DPN: Decoupling Partition and Navigation for Neural Solvers of Min-max Vehicle Routing Problems

Zhi Zheng [*1] Shunyu Yao [*2 3] Zhenkun Wang [3] Xialiang Tong [4] Mingxuan Yuan [4] Ke Tang [1]

Abstract

The min-max vehicle routing problem (min-max VRP) traverses all given customers by assigning several routes and aims to minimize the length of the longest route. Recently, reinforcement learning (RL)-based sequential planning methods have exhibited advantages in solving efficiency and optimality. However, these methods fail to exploit the problem-specific properties in learning representations, resulting in less effective features for decoding optimal routes. This paper considers the sequential planning process of min-max VRPs as two coupled optimization tasks: customer partition for different routes and customer navigation in each route (i.e., partition and navigation). To effectively process min-max VRP instances, we present a novel attention-based Partition-and-Navigation encoder (P&N Encoder) that learns distinct embeddings for partition and navigation. Furthermore, we utilize an inherent symmetry in decoding routes and develop an effective agent-permutation-symmetric (APS) loss function. Experimental results demonstrate that the proposed Decoupling-Partition-Navigation (DPN) method significantly surpasses existing learning-based methods in both single-depot and multi-depot min-max VRPs. Our code is available at [1].

[*]Equal contribution [1]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China [2]Department of Computer Science, City University of Hong Kong, Hong Kong SAR. [3]School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, China [4]China Huawei Noah's Ark Lab, Shenzhen, China.. Correspondence to: Zhenkun Wang <wangzhenkun90@gmail.com>.

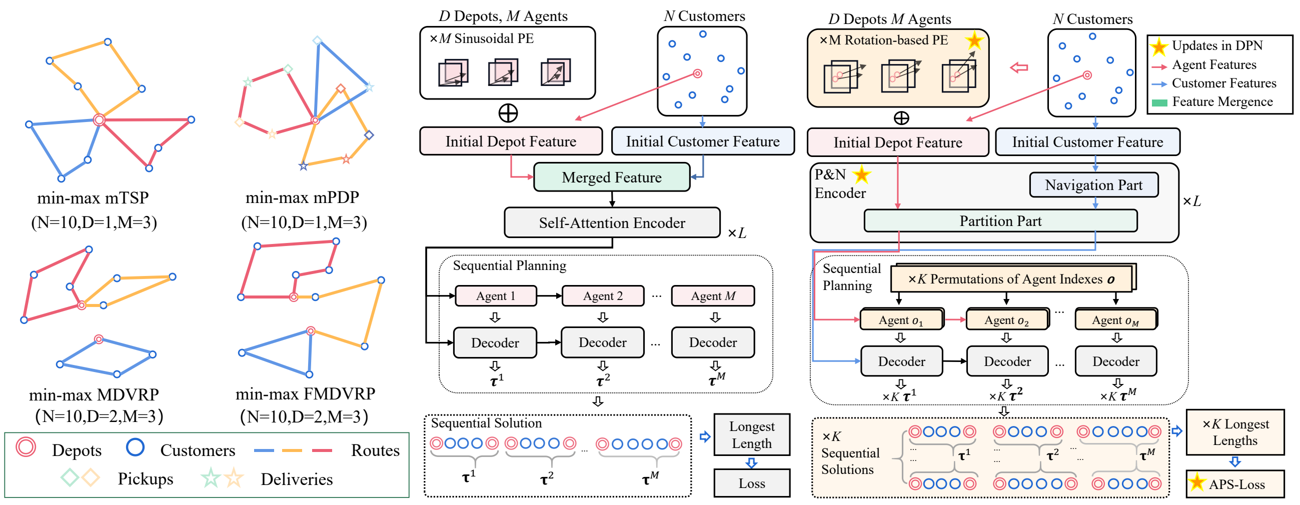
Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]https://github.com/CIAM-Group/NCO_code/tree/main/single_objective/DPN-minmaxVRP-master

