDTSP aims to determine the shortest Hamiltonian cycle that visits all pickup and delivery nodes and finally returns to the depot, while adhering to two key constraints: (1) each pickup and delivery node is visited exactly once; (2) each pickup node must precede its paired delivery node.

Let D_{ij} denote the Euclidean distance between nodes x_i and x_j, and let f represent the vehicle speed. Let $y_{ij} \in \{0, 1\}$ be a binary decision variable indicating whether the vehicle travels directly from node x_i to node x_j. Let B_i be the arrival time of node x_i and $\mathcal{M}$ be a sufficiently large constant. The PDTSP can be formulated as follows:

$$min \sum_{i\in V}\sum_{j\in V}\frac{D_{ij}}{f}y_{ij}. \tag{1}$$

subject to the following constraints,

$$\sum_{j\in V} y_{ij} = 1, \qquad i \in V, \tag{2}$$

$$\sum_{i\in V} y_{2n+1,i} = 0, \tag{3}$$

$$\sum_{j\in V} y_{ij} = \sum_{j\in V} y_{ji}, \qquad i \in V, \tag{4}$$

$$B_j \geq B_i + \frac{D_{ij}}{f} - \mathcal{M}(1 - y_{ij}), \qquad i, j \in V', \tag{5}$$

$$B_{i+n} \geq B_i + \frac{D_{i,i+n}}{f}, \qquad i \in P, \tag{6}$$

Table 1
Summary of related papers.

Reference	Problems	Solution method	Search strategy	Diversity mechanism	Model complexity
DACT (Ma et al., 2021)	TSP, CVRP	DRL Improvement	Single-policy sampling	Random sampling	Lightweight
N2S (Ma et al., 2022)	PDTSP	DRL Improvement	Single-policy sampling	Random sampling	Lightweight
AM (Kool et al., 2019)	TSP, CVRP	DRL Construction	Single-policy sampling	Random sampling	Lightweight
MDAM (Xin et al., 2021)	TSP, CVRP	DRL Construction	Multi-decoder sampling	Multi-decoder	Heavy
Heter-AM (Li et al., 2021)	PDTSP	DRL Construction	Single-policy sampling	Random sampling	Lightweight
POMO (Kwon et al., 2020)	TSP, CVRP	DRL Construction	Single-policy sampling	Multi-start	Lightweight
PD-SNO (Li et al., 2024)	PDTSP, m-PDTSP	DRL Construction	Single-policy sampling	Multi-start	Lightweight
Poppy (Grinsztajn et al., 2023)	TSP, CVRP	DRL Construction	Multi-decoder sampling	Multi-decoder	Heavy
This approach	PDTSP, m-PDTSP	DRL Construction	Population sampling	Adaptive perturbation	Lightweight

$$y_{ij} = \{0, 1\}, \qquad i, j \in V', \tag{7}$$

$$B_i \geq 0, \qquad i \in V'. \tag{8}$$

The objective function aims to minimize the total travel time of the vehicle. Constraint (2) and (3) ensure that each node (including both the depot and its copy) is visited exactly once. Constraint (4) enforces flow conservation, ensuring the continuity of the route through each node. Constraint (5) calculates the arrival times based on the tour sequence. Constraint (6) enforces the precedence relationship between each pickup node and its corresponding delivery node. Constraint (7) and Constraint (8) define the binary decision variable and enforce non-negativity of arrival times, respectively.

The m-PDTSP extends the PDTSP by introducing a limited capacity R_{max} for the vehicle. In this setting, each node $x_i \in V$ is featured by 2-dimensional location coordinates c_i and 1-dimensional demand d_i (the demand for depot is 0), where the demand at each delivery node is the negative of its pairing pickup node, i.e., $d_{i+n} = -d_i,\ d_i \geq 0,\ i \in P$. Let R_i denote the loading amount of the vehicle after visiting the node x_i, where the initial loading amount at the depot is $R_0 = 0$. Based on the above definition in PDTSP, m-PDTSP further incorporates the following constraints:

$$R_j \geq R_i + d_j - \mathcal{M}(1 - y_{ij}), \qquad i, j \in V, \tag{9}$$

$$0 \leq R_i \leq R_{max}, \qquad i \in V. \tag{10}$$

$$R_i \geq d_i \sum_{j \in V} y_{ij}, \qquad i \in P. \tag{11}$$

Specifically, Constraint (9) indicates that the difference of loading amount before and after each vehicle serving a node equals to its demand, whether it is a pickup node or a delivery node. Constraint (10) imposes that the loading amount of each vehicle is non-negative and cannot exceed its capacity. Constraint (11) guarantees sufficient loading amount to later fulfill its paired delivery node's demand $d_{i+n} = -d_i\ (i \in P)$.

3.2. Autoregressive deep models for VRPs

Neural construction methods typically utilize an encoder–decoder structured policy network π_θ with trainable parameters θ to construct solutions in an autoregressive manner. Given a VRP instance $\mathcal{X}$ of problem size N, the encoder projects the problem-specific features into high-dimensional node embeddings for informative representation learning. Afterwards, the decoder sequentially constructs a solution τ by iteratively selecting a node based on the learned node embeddings and partial tour at each decoding step. During the solution construction, all problem constraints are satisfied by masking the invalid nodes. A feasible solution is constructed until all customer nodes are selected, which is expressed by the factorization below,

$$\mathcal{P}(\tau \mid \mathcal{X}) = \prod_{t=1}^{n} \pi_\theta(\tau_t \mid \mathcal{X}, \tau_{1:t-1}). \tag{12}$$

where the decision making concerning the iterative node selection will be performed based on the learnt π_θ.

4. Methodology

In this section, we present the details of our encoder–decoder structured Lightweight Population-based Policy Optimization (LPPO). The concrete architecture of LPPO is illustrated in Fig. 2, where we take a PDTSP instance with 3 customer requests (i.e., 3 paired pickup and delivery nodes) as an example. Specifically, we adopt a Transformer-style encoder to extract problem-specific features and generate high-demensional node embeddings. In the decoder, our LPPO first leverages a Localized Attention Synthesis (LAS) network to prioritize effective decision-relevant information relative to the current node at each decoding step. Within the LAS network, a hypernetwork is incorporated to intelligently emphasize salient features and adjust the attention weights, thereby ameliorating the node selection process. Subsequently, we introduce the population-based search strategy that combines the refined attention weights from LAS network and the designed perturbation factors to produce a diverse population of search strategies through a lightweight network. This mechanism empowers the model to rapidly engender diverse solutions, which significantly improves both solution performance and the adaptability of the model to new problem variants.

4.1. Encoder

Similar to Heter-AM (Li et al., 2021), we adopt a Transformer-style encoder based on a heterogeneous attention mechanism. Compared to canonical VRPs, PDP variants introduce a one-to-one pairing relationship between pickup and delivery nodes, necessitating more efficient feature extraction. Given an instance $\mathcal{X}$ with a total of N nodes, we explicitly encode pairing relationships by concatenating each pickup node with its paired delivery node, i.e., $x_i = [x_i; x_{i+n}],\ i \in P$, where $\};\varepsilon$ denotes vector concatenation. The encoder then projects both the modified pickup node features and the remaining original features into the initial node embedding $\boldsymbol{h}_i^{\mathbf{0}} \in \mathbb{R}^{d_h},\ i \in V$, with $d_h = 128$. Subsequently, the encoder processes these embeddings through L heterogeneous attention layers, resulting in the final node representation $\boldsymbol{h}_i^{\boldsymbol{L}}$. Each attention layer is composed of a heterogeneous multi-head attention (Heter-MHA) sublayer and a feed-forward (FF) sublayer.

Heter-MHA sublayer. Similar to Heter-AM (Li et al., 2021), we employ a heterogeneous multi-head attention mechanism to capture complex relationships in PDPs. Our model consists of two main components: (1) a standard self-attention that models interactions among all nodes to extract comprehensive features; and (2) a pairing attention that explicitly focuses on the dependencies between each pickup node and its corresponding delivery node (and vice versa). Unlike Heter-AM that defines six distinct types of problem-specific attentions to model all possible interactions among pickup and delivery nodes, we adopt a simplified yet effective variant that retains only the essential attention types, i.e., standard and pairing attentions. This reduction offers better computational efficiency without significant performance degradation, as it concentrates on the most critical structural relationships inherent to PDPs.

For the standard self-attention, given the embeddings $\boldsymbol{h}_i^{\boldsymbol{l-1}}$ from layer $l-1$, we compute the *query, key* and *value* vectors for each attention head $y \in 1, \ldots, Y$ $(Y = 8)$, omitting y here for brevity:

$$Q_i = W_Q^l h_i^{l-1},\ K_i = W_K^l h_i^{l-1},\ V_i = W_V^l h_i^{l-1}, \quad i \in V, \tag{13}$$

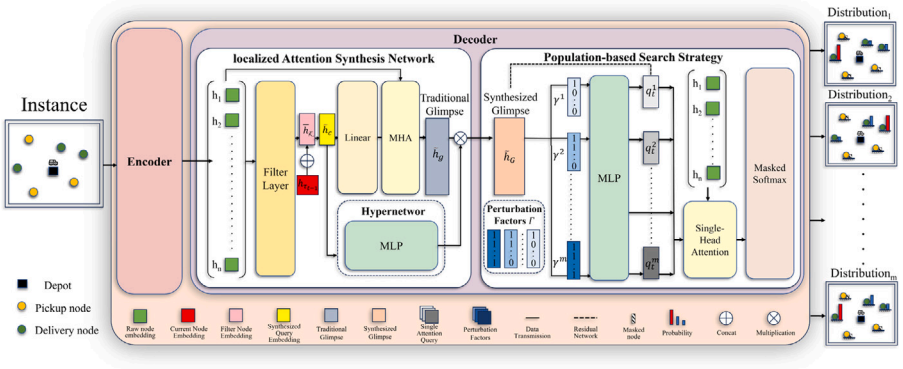

Fig. 2. Illustration of our policy network for a PDTSP instance with 6 customer nodes.

where $\boldsymbol{W}_{\boldsymbol{Q}}^{l}, \boldsymbol{W}_{\boldsymbol{K}}^{l} \in \mathbb{R}^{d_h \times d_k}$ and $\boldsymbol{W}_{\boldsymbol{V}}^{l} \in \mathbb{R}^{d_h \times d_v}$ are trainable matrices, with $d_k = d_v = d_h/Y$. The attention weights are calculated via scaled dot-product:

$$a_{ij} = \text{Softmax}\left(\frac{Q_i^\top K_j}{\sqrt{d_k}}\right), \quad i, j \in V, \tag{14}$$

indicating the relative importance of node x_j to node x_i.

For the pairing attention, we directly models the interactions between paired nodes. With $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{P}}$, $i \in P$ and $\boldsymbol{h}_{\boldsymbol{j}}^{\boldsymbol{D}}$, $j \in D$ referring to the embedding of the ith pickup node and jth delivery node in layer $l-1$, respectively, the pairing attention weights could be formulated as below (we omit layer $l-1$ for clarity),

$$\begin{aligned} Q_i^{pd} &= W_Q^{pd} h_i^P, \ K_{i+n}^{pd} = W_K h_{i+n}^D, \ V_{i+n}^{pd} = W_V h_{i+n}^D, \quad i \in P \\ a_{i,i+n}^{pd} &= \text{Softmax}\left(\frac{Q_i^{pd} * K_{i+n}^{pd}}{\sqrt{d_k}}\right), \quad i \in P, \end{aligned} \tag{15}$$

$$\begin{aligned} Q_j^{dp} &= W_Q^{dp} h_j^D, \ K_{j-n}^{dp} = W_K h_{j-n}^P, \ V_{j-n}^{dp} = W_V h_{j-n}^P, \quad j \in D \\ a_{j,j-n}^{dp} &= \text{Softmax}\left(\frac{Q_j^{dp} * K_{j-n}^{dp}}{\sqrt{d_k}}\right), \quad j \in D. \end{aligned} \tag{16}$$

where $\boldsymbol{W}_{\boldsymbol{Q}}^{\boldsymbol{pd}}, \boldsymbol{W}_{\boldsymbol{Q}}^{\boldsymbol{dp}} \in R^{d_h \times d_k}$ are trainable matrices for the pickup-to-delivery and delivery-to-pickup attention, respectively. Following Heter-AM, we employ element-wise multiplication "*" to compute pairing attention, which emphasizes dimension-wise interactions within paired nodes, better preserving these critical pairings and improving solution quality in PDPs. Moreover, to improve efficiency, all *key* and *value* projections are shared across attention types.

Each head's output $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{y}}$ is aggregated differently depending on the node type and attention type as below,

$$h_i^y = \begin{cases} a_{ij} V_j, & i = 0, \ j \in V, \ y \in \{1, \ldots, Y\}, \\ a_{ij} V_j + a_{i,i+n}^{pd} * V_{i+n}^{pd}, & i \in P, \ j \in V, \ y \in \{1, \ldots, Y\}, \\ a_{ij} V_j + a_{i,i-n}^{dp} * V_{i-n}^{dp}, & i \in D, \ j \in V, \ y \in \{1, \ldots, Y\} \end{cases} \tag{17}$$

The final output of the Heter-MHA layer is the concatenation of all heads, followed by a linear transformation as below,

$$\hat{h}_i = \text{Heter-MHA}(Q_i^y, K_j^y, V_j^y) = [h_i^1; h_i^2; \ldots; h_i^Y] W_O. \tag{18}$$

where $\boldsymbol{W}_{\boldsymbol{O}} \in \mathbb{R}^{d_h \times d_h}$ is a trainable projection matrix.

FF sublayer. The FF sublayer refines the Heter-MHA outputs $\hat{\boldsymbol{h}}_{\boldsymbol{i}}$ using two linear transformations with a ReLU activation in between as follows,

$$\text{FF}(\hat{h}_i) = W_F^1 \ \text{ReLU}(W_F^0 \hat{h}_i + b_F^0) + b_F^1. \tag{19}$$

where $\boldsymbol{W}_{\boldsymbol{F}}^{\boldsymbol{0}} \in \mathbb{R}^{d_h \times d_f}$ and $\boldsymbol{W}_{\boldsymbol{F}}^{\boldsymbol{1}} \in \mathbb{R}^{d_f \times d_h}$ are the weight matrices for the first and second linear transformations, respectively. $\boldsymbol{b}_{\boldsymbol{F}}^{\boldsymbol{0}} \in \mathbb{R}^{n \times d_f}$, and $\boldsymbol{b}_{\boldsymbol{F}}^{\boldsymbol{1}} \in \mathbb{R}^{n \times d_h}$ are the corresponding bias vectors. The hidden dimension is set to $d_f = 512$.

In line with Heter-AM, we apply residual connections (He, Zhang, Ren, & Sun, 2016) and batch normalization (BN) (Ioffe & Szegedy, 2015) to both the Heter-MHA and FF sublayers to mitigate gradient vanishing and improve convergence. The computation at layer l is as follows,

$$\begin{aligned} \hat{h}_i^l &= \text{BN}^l(h_i^{l-1} + \text{Heter-MHA}^l(Q_i^y, K_j^y, V_j^y)), \quad i, j \in V, \ y \in [1, \ldots, Y], \\ h_i^l &= \text{BN}^l(\hat{h}_i^l + \text{FF}^l(\hat{h}_i^l)), \quad i \in V, \ y \in [1, \ldots, Y], \end{aligned} \tag{20}$$

where $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{l}}$ refers to the node embeddings at the lth layer, and parameters are independently trained for different layers. Through L layers, the final node embeddings $\boldsymbol{h}_{\boldsymbol{i}}^{\boldsymbol{L}}$ are forwarded to the decoder for subsequent processing.

4.2. Decoder

In the decoder, we first leverage a localized attention synthesis (LAS) network for better reflecting current state representation. Then, we introduce a population-based search strategy to significantly enhance serach diversity through a lightweight network.