1. Introduction

The vehicle routing problem (VRP) aims to determine a set of routes that satisfies the demand of all customers by minimizing a distance-based objective function (Toth & Vigo, 2002; 2014). As a variant of VRP, the min-max vehicle routing problem (min-max VRP), instead of minimizing the total length of all routes (i.e., min-sum), seeks to reduce the length of the longest one among all the routes (i.e., min-max). Many real-world applications, such as transportation planning (Delgado Serna & Pacheco Bonrostro, 2001), disaster management (Cheikhrouhou & Khoufi, 2021), and robotics (Faigl et al., 2016; David & Rögnvaldsson, 2021) are more consistent with the min-max VRP due to their operational requirements. Although the min-max VRP is of great significance and has attracted widespread attention, solving it is still very challenging (Kumar & Panneerselvam, 2012). The min-max VRP emphasizes balancing the lengths between different routes, thereby it considers not only the order of customers within each route but also the partition of customers (França et al., 1995). Efficient solvers should simultaneously address both a customer partition task and a navigation task for customers assigned to each route (i.e., partition and navigation tasks) (Arkin et al., 2006; Carlsson et al., 2009; Narasimha et al., 2013; Vandermeulen et al., 2019).

Recently, reinforcement learning (RL)-based neural solvers have been successfully applied to classical min-sum VRPs such as the traveling salesman problem (TSP) (Bello et al., 2017) and the capacitated vehicle routing problem (CVRP) (Nazari et al., 2018). These solvers show competitive performances in efficiency and accuracy compared to traditional heuristic methods (**?**Gao et al., 2023a). Meanwhile, some learning-based methods have emerged to handle some min-max VRP variants. These methods include two-stage methods (Kaempfer & Wolf, 2018; Hu et al., 2020; Liang et al., 2023), learning improvement heuristics (Kim et al., 2022a), parallel planning methods (Cao et al., 2021; Park et al., 2021; Gao et al., 2023b), and sequential planning methods (Son et al., 2024). Two-stage methods and learning improvement heuristics highly rely on handcrafted heuristic algorithms (Liang et al., 2023). Parallel planning methods employ multiple decentralized models to construct the set



(a) Four examples of min-max VRPs (b) Existing framework (c) DPN (Ours)

Figure 1. (a) The instances and solutions of four involved min-max VRPs. (b) The sequential planning framework proposed in Son et al. (2024). (c) The sequential planning framework of the proposed DPN. To decouple the partition and navigation features and improve the representation ability, we conduct modifications to the existing framework by proposing a novel P&N Encoder, utilizing agent-permutation-symmetries (APS) in loss calculation, and introducing a Roataion-based positional encoding (Rotation-based PE) for agent representations.

of routes in parallel, which derives a complex joint action space and causes difficulties in training (Son et al., 2024). Sequential planning methods, on the other hand, sequentially construct the set of routes with a single model. This approach significantly reduces the decision space complexity in training, facilitating the exploration of the optimal solution.

As shown in Figure 1(b), existing sequential planning methods process min-max VRP instances with an encoder-decoder model structure and utilize multiple agent embeddings to construct the routes one by one. However, as specialized solvers of min-max VRPs, existing methods fail to leverage problem-specific properties, potentially impairing their representation ability. Firstly, directly processing the merged feature through several self-attention-based encoder layers leads to ambiguous representations for customers and agents. This ambiguity blurs the distinction between the partition and navigation tasks, potentially resulting in routes with imbalanced lengths. Moreover, the decoding strategy that employs agents to generate routes in a fixed order (i.e., from the first agent to the last one) could potentially trap the solver in local minima. Lastly, the adopted sinusoidal positional encoding (PE) is crucial for learning partition, but excluding the depot coordinates in its formula might weaken the model's generalization ability across different depot locations.

This paper aims to fully exploit the problem-specific properties of the min-max VRP, particularly the requirements of partition and navigation. To preserve independent representations for partition and navigation, we propose the Decoupling-Partition-Navigation (DPN) method. As shown in Figure 1(c), it adopts a novel attention-based Partition-and-Navigation encoder (P&N Encoder) consisting of a navigation part and a partition part in each layer. Moreover, we propose an agent-permutation-symmetric (APS) loss function, leveraging the equivalence of optima to improve training efficiency. Beyond that, we use a depot-location-aware Rotation-based PE to enhance partition-related representations. We conduct experiments on the four min-max VRPs exhibited in Figure 1(a), i.e., min-max multi-agent traveling salesman problem (min-max mTSP), min-max multi-agent pickup and delivery problem (min-max mPDP), min-max multi-depot VRP (min-max MDVRP), and min-max flexible multi-depot VRP (min-max FMDVRP). Experimental results indicate that DPN can significantly outperform the other neural solvers on all four min-max VRPs.