4.2.1. Localized attention synthesis network

During the decoding process, we introduce the Local Attention Subnetwork (LAS) to emphasize salient features of the current node and its immediate vicinity. This design ensures that decision-making is informed by the most relevant local information. As defined in Section 3.2, τ_{t-1} denotes the current node at each decoding step t. We define $\mathcal{U}_t$ as the set of unvisited nodes at step t, and $|\mathcal{U}_t|$ represents the number of unvisited nodes in set $\mathcal{U}_t$. To emphasize the most pertinent nodes, we identify the $\mathcal{K}$ nearest unvisited nodes to τ_{t-1}, denoted as $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{\mathcal{K}}$, based on the Euclidean distance. The filtered node embeddings $\tilde{\boldsymbol{h}}_{\boldsymbol{\mathcal{K}}}$ are then constructed as follows:

$$\tilde{h}_{\mathcal{K}} = \begin{cases} [h_{\mathcal{N}_1}; h_{\mathcal{N}_2}; \ldots; h_{\mathcal{N}_{\mathcal{K}}}], & \text{if } |\mathcal{U}_t| \geq \mathcal{K}, \\ [h_{\mathcal{N}_1}; \ldots; h_{\mathcal{N}_{|\mathcal{U}_t|}}; \underbrace{h_{\text{pad}}; \ldots; h_{\text{pad}}}_{\mathcal{K}-|\mathcal{U}_t| \text{ times}}], & \text{if } |\mathcal{U}_t| < \mathcal{K}, \end{cases} \tag{21}$$

where $\boldsymbol{h}_{\boldsymbol{\mathcal{N}}_{\boldsymbol{i}}}$ denotes the embedding of the ith nearest unvisited node $\mathcal{N}_i$ relative to τ_{t-1}, and $\boldsymbol{h}_{\text{pad}}$ is the padding embedding defined as the mean of the available unvisited node embeddings, i.e., $h_{\text{pad}} = \frac{1}{|\mathcal{U}_t|} \sum_{i=1}^{|\mathcal{U}_t|} h_{\mathcal{N}_i}$. This strategy ensures that LAS consistently receives a fixed-size input vector, facilitating parallel training and inference. By concentrating on the local neighborhood of the current node, LAS effectively captures decision-relevant information and further enhances the model's performance.

Subsequently, the filtered node embeddings $\tilde{\boldsymbol{h}}_{\boldsymbol{\mathcal{K}}}$ are processed through a multi-layer perceptron (MLP) (Riedmiller & Lernen, 2014) to obtain $\bar{h}_{\mathcal{K}} = \text{MLP}(\tilde{h}_{\mathcal{K}})$, which is subsequently concatenated with

the current node embedding $h_{\tau_{t-1}}$ to form the refined context vector $\tilde{h}_c$, encapsulating both the current state and the relevant neighborhood information:

$$\tilde{h}_c = [h_{\tau_{t-1}}; \overline{h}_{\mathcal{K}}]. \tag{22}$$

Following the Transformer-style decoder (Vaswani et al., 2017), we compute the glimpse $\tilde{h}_g$ by passing the refined context vector through a multi-head attention (MHA) sublayer:

$$\begin{aligned} Q^g = W_Q^g \tilde{h}_c,\ K_i^g = W_K^g h_i^L,\ V_i^g = W_V^g h_i^L, \quad i \in V \\ \tilde{h}_g = \mathrm{MHA}(Q^g,\ K_i^g,\ V_i^g). \end{aligned} \tag{23}$$

where h^L denotes the set of node embeddings from the encoder, and $W_Q^g \in \mathbb{R}^{d_h \times d_k}$, $W_K^g \in \mathbb{R}^{d_h \times d_k}$, and $W_V^g \in \mathbb{R}^{d_h \times d_v}$ are trainable parameter matrices. The MHA sublayer here refers to the standard multi-head attention mechanism, as defined in the encoder's Heter-MHA sublayer for the case of standard self-attention.

To enhance the flexibility and adaptability of our approach, we leverage the concept of hypernetworks (Chauhan, Zhou, Lu, Molaei, & Clifton, 2024), which have proven effective in dynamically adjusting the weights of a primary network to better address specific tasks. In our approach, the refined context vector $\tilde{h}_c$ is fed into an MLP-based hypernetwork, producing adaptive weights that modulate the original glimpse $\tilde{h}_g$, yielding the synthesized attention glimpse $\tilde{h}_G$:

$$\tilde{h}_G = W_Q^P \tilde{h}_g * \mathrm{MLP}(\tilde{h}_c), \tag{24}$$

where $W_Q^P \in \mathbb{R}^{d_h \times d_h}$ is a trainable matrix that applies a linear transformation to the node embeddings $\tilde{h}_g$ to enhance their representational capacity. This mechanism allows the model to dynamically adjust its attention focus based on the current context, thereby improving its decision-making capabilities.

4.2.2. Population-based search strategy

Combinatorial optimization problems like PDPs are inherently NP-hard, with solution spaces that expand exponentially as problem scales up. This complexity often leads to challenges like premature convergence in homogeneous search strategies. To address this, *diversity search* strategies have been developed to systematically explore distinct regions of the solution space, thereby reducing the risk of becoming trapped in suboptimal solutions (Qian, Xue, & Wang, 2024).

As illustrated in Fig. 3(a), existing neural construction models typically generate multiple solutions by sampling from a fixed probability distribution, which inherently limits search diversity. To overcome this limitation while ensuring computational efficiency, we propose a population-based search strategy that cultivates a population of multiple search strategies characterized by distinct probability distributions using a lightweight network, as depicted in Fig. 3(b). Specifically, we establish a varied set of perturbation factors $\Gamma = (\gamma_1, \ldots, \gamma_M)$, with each factor represented as a unique bit vector. Alternative representations can also be employed, provided they are sufficiently distinguishable for the network.

With the synthesized glimpse $\tilde{h}_G$ obtained from the LAS network, we condition the solution generation process on a perturbation factor $\gamma^m \in \Gamma$ by concatenating it with $\tilde{h}_G$. Each combined input is then processed through a MLP layer to generate a population of M diverse attention weight combinations. Given $k_i = W_K^F h_i^L,\ i \in V$ ($W_K^F \in \mathbb{R}^{d_h \times d_h}$ is the trainable matrix), the *compatibility* u_t^m with all nodes at step t based on the factor γ^m is computed as follows,

$$q_t^m = (\tilde{h}_G + \mathrm{MLP}([\tilde{h}_G; \gamma^m])), \quad m = 1, \ldots, M. \tag{25}$$

$$u_t^m = C \cdot \mathrm{TANH}\left(\frac{q_t^{mT} k_i}{\sqrt{d_k}}\right), \quad m = 1, \ldots, M. \tag{26}$$

where C is set to 10 to clip the result for better exploration. Finally, we derive a population of diverse search strategies, each represented by a probability distribution for selecting the next node to visit:

$$\mathcal{P}\left(\tau_t^m \mid \mathcal{X}, \tau_{1:t-1}^m, \gamma^m\right) = \mathrm{Softmax}(u_t^m), \quad m = 1, \ldots, M. \tag{27}$$

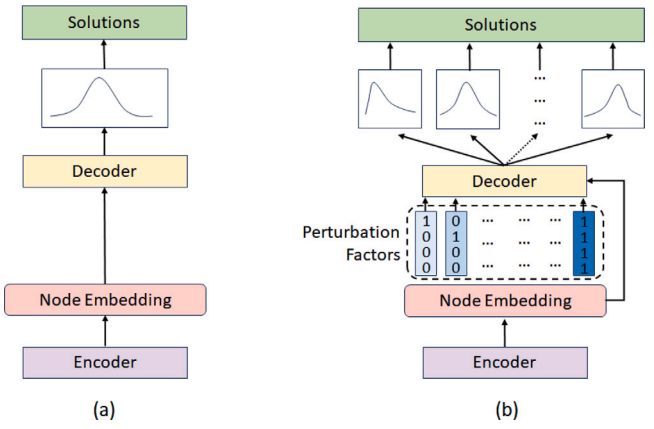

Fig. 3. Structures of neural heuristics. (a) Existing decoding structure with single search strategy; (b) Our decoding structure with population-based search strategy.

The probability of constructing a complete solution is factorized as:

$$\mathcal{P}(\tau^m \mid \mathcal{X}) = \prod_{t=1}^{N} \pi_\theta(\tau_t^m \mid \mathcal{X}, \tau_{1:t-1}, \gamma^m), \quad m = 1, \ldots, M. \tag{28}$$

This population-based search strategy enables dynamic adaptation of the probability distributions for node selection, facilitating search diversity through a single decoder. Unlike Poppy (Grinsztajn et al., 2023), which relies on multiple dedicated decoder networks, incurring substantial parameter growth and runtime overhead, LPPO maintains a single decoder and augments it with a *lightweight* MLP that integrates diverse perturbation vectors for diverse search strategies. This design delivers both efficiency and search diversity without multiplying model size or compute cost.

4.3. Training algorithm

As summarized in Algorithm 1, we adopt a policy gradient-based reinforcement learning algorithm (Akkerman, Mes, & van Jaarsveld, 2025) to train our LPPO network, denoted as π_θ with learnable parameters θ. A Monte Carlo method (Rubinstein & Kroese, 2016) is applied to update the parameters to iteratively improve the policy network. In each training iteration, we first randomly generate B instances on the fly (in line 3). Subsequently, we sample M unique perturbation factors (in line 5) to form M diverse search strategies for node selection. Using a single policy network, we construct M diverse solutions for each problem instance based on the perturbation factors (in line 6).

The rewards associated with these M solutions are calculated (in line 7), where $L(\tau_b^m, \mathcal{X}_b)$ denotes the tour length of solution τ_b^m for instance $\mathcal{X}_b$ for both PDTSP and m-PDTSP. Similar to POMO (Kwon et al., 2020), we employ a shared baseline by averaging the rewards from M sampled solutions (in line 9), thereby reducing variance in the policy gradients compared to the greedy-rollout methods (Grinsztajn et al., 2023). Notably, we update the model weights based solely on the best solution among M solutions. Let τ^* denote the best solution and γ^* denote the corresponding perturbation vector, we compute the updating gradient as follows:

$$\nabla_\theta J(\theta) \leftarrow \frac{1}{B} \sum_{b=1}^{B} (R_b^* - R_b^B) \nabla_\theta log \pi_\theta(\tau_b^* | \mathcal{X}_b, \gamma_b^*). \tag{29}$$

From the perturbation factor set Γ that contains M factors in total, we can sample varying number of factors, denoted as $\mathbb{M} \leq M$, to achieve a varying number of search strategies. The choice of $\mathbb{M}$ significantly influences both training efficiency and the overall model performance. A larger $\mathbb{M}$ increases the likelihood of finding the best perturbation factor γ^* from the set, while leading to a linear increase of training costs. In this study, we set $\mathbb{M}$=128 during training, promoting search diversity while mitigating excessive computational overhead.

ALGORITHM 1: LPPO Training

Input: Number of iterations E; batch size B; perturbation factors set Γ; number of perturbation factors M; policy network π_θ with parameters θ; learning rate η.
foreach *iteration* $= 1, 2, ..., E$ **do**
  Generate B problem instances randomly;
  foreach $b = 1, 2, ..., B$ **do**
    Get M perturbation factors $\{\gamma_b^m \in \Gamma\}_{m=1}^M$;
    Sample M solutions based on γ_b^m using Eq. (28), i.e., $\tau_b^m = \pi_\theta(\cdot|\mathcal{X}_b, \gamma_b^m),\ m = 1, ..., M$;
    Retrieve rewards for M solutions, i.e., $R_b^m = -L(\tau_b^m, \mathcal{X}_b), m = 1, ..., M$;
    Get best reward $R_b^* = \max(R_b^1, ..., R_b^M)$;
    Compute baseline reward $R_b^B = 1/M \sum_{i=1}^M R_b^m, m = 1, ..., M$;
  end
  Compute loss $\nabla_\theta J(\theta)$ using Eq. (29);
  $\theta = \theta + \eta \nabla_\theta J(\theta)$;
end

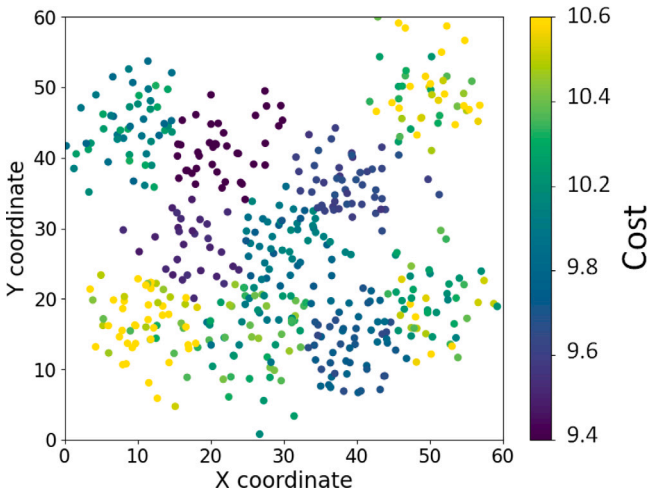

Fig. 4. 2D visualization of perturbation factors. The darker color indicates higher quality of solutions derived from the perturbation factors.

4.4. Cluster search scheme for inference

Fig. 4 illustrates a 2D visualization of these perturbation factors, where darker colors correspond to higher-quality solutions. This visualization reveals that perturbation factors naturally form distinct clusters, each associated with similar solution qualities.

Given multiple search strategies facilitated by the perturbation factors introduced in Section 4.2.2, we aim to efficiently identify the most effective strategy during inference to generate high-quality solutions. Fig. 4 illustrates a 2-dimensional visualization of perturbation factors, where darker colors correspond to higher-quality solutions derived from these factors. This visualization reveals that perturbation factors naturally form distinct clusters, each associated with similar solution qualities. Motivated by this insight, we propose a simple yet efficient clustering-based search scheme that rapidly identifies perturbation factors likely to yield promising solutions. Specifically, we partition the entire set of perturbation factors into clusters, select the most promising clusters associated with higher-quality solutions, and discard clusters linked to poorer performance. Final solutions are then generated by sampling perturbation factors within the selected clusters.

Given a set Γ of M perturbation factors, we apply the K-means clustering algorithm (Kodinariya, Makwana et al., 2013) to partition Γ into $Z = \sqrt{M}$ clusters, ensuring balanced cluster sizes. Each cluster thus contains approximately $F = M/Z$ perturbation factors. For each cluster, we identify its centroid perturbation factor as the cluster representative. We then generate one solution per cluster by sampling based on the representative perturbation factor. Next, we evaluate each cluster by assigning a score S_i based on the negative cost of the solution derived from its representative perturbation factor, for $i = 1, \ldots, Z$. During inference, a total of $\mathbb{N}$ solutions are sampled. We select the top $\mathbb{N}/F$ clusters with the highest scores, focusing our sampling efforts on the most promising regions of the perturbation factor space. Specifically, each selected cluster is responsible for generating F solutions by sampling from its constituent perturbation factors. Finally, among all $\mathbb{N}$ sampled solutions, we select the one with the best objective value as the final solution.

5. Experiments

In this section, we evaluate our LPPO by applying it to a neural construction model specialized for PDTSP, i.e., Heter-AM (Li et al., 2021), and conduct experiments on two representative PDP variants, i.e., PDTSP and m-PDTSP (Li et al., 2024). The LPPO models and baselines are tested on a server equipped with RTX 2080 TI GPU cards and Intel Xeon E5-2680 CPU @ 2.40 GHz.

5.1. Experimental settings

Following the established convention (Li et al., 2024), we randomly generate the locations of the depot and customer nodes (pickup and delivery pair) using a 2-dimensional uniform distribution within the [0,1] square (for both PDTSP and m-PDTSP), and the demand of the customers nodes using a 1-dimensional uniform distribution in the range of 1 to 10 (for m-PDTSP only). The problem sizes are set to 21, 51, and 101 (including one depot and all nodes of pickup and delivery). Regarding the capacity constraint in m-PDTSP, we set the vehicle capacity to 20 across all problem sizes. Regarding training, we adhere to most of the setups in POMO and summarize the hyperparameter settings in Table 2. For our approach, we train our models for 200, 400, and 1000 epochs for instances with size of 21, 51, and 101, respectively. In each epoch, 2000 batches with 50 instances are processed. Regarding testing, we generate 2000 instances randomly for each size and fix them for both our approach and all baseline methods, using the same distribution with the training one.

5.2. Perturbation factors analysis

In Table 3, we verify the effects of the numbers of sampled perturbation factors $\mathcal{M}$ on the model performance by training our LPPO using different $\mathcal{M}$ (from 32 to 256) for solving PDTSP and m-PDTSP instances with 101 nodes. The gaps on fixed test datasets are computed relative to the solutions yielding the lowest objective values, i.e., total length of the solutions, and the time refers to the training time per epoch across all methods. We can observe that an increased number of sampled perturbation factors generally correlates with lower objective values and reduced optimality gaps, but longer computation time. To strike an effective balance between solution quality and computational efficiency, we adopt $\mathcal{M} = 128$ for training in the following experiments. Regarding testing, $\mathcal{M}$ is set to 100 and 500, respectively.