The contributions of this paper can be summarized as follows: **1):** We propose an RL-based sequential planning method for min-max VRPs that utilizes a novel attention-based P&N Encoder to process decoupled representations of the partition and navigation tasks. **2):** The proposed DPN leverages symmetries in the permutation of agents and presents a novel loss function that speeds up the convergence in the RL training. **3):** Experiments demonstrate that the proposed DPN can significantly outperform existing neural solvers on four representative min-max VRPs.

2. Preliminaries

2.1. Sequential Planning for Solving Min-max VRPs

A min-max VRP instance with M routes (generated by M agents), D depots, and N customers is defined over a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Each $v_i \in \mathcal{V}$ represents a depot or a customer and $e_{ij} \in \mathcal{E}$ represents the edge between node v_i and v_j. The solution (i.e., a set of routes) $\mathcal{T}$ is formed by node indexes in $\mathcal{V}$, and each customer in $\mathcal{V}$ can only get visited once. Moreover, the number of routes in solution $\mathcal{T}$ is restricted to M (i.e., $\mathcal{T} = \{\boldsymbol{\tau}^1, \dots, \boldsymbol{\tau}^M\}$). Each route $\boldsymbol{\tau}^i$ for $i \in \{1, \dots, M\}$ only starts and ends at a depot. The objective function of the min-max VRP can be formulated as

$$\underset{\mathcal{T} \in \Omega}{\text{minimize}} \quad f(\mathcal{T}) = \max_{i \in \{1,\dots,M\}} L(\boldsymbol{\tau}^i), \tag{1}$$

where Ω is a set consisting of all feasible solutions, and $L(\boldsymbol{\tau}^i)$ calculates the Euclidean length of route $\boldsymbol{\tau}^i$.

Unlike the parallel planning method that simultaneously constructs the M routes, the sequential planning methods generate the M routes one by one. During the construction of each route, the sequential planning model gradually selects the next customer to extend the route and finally returns to the depot. The sequential planning process can be formalized as a Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \boldsymbol{r}, \mathcal{P}\}$ (Son et al., 2024). At the t-th step, the action $a_t \in \mathcal{A}$ represents the index of the selected node from unvisited customers or the set of depots, and the state $s_t \in \mathcal{S}$ records the partial solution currently constructed, the instance $\mathcal{G}$, and the number of agents M. The terminal state s_T contains a feasible solution $\mathcal{T} \in \Omega$ with a min-max reward $r_T = -f(\mathcal{T})$ formulated in Eq. (1). Parameterized by θ, the policy p_θ for sampling the set of routes τ can be calculated as

$$p_\theta(\mathcal{T}|\mathcal{G}, M) = \prod_{t=1}^{T-1} p_\theta(a_t|s_t) = \prod_{i=1}^{M} p(\boldsymbol{\tau}^i|\boldsymbol{\tau}^{1:i}, \mathcal{G}, \theta), \tag{2}$$

where $\boldsymbol{\tau}^{1:i}$ represents all the routes generated before $\boldsymbol{\tau}^i$. More details of the MDP are presented in Appendix B.1 and a comprehensive literature review about neural solvers is in Appendix A.