5.3. Comparison analysis

We compare LPPO with following baseline methods: (1) highly specialized VRP solvers: Gurobi (L.L.C. Gurobi Optimization, 2021) and LKH3 (Helsgaun, 2017); (2) the state-of-the-art neural improvement method: N2S (Ma et al., 2022); (3) prevailing and efficient neural construction methods: POMO (Kwon et al., 2020), Heter-POMO (Li et al., 2021) and the state-of-the-art PDP solver PD-SNO (Li et al.,

Table 2
Parameter settings of our LPPO.

Parameter	Value	Description
embedding_dim d_h	128	Dimensionality of the feature embeddings.
sqrt_embedding_dim $\sqrt{d_h}$	$\sqrt{128}$	Square root of the embedding dimension, used for normalization
encoder_layer_num L	6	Number of layers in the encoder.
head_num Y	8	Number of attention heads in multi-head attention.
logit_clipping C	10	Clipping range for logit values to stabilize training.
ff_hidden_dim d_f	512	Hidden layer dimension in the feedforward network.
key_dim d_k	16	Dimension of single key vector in multi-head attention mechanism, $d_k = d_h/Y$.
value_dim d_v	16	Dimension of single value vector in multi-head attention mechanism, $d_v = d_h/Y$.
lr η	1e−4	Learning rate.
epochs E	1000	Number of times the training episodes are repeated.
train_episodes S	10 000	Number of training instances per epoch.
train_batch_size B	512	Number of instances used in each training batch.

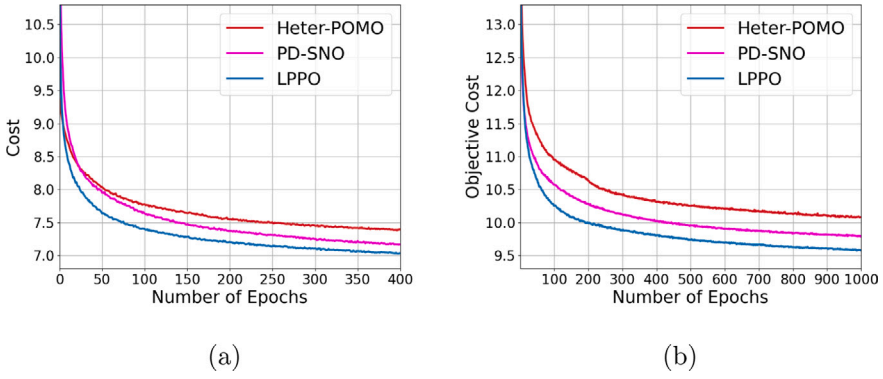

Fig. 5. Convergence curves of Heter-POMO, PD-SNO and LPPO during training. (a) PDTSP50. (b) PDTSP100.

Table 3
Gaps of LPPO with different numbers of perturbation factors.

Factor number	PDTSP-101			m-PDTSP-101		
	Obj.	Gap	Time	Obj.	Gap	Time
M = 32	9.462	0.48%	9 m	16.669	0.46%	10 m
M = 64	9.437	0.22%	11 m	16.624	0.19%	12 m
M = 128	9.419	0.03%	16 m	16.599	0.04%	19 m
M = 256	9.416	0.00%	32 m	16.592	0.00%	34 m

2024). Notably, Heter-POMO refers to the integration of the encoder from Heter-AM and the decoder from POMO to achieve better performance than either original method alone, as originally proposed in the N2S (Ma et al., 2022). We test strong baselines POMO and Heter-POMO on our test dataset based on pre-trained models, and retrain PD-SNO on both problems following its original setting due to the lack of pre-trained models. For Gurobi, we report results both with and without a 120-second time limit, and for LKH3, we present results based on two iteration settings, namely LKH(2000) and LKH(5000). For fair comparison, we follow the settings established by the state-of-the-art PDP solver PD-SNO, sampling 100 and 500 solutions, respectively, for inference. Following the guidelines provided in PD-SNO, all other baseline methods are evaluated under the same experimental settings. Furthermore, we apply "x8" augmentation strategy proposed by POMO during inference, which exploits symmetries inherent in VRP problems to enhance solution quality, across all neural methods.

We report the performance of our LPPO and all baselines in Tables 4 and Table 5 for PDTSP and m-PDTSP, respectively. We evaluate the objective values, optimality gaps, and computation times for all instance, utilizing the best results obtained by neural models to compute optimality gaps across all problem sizes. From Tables 4 and 5, we can observe that the exact solver Gurobi (without time limits) is only available for small-sized instances, e.g., PDTSP21 and m-PDTSP21, failing to produce solutions within reasonable time for larger sizes.

For neural construction methods, sampling strategy (s.) consistently achieves lower objective values than greedy strategy (g.), which demonstrates the effectiveness of sampling strategy in enhancing the solution quality, despite of longer computation time. Notably, our LPPO(s.500) achieves lowest objective values and optimality gaps across all problem sizes for both PDTSP and m-PDTSP with comparable computation time, particularly when compared to the state-of-the-art PDP solver PD-SNO. Specifically, even our sole LPPO(s.100) outperforms both POMO and Heter-POMO in all aspects. Furthermore, our LPPO(s.500) surpasses the highly specialized LKH3 across all sizes, and delivers comparable performance to Gurobi on small size (i.e., 20) for both PDTSP and m-PDTSP, which further demonstrates the effectiveness of our approach.

To further verify the efficiency of our LPPO against strong neural baseline methods, we illustrate their training curves for PDTSP instances of sizes 51 and 101 in Fig. 5, where POMO is omitted due to its clear inferiority to Heter-POMO. The horizontal axis refers to the training epochs and the vertical one refers to the objective values. We can observe that as the training progresses, our LPPO achieves both faster convergence and lower objective values than Heter-POMO and the state-of-the-art PD-SNO, reaffirming the superiority of our approach.

5.4. Generalization analysis

To verify the generalization of our LPPO, we continue to conduct two types of experiments: (1) cross-size: apply the policy learn for a problem size to a larger one; (2) cross-distribution: apply the policy learnt for uniform distribution to others. For both types of generalization, we compare our LPPO (s.500), SNO (s.500), and N2S (t.2000) given their superior performance indicated in Tables 4 and 5, all of which were trained on size 101 with uniform distribution.

(1) *Cross-size generalization*: we evaluate our LPPO and baseline methods on two benchmark datasets from Li et al. (2024), Renaud,

Table 4
Comparison with various baselines on PDTSP.

Method	PDTSP-21			PDTSP-51			PDTSP-101		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Gurobi	**4.570**	**0.00%**	3 m	–	–	–	–	–	–
Gurobi(120 s)	**4.570**	**0.00%**	3 m	6.895	0.42%	17 h	9.598	1.73%	66.7 h
LKH3(t.2000)	4.606	0.77%	15 m	6.916	0.73%	1.1 h	9.522	0.92%	6 h
LKH3(t.5000)	4.573	0.06%	41 m	6.891	0.36%	2.7 h	9.484	0.52%	13.8 h
N2S(t.100)	4.585	0.33%	5 m	7.156	4.22%	6 m	10.230	8.43%	13 m
N2S(t.500)	4.576	0.13%	23 m	6.964	1.43%	34 m	9.711	2.93%	1.1 h
N2S(t.2000)	4.571	0.02%	1.5 h	6.888	0.32%	2.2 h	9.475	0.42%	4.8 h
POMO(g.)	4.644	1.63%	2 s	7.192	4.75%	7s	10.077	6.80%	13 s
POMO(s.500)	4.594	0.53%	15 m	7.070	2.97%	56 m	9.851	4.40%	1.8 h
POMO(s.2000)	4.591	0.46%	26 m	7.042	2.56%	1.8 h	9.811	3.97%	5.9 h
Heter-POMO(g.)	4.600	0.66%	3 s	7.144	4.05%	12 s	10.053	6.55%	16 s
Heter-POMO(s.500)	4.585	0.33%	18 m	7.018	2.22%	1.1 h	9.697	2.78%	2 h
Heter-POMO(s.2000)	4.574	0.09%	34 m	6.939	1.06%	2.1 h	9.583	1.57%	6.2 h
PD-SNO(g.)	4.590	0.44%	3 s	6.924	0.85%	8 s	9.558	1.30%	18 s
PD-SNO(s.100)	4.584	0.31%	7 m	6.885	0.28%	20 m	9.479	0.47%	42 h
PD-SNO(s.500)	4.571	0.02%	31 m	6.876	0.15%	1.1 h	9.447	0.13%	3.6 h
LPPO(g.)	4.574	0.07%	1 s	6.882	0.23%	5 s	9.498	0.66%	12 s
LPPO(s.100)	4.572	0.03%	2 m	6.871	0.07%	6 m	9.451	0.17%	20 m
LPPO(s.500)	4.571	0.02%	28 m	**6.866**	**0.00%**	52 m	**9.435**	**0.00%**	2.8 h

Bold refers to the best performance among all methods.

Table 5
Comparison with various baselines on m-PDTSP.

Method	m-PDTSP-21			m-PDTSP-51			m-PDTSP-101		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Gurobi	**5.283**	**0.00%**	3 m	–	–	–	–	–	–
Gurobi(120 s)	**5.283**	**0.00%**	3 m	9.593	0.23%	18.7 h	16.792	1.16%	66.7 h
LKH3(t.2000)	5.314	0.64%	26 m	9.605	0.35%	1.7 h	16.713	0.69%	6.4 h
LKH3(t.5000)	5.284	0.07%	50 m	9.580	0.09%	3.3 h	16.627	0.17%	15 h
N2S(t.100)	5.297	0.31%	5 m	9.836	2.77%	8 m	17.684	6.53%	15 m
N2S(t.500)	5.288	0.14%	28 m	9.662	0.95%	42 m	16.962	2.19%	1.2 h
N2S(t.2000)	5.285	0.09%	1.6 h	9.592	0.22%	3.3 h	16.638	0.23%	5.2 h
POMO(g.)	5.348	1.28%	2 s	9.866	3.08%	9 s	17.524	5.57%	16 s
POMO(s.500)	5.307	0.50%	18 m	9.794	2.33%	1 h	17.285	4.13%	2 h
POMO(s.2000)	5.298	0.33%	30 m	9.775	2.13%	2.2 h	17.152	3.33%	6.1 h
Heter-POMO(g.)	5.336	1.06%	3 s	9.804	2.43%	13 s	17.224	3.76%	19 s
Heter-POMO(s.500)	5.301	0.39%	24 m	9.762	1.99%	51 m	16.944	2.08%	2.4 h
Heter-POMO(s.2000)	5.286	0.11%	1.4 h	9.645	0.77%	3.3 h	16.766	1.02%	6.4 h
PD-SNO(g.)	5.041	0.41%	4 s	9.505	2.49%	10 s	16.155	1.77%	21 s
PD-SNO(s.100)	5.293	0.24%	8 m	9.591	0.21%	24 m	16.733	0.81%	48 m
PD-SNO(s.500)	5.285	0.08%	42 m	9.583	0.13%	1.2 h	16.621	0.13%	3.9 h
LPPO(g.)	5.292	0.22%	2 s	9.697	1.32%	7 s	16.931	1.99%	15 s
LPPO(s.100)	5.284	0.07%	3 m	9.622	0.52%	7 m	16.725	0.76%	27 m
LPPO(s.500)	5.283	0.05%	36 m	**9.571**	**0.00%**	1.1 h	**16.599**	**0.00%**	3 h

Boctor, and Laporte (2002) for PDTSP and m-PDTSP instances of sizes 101 and 201, respectively. The results are summarized in Tables 6 and 7. For m-PDTSP instances, we adopt LKH3 (t.10000) as the heuristic baseline since PD-SNO does not provide responsive optimal costs. Similar to PD-SNO, we define "B.Obj" as the best objective value and "A.Obj" as the average objective value across all solutions for all neural methods. Both Tables 6 and 7 reveals that our LPPO consistently achieves the lowest average best cost (B.Obj) across multiple scales for both PDTSP and m-PDTSP, outperforming the prevailing N2S (t.2000) and the state-of-the-art PD-SNO (s.500).

Furthermore, regarding the average cost across all solutions (A.obj), our LPPO continues to exceed the performance of other neural baseline methods, underscoring the effectiveness of our method. Notably, for m-PDTSP instance of size 201, LPPO significantly surpasses PD-SNO and N2S, achieving lower objective values than the well-established heuristic solver LKH3 (t.10000), which highlights stable superiority of our method.

(2) *Cross-distribution generalization:* Following previous studies (Li et al., 2021; Zhou, Wu, Song, Cao, & Zhang, 2023), we establish a comprehensive benchmark comprising 1,000 instances for each of PDTSP and m-PDTSP problems under three distinct distributions, i.e., Cluster, Mixed and Gaussian (different from Uniform distribution during training), to verify the cross-distribution generalization of learning-based methods. We record the results in Table 8, where "Avg." indicates the average objective value among all distributions. As evidenced in Table 8, our LPPO outperforms both the state-of-the-art PD-SNO and prevailing N2S in terms of both best objective value (B.obj) and average objective value among all solutions (A.obj) across all distributions for both PDTSP and m-PDTSP. This further validates the robustness and effectiveness of our method on generalizing to different distributions.

Table 6
Cross-size generalization results on benchmark dataset of PDTSP.

Ins.	Size	Opt.	LPPO (s.500)		PD-SNO (s.500)		N2S (t.2000)	
			B.Obj	A.Obj	B.Obj	A.Obj	B.Obj	A.Obj
P01	101	799	803	805	**799**	801	800	802
P02	101	729	**718**	719	729	733	731	732
P03	101	748	750	751	**748**	749	749	751
P04	101	807	808	809	**807**	808	813	819
P05	101	783	**783**	785	783	786	785	785
P06	101	755	**755**	764	755	757	756	758
P07	101	767	769	770	**767**	769	772	774
P08	101	762	**762**	763	762	765	764	764
P09	101	766	**765**	767	766	768	767	767
P10	101	754	**754**	757	754	758	757	766
Avg.		767	**767**	769	767	769	769	772
P11	201	1038	1062	1070	**1059**	1071	1062	1068
P12	201	1086	1100	1110	**1096**	1106	1106	1127
P13	201	1070	**1082**	1092	1092	1101	1102	1112
P14	201	1050	**1061**	1068	1066	1073	1072	1076
P15	201	1052	**1067**	1086	1078	1092	1088	1094
P16	201	1059	**1069**	1077	1073	1084	1082	1086
P17	201	1036	**1052**	1061	1068	1079	1074	1076
P18	201	1079	**1094**	1106	1097	1110	1104	1112
P19	201	1050	1086	1093	**1066**	1076	1077	1086
P20	201	1085	**1104**	1115	1108	1122	1118	1126
Avg.		1065	**1078**	1088	1080	1091	1088	1096

Bold refers to the best performance among all neural methods.

Table 7
Cross-size generalization result on benchmark dataset of m-PDTSP.

Ins.	Size	LKH3[a]	LPPO (s.500)		PD-SNO (s.500)		N2S (t.2000)	
		(t.10000)	B.Obj	A.Obj	B.Obj	A.Obj	B.Obj	A.Obj
P01	101	1614	**1613**	1637	1649	1653	1669	1682
P02	101	1746	1751	1774	**1741**	1743	1764	1784
P03	101	1652	**1648**	1671	1710	1721	1736	1771
P04	101	1633	**1633**	1681	1739	1749	1764	1771
P05	101	1559	**1561**	1608	1630	1632	1656	1668
P06	101	1728	**1726**	1754	1754	1756	1770	1786
P07	101	1507	**1509**	1520	1558	1563	1574	1584
P08	101	1688	**1680**	1721	1756	1759	1783	1797
P09	101	1794	**1794**	1847	1840	1847	1863	1876
P10	101	1608	**1605**	1631	1643	1646	1658	1665
Avg.		1654	**1653**	1684	1702	1707	1724	1738
P11	201	2859	**2859**	2910	2942	2966	2977	3019
P12	201	2933	**2895**	2963	2998	3031	3052	3102
P13	201	3067	**2955**	3022	3055	3079	3114	3138
P14	201	2816	**2816**	2890	2963	2989	3020	3054
P15	201	3062	**3046**	3100	3177	3198	3240	3264
P16	201	3179	**3162**	3225	3246	3274	3302	3326
P17	201	3011	**2993**	3099	3241	3264	3300	3334
P18	201	3395	**3363**	3431	3438	3464	3480	3529
P19	201	3277	**3275**	3319	3366	3398	3431	3466
P20	201	3169	**3161**	3247	3297	3328	3352	3382
Avg.		3077	**3052**	3120	3172	3199	3227	3261

[a] PD-SNO does not provide optimal costs for this dataset, thus we adopt LKH3 (t.10000) as the heuristic baseline.