2.2. Partition and Navigation in Min-max VRPs

The tasks of assigning customers to M routes (i.e., partition) and optimizing the routing of customers assigned to each route (i.e., navigation) are considered simultaneously in min-max solvers (Carlsson et al., 2009; Narasimha et al., 2013). These two tasks can be defined as follows:

Partition. Customers and depots assigned to the route $\boldsymbol{\tau}^i$ for $i \in \{1, \dots, M\}$ forms a partition of $\mathcal{G}$. Each sub-graph is denoted as $\mathcal{G}^i$. The partition function $P_{\theta,M}(\mathcal{G}) = \{\mathcal{G}^1, \dots, \mathcal{G}^M\}$ with parameter θ generates sub-graph partitions with

$$\bigcap_{i \in \{1,\dots,M\}} \mathcal{G}^i \subseteq \text{Depots}, \quad \bigcup_{i \in \{1,\dots,M\}} \mathcal{G}^i = \mathcal{G}. \tag{3}$$

Navigation. The navigation task (generally being TSP) optimizes routings in each sub-graph $\mathcal{G}^i$ for $i \in \{1, \dots, M\}$. For sequential planning methods, if the number of nodes in $\mathcal{G}^i$ is n^i, the navigation policy π_θ of $\mathcal{G}^i$ can written as follows:

$$\pi_\theta(\boldsymbol{\tau}^i|\mathcal{G}^i) = \prod_{t=1}^{n^i} p(\boldsymbol{\tau}^i(t)|\boldsymbol{\tau}^i(1:t), \mathcal{G}^i, \theta), \tag{4}$$

where $\boldsymbol{\tau}^i(t)$ is the t-th node index in $\boldsymbol{\tau}^i$ and $\boldsymbol{\tau}^i(1:t)$ represents indexes of nodes selected before $\boldsymbol{\tau}^i(t)$. Integrating the partition and navigation policy into Eq. (1), the optimal parameter θ^* can be obtained equivalently as follows:

$$\theta^* = \arg\min_\theta \max_{G^i \in P_{\theta,M}(\mathcal{G})} \mathbb{E}_{\pi_\theta(\boldsymbol{\tau}^i|\mathcal{G}^i)}\left[L(\boldsymbol{\tau}^i)\right]. \tag{5}$$

2.3. Transformer Blocks

The Transformer block (Vaswani et al., 2017) is widely adopted in sequential planning models. It contains a multi-head-attention function and a feed-forward function.

Attention Mechanism. The attention mechanism is a fundamental component in deep learning models (Niu et al., 2021). The classical attention function allows two embeddings to share information, and it can be expressed as

$$Y = \text{Attn}(X, C) = \text{Softmax}\left(\frac{XW_Q(CW_K)^\intercal}{\sqrt{d}}\right) CW_V, \tag{6}$$

where $X \in R^{n \times d}$ and $C \in R^{m \times d}$ are inputted embeddings. $W_Q, W_K \in R^{d \times d_k}$, and $W_V \in R^{d \times d_v}$ are trainable query, key, and value parameters, respectively. The Softmax function is applied independently across each row, and the output of the attention function is a matrix $Y \in R^{n \times d}$. As a special case, the self-attention function calculates the attention function with two same inputs (i.e., Attn(X', X')). For more efficiency, the multi-head attention function applies the attention function in Eq. (6) on H different "heads" with independent parameters, i.e.,

$$\begin{aligned} \text{MHA}(X, C) &= \text{Concat}(Y_1, \dots, Y_H)W_p, \\ \text{where} \quad Y_i &= \text{Attn}_i(X, C), \forall i \in \{1, \dots, H\}, \end{aligned} \tag{7}$$

where $d_k = d_v = \frac{d}{H}$ in each Attn$_i$. $W_p \in R^{d \times d}$ denotes a trainable square projection matrix that combines different attention heads. The transformer blocks (Vaswani et al., 2017; Luo et al., 2024) also include the feed-forward function, i.e.,

$$\text{FF}(X) = (\text{ReLU}(XW_1))\, W_2, \tag{8}$$

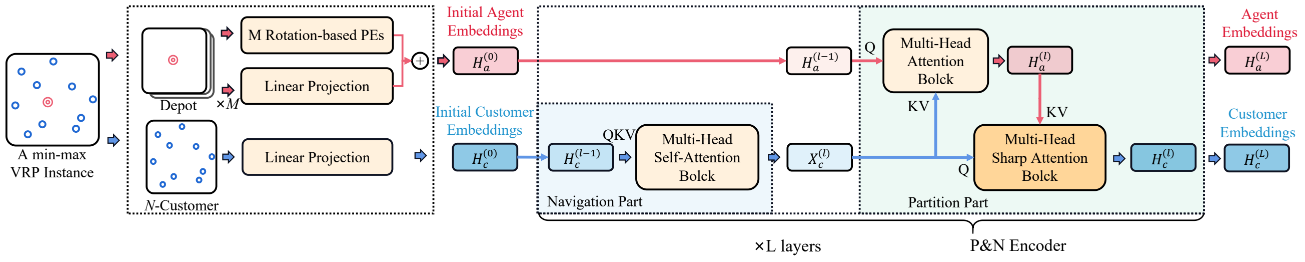


Figure 2. Initial embeddings and the proposed P&N Encoder.