5.5. Further analysis of our LPPO

We further provide more analysis to study the effects of the population-based search strategy and the individual components of our LPPO framework.

(1) *Effect of the population-based search strategy*: To verify that the proposed population-based search strategy could effectively deliver diverse search via perturbation factors, we visualize the probability distributions for selecting the next node for our LPPO (using $\mathcal{M} = 4$) and the state-of-the-art PD-SNO during solving a PDTSP-51 instance in Fig. 6. Given a fixed partial solution, we plot the probability distributions for selecting the next node at the 20th decoding step for LPPO and PD-SNO. We find that PD-SNO relies on a single probability distribution, where the sole diversity in solutions comes from sampling with the same distribution. In contrast, our LPPO could explore more diverse solutions by diversifying the probability distributions via different perturbation factors, which helps improve both sampling efficiency and solution quality (final objective values are 6.863 and 6.912 for LPPO and PD-SNO).

Table 8
Cross-distribution generalization results on generated dataset.

Method	Cluster	Mix		Gaussian		Avg.			
		B.Obj	A.Obj	B.Obj	A.Obj	B.Obj	A.Obj	B.Obj	A.Obj
PDTSP	LPPO	**467**	**472**	**872**	**877**	**986**	**990**	**775**	**779**
	PD-SNO	468	473	874	880	986	991	776	781
	N2S	473	477	878	885	989	995	780	786
m-PDTSP	LPPO	**527**	**539**	**967**	**998**	**1091**	**1130**	**861**	**889**
	PD-SNO	541	562	982	1023	1133	1173	885	919
	N2S	545	564	995	1031	1142	1186	894	927

Table 9
Effects of each component of LPPO.

LAS	Population	Cluster	PDTSP-101		m-PDTSP-101	
×	×	×	10.060	5.03%	17.785	3.01%
✓	×	×	9.966	4.05%	17.566	1.74%
×	✓	×	9.820	2.52%	17.457	1.11%
×	✓	✓	9.780	2.10%	17.421	0.90%
✓	✓	×	9.623	0.46%	17.310	0.26%
✓	✓	✓	**9.578**	**0.00%**	**17.265**	**0.00%**

(2) *Effect of Each Component of the LPPO*: In Table 9, we conduct an ablation study to showcase the effect of each component within our LPPO, i.e., LAS network, population-based search strategy and cluster search scheme (only for inference), on PDTSP101 and m-PDTSP101. The optimality gaps are calculated relative to the solutions obtained by the method with lowest objective values. The markers "✓" and "×" denote the inclusion or exclusion of the respective components. From Table 9, we find that each component contributes positively to performance enhancement. Further combining three components together, our LPPO (the final row) achieves the best performance in terms of the objective values and optimality gaps.

6. Conclusion

In this paper, we propose a novel LPPO framework aimed at enhancing the search diversity of neural construction methods for solving PDTSP and m-PDTSP. By employing unique perturbation factors, our LPPO empowers deep models to learn a diverse population of search strategies characterized by distinct probability distributions through a single decoder. The integration of a novel LAS network allows our approach to prioritize effective decision-relevant features, which contributes to ameliorate the node selection process. Additionally, LPPO incorporates a cluster search scheme during inference that rapidly identifies the most effective search strategies from the population to boost search efficiency and solution quality. Extensive experiments show that our LPPO consistently outperforms the state-of-the-art neural construction models for PDP with enhanced sampling efficiency. In future, we will investigate how to extend LPPO to other combinatorial optimization problems like other routing and scheduling challenges, which may present unique challenges such as handling real-time demand fluctuations, integrating heterogeneous constraints and balancing efficiency with adaptability in dynamic environments.

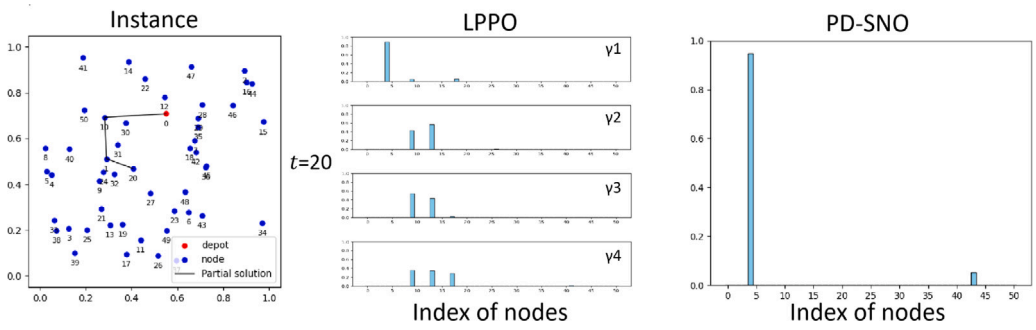

Fig. 6. Probability distributions of LPPO and PD-SNO for taking actions at last visited node 20 given a fixed partial solution {0, 10, 1, 20}. The distributions at 20th decoding step are presented.

CRediT authorship contribution statement

Yizhou Liu: Methodology, Writing – original draft. **Li Li:** Formal analysis, Writing – original draft. **Yixin Xu:** Supervision, Writing – review & editing. **Tang Liu:** Funding acquisition, Writing – review & editing. **Rong Cheng:** Project administration, Supervision. **Die Wu:** Data curation, Supervision. **Jilin Yang:** Funding acquisition, Visualization. **Jingwen Li:** Writing – review & editing, Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62306201, 62072320), the Natural Science Foundation of Sichuan Province, China (2025ZNSFSC1466, 2022NSFSC0569, 2025ZNSFSC0501), and the Humanities and Social Science Fund of the Ministry of Education of China (23YJA630114).

Data availability

Data will be made available on request.

References

Akkerman, F., Mes, M., & van Jaarsveld, W. (2025). A comparison of reinforcement learning policies for dynamic vehicle routing problems with stochastic customer requests. *Computers & Industrial Engineering*, *200*, Article 110747. http://dx.doi.org/10.1016/j.cie.2024.110747, URL: https://www.sciencedirect.com/science/article/pii/S0360835224008696.

Alqahtani, M., Scott, M. J., & Hu, M. (2022). Dynamic energy scheduling and routing of a large fleet of electric vehicles using multi-agent reinforcement learning. *Computers & Industrial Engineering*, *169*, Article 108180. http://dx.doi.org/10.1016/j.cie.2022.108180, URL: https://www.sciencedirect.com/science/article/pii/S0360835222002509.

Ammon, S., Phillipson, F., & Almeida, R. J. (2024). A supervised machine learning approach for the vehicle routing problem. In *ICORES* (pp. 364–371).

Arora, S. (2003). Approximation schemes for NP-hard geometric optimization problems: A survey. *Mathematical Programming*, *97*(1), 43–69.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Büyüközkan, K., lmaz, B. G. Y., Özçelik, G., & lmaz, Ö. F. Y. (2025). An optimization model and customized solution approaches for in-plant logistic problem within the context of lean management. *Computers & Industrial Engineering*, *200*, Article 110832. http://dx.doi.org/10.1016/j.cie.2024.110832, URL: https://www.sciencedirect.com/science/article/pii/S0360835224009549.

Chang, X., Shi, J., Luo, Z., & Liu, Y. (2023). Adaptive large neighborhood search algorithm for multi-stage weapon target assignment problem. *Computers & Industrial Engineering*, *181*, Article 109303. http://dx.doi.org/10.1016/j.cie.2023.109303, URL: https://www.sciencedirect.com/science/article/pii/S0360835223003273.

Chauhan, V. K., Zhou, J., Lu, P., Molaei, S., & Clifton, D. A. (2024). A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, *57*(9), 1–29.

Chen, X., & Tian, Y. (2019). Learning to perform local rewriting for combinatorial optimization. In *Advances in neural information processing systems*: *vol. 32*.

Choo, J., Kwon, Y. D., Kim, J., Jae, J., Hottung, A., Tierney, K., & Gwon, Y. (2022). Simulation-guided beam search for neural combinatorial optimization. In *Advances in neural information processing systems*: *vol. 35*, (pp. 8760–8772).

Dondo, R., Méndez, C. A., & Cerdá, J. (2011). The multi-echelon vehicle routing problem with cross docking in supply chain management. *Computers & Chemical Engineering*, *35*(12), 3002–3024.

Drakulic, D., Michel, S., Mai, F., Sors, A., & Andreoli, J.-M. (2023). Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. In *Advances in neural information processing systems*: *vol. 36*, (pp. 77416–77429).

Du, S., Zhu, Z., Wang, X., Han, H., & Qiao, J. (2024). Real-time local path planning strategy based on deep distributional reinforcement learning. *Neurocomputing*, *599*, Article 128085.

Gao, C., Shang, H., Xue, K., Li, D., & Qian, C. (2024). Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. *International Joint Conference on Artificial Intelligence*.

Grinsztajn, N., Furelos-Blanco, D., Surana, S., Bonnet, C., & Barrett, T. (2023). Winner takes it all: Training performant RL populations for combinatorial optimization. In *Advances in neural information processing systems*: *vol. 36*, (pp. 48485–48509).

Guo, J., Long, J., Xu, X., Yu, M., & Yuan, K. (2022). The vehicle routing problem of intercity ride-sharing between two cities. *Transportation Research Part B: Methodological*, *158*, 113–139.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Helsgaun, K. (2017). An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, *12*, 966–980.

Hottung, A., & Tierney, K. (2020). Neural large neighborhood search for the capacitated vehicle routing problem. In *ECAI 2020* (pp. 443–450). IOS Press.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).

Joshi, C. K., Laurent, T., & Bresson, X. (2019). An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in neural information processing systems*: *vol. 30*.

Kodinariya, T. M., Makwana, P. R., et al. (2013). Review on determining number of cluster in K-means clustering. *International Journal*, *1*(6), 90–95.

Kool, W., van Hoof, H., & Welling, M. (2019). Attention, learn to solve routing problems!. In *International conference on learning representations*.

Kwon, Y. D., Choo, J., Kim, B., Yoon, I., Gwon, Y., & Min, S. (2020). Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in neural information processing systems*: *vol. 33*, (pp. 21188–21198).

Li, Y., Guo, J., Wang, R., & Yan, J. (2023). T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. In *Advances in neural information processing systems*: *vol. 36*, (pp. 50020–50040).

Li, J., Ma, Y., Cao, Z., Wu, Y., Song, W., Zhang, J., & Chee, Y. M. (2023). Learning feature embedding refiner for solving vehicle routing problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Li, J., Niu, Y., Zhu, G., & Xiao, J. (2024). Solving pick-up and delivery problems via deep reinforcement learning based symmetric neural optimization. *Expert Systems with Applications*, *255*, Article 124514.

Li, J., Xin, L., Cao, Z., Lim, A., Song, W., & Zhang, J. (2021). Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, *23*(3), 2306–2315.

Liang, Y., Gao, Y., & Shen, Z. (2023). Transformer vehicle re-identification of intelligent transportation system under carbon neutral target. *Computers & Industrial Engineering*, *185*, Article 109619. http://dx.doi.org/10.1016/j.cie.2023.109619, URL: https://www.sciencedirect.com/science/article/pii/S0360835223006435.

L. L. C. Gurobi Optimization (2021). Gurobi optimizer reference manual. URL: http://www.gurobi.com.

Lu, H., Zhang, X., & Yang, S. (2019). A learning-based iterative method for solving vehicle routing problems. In *International conference on learning representations*.

Luo, Z., Gu, R., Poon, M., Liu, Z., & Lim, A. (2022). A last-mile drone-assisted one-to-one pickup and delivery problem with multi-visit drone trips. *Computers & Operations Research*, *148*, Article 106015. http://dx.doi.org/10.1016/j.cor.2022.106015, URL: https://www.sciencedirect.com/science/article/pii/S0305054822002453.

Ma, Y., Li, J., Cao, Z., Song, W., Guo, H., Gong, Y., & Chee, M. C. (2022). Efficient neural neighborhood search for pickup and delivery problems. In *Proceedings of international joint conference on artificial intelligence vienna, Austria* (pp. 23–29).

Ma, Y., Li, J., Cao, Z., Song, W., Zhang, L., Chen, Z., & Tang, J. (2021). Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in neural information processing systems*: *vol. 34*, (pp. 11096–11107).

Min, Y., Bai, Y., & Gomes, C. P. (2023). Unsupervised learning for solving the travelling salesman problem. In *Advances in neural information processing systems*: *vol. 36*, (pp. 47264–47278).

Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. In *Advances in neural information processing systems*: *vol. 31*.

Nazari, M., Oroojlooy, A., Takáč, M., & Snyder, L. V. (2018). Reinforcement learning for solving the vehicle routing problem. In *Advances in neural information processing systems*, (pp. 9861–9871).

Olgun, B., Koç, Ç., & parmak, F. A. (2021). A hyper heuristic for the green vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, *153*, Article 107010. http://dx.doi.org/10.1016/j.cie.2020.107010, URL: https://www.sciencedirect.com/science/article/pii/S036083522030680X.

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2019). A generic exact solver for vehicle routing and related problems. In *Integer programming and combinatorial optimization: 20th international conference* (pp. 354–369). Springer.

Qian, C., Xue, K., & Wang, R. J. (2024). Quality-diversity algorithms can provably be helpful for optimization. arXiv preprint arXiv:2401.10539.

Qin, B., He, W., Zhang, B., & Li, J. (2021). A multi-agent reinforcement learning framework with recurrent communication module for traffic light control. In *2021 IEEE 4th international conference on information systems and computer aided education* (pp. 117–121). http://dx.doi.org/10.1109/ICISCAE52414.2021.9590701.

Qiu, R., Sun, Z., & Yang, Y. (2022). Dimes: A differentiable meta solver for combinatorial optimization problems. In *Advances in neural information processing systems*: *vol. 35*, (pp. 25531–25546).

Renaud, J., Boctor, F. F., & Laporte, G. (2002). Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, *29*(9), 1129–1141.

Riedmiller, M., & Lernen, A. (2014). Multi layer perceptron. *Machine Learning Lab Special Lecture, University of Freiburg*, *24*.

Rubinstein, R. Y., & Kroese, D. P. (2016). *Simulation and the Monte Carlo method*. John Wiley & Sons.

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. arXiv preprint arXiv:1803.02155.

Sun, Z., & Yang, Y. (2023). Difusco: Graph-based diffusion solvers for combinatorial optimization. In *Advances in neural information processing systems*: *vol. 36*, (pp. 3706–3731).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems*: *vol. 28*.

Wu, Y., Song, W., Cao, Z., Zhang, J., & Lim, A. (2021). Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(9), 5057–5069.

Wu, B., Zuo, X., Chen, G., Ai, G., & Wan, X. (2024). Multi-agent deep reinforcement learning based real-time planning approach for responsive customized bus routes. *Computers & Industrial Engineering*, *188*, Article 109840. http://dx.doi.org/10.1016/j.cie.2023.109840, URL: https://www.sciencedirect.com/science/article/pii/S0360835223008641.

Xin, L., Song, W., Cao, Z., & Zhang, J. (2021). Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 35, no. 13*, (pp. 12042–12049).

Zhang, L., Li, J., Zhu, Y., Shi, H., & Hwang, K.-S. (2022). Multi-agent reinforcement learning by the actor-critic model with an attention interface. *Neurocomputing*, *471*, 275–284. http://dx.doi.org/10.1016/j.neucom.2021.06.049, URL: https://www.sciencedirect.com/science/article/pii/S0925231221009735.

Zhou, J., Wu, Y., Song, W., Cao, Z., & Zhang, J. (2023). Towards omni-generalizable neural methods for vehicle routing problems. In *International conference on machine learning* (pp. 42769–42789). PMLR.