where FF and ReLU represent the feed-forward function and the ReLU activation function (Nair & Hinton, 2010), respectively. $W_1 \in R^{d\times d_h}$, and $W_2 \in R^{d_h\times d}$ are trainable projections and d_h represents the hidden space dimension.

3. Methodology

As shown in Figure 1(c), DPN adopts the general framework of the sequential planning method with a multi-layer encoder and a single-layer decoder. The L-layer encoder processes the initial embeddings of customer coordinates to customer embeddings and encodes the initial embeddings of depot coordinates and the M-route constraint (represented by PEs) to agent embeddings. By handling these embeddings and a contextual representation of the MDP state s_t, the decoder is repeatedly employed to generate actions. The proposed DPN integrates problem-specific properties in three key components of the original sequential planning methods. Firstly, DPN presents a bi-part attention-based P&N Encoder, which utilizes separate structures to capture decoupled features for partition and navigation. Secondly, we develop a novel agent-permutation-symmetric loss (APS-Loss) function that leverages symmetries in decoding routes for effective explorations in customer partition. Finally, we adopt the Rotation-based PE to incorporate the coordinates of depots into the PE calculation to achieve superior representations for customer partition.

The proposed DPN can efficiently address both single-depot and multi-depot min-max VRPs. For clarity, this Section only illustrates the details of DPN in solving the single-depot ones (e.g., the min-max mTSP). The P&N Encoder for multi-depot min-max VRPs can be found in Section 5.1, with additional implementation details, motivations, and necessity statements provided in Appendix C.

3.1. P&N Encoder

Figure 2 depicts the input and architecture of the proposed P&N Encoder. Linear projections and several PEs convert an instance into initial embeddings, and each layer of the P&N Encoder comprises two main components: a navigation part and a partition part.

Initial Embeddings. A single-depot instance with N customers and M routes (i.e., number of agents) is mapped to initial agent embeddings $H_a^{(0)} \in \mathcal{R}^{M\times d}$ and initial customer embeddings $H_c^{(0)} \in \mathcal{R}^{N\times d}$ before being fed to the P&N Encoder. In processing these embeddings, the coordinate of the depot $\boldsymbol{x}_d \in \mathcal{R}^{1\times 2}$ and the coordinates of customers $\boldsymbol{x} \in \mathcal{R}^{N\times 2}$ are embedded into $H_a^{(0)}$ and $H_c^{(0)}$ via linear projections. M various d-dimensional PEs are also added in $H_a^{(0)}$ to distinguish the multiple agents.

Motivation. As Figure 1(b), the previous sequential planning methods adopt multi-head self-attention to process the merged features of agents and customers (i.e., compute multi-head self-attention to Concat$[H_a^{(0)}, H_c^{(0)}]$). However, as further discussed in Appendix C.1, these calculations can be decomposed into four distinct kinds of relations, and the Softmax function in self-attention functions directly normalizes the customer-customer relations and the agent-customer relations, potentially confusing the heterogeneous representations for partition and navigation. To achieve decoupled representations, the proposed P&N Encoder structurally separates the representation encoding process of these two tasks into navigation and partition parts, respectively. In the navigation part, customer embeddings conduct self-attention for navigation features. In the partition part, two attention blocks are calculated between agents and customers to enhance representations for the customer partition task.

Navigation Part. The partition part in the l-th layer P&N Encoder calculates customer embeddings $H_c^{(l-1)}$ independently by a multi-head self-attention block. This part focuses on customer routing based on $H_c^{(l-1)}$ and excludes the impact of learning partitions. It consists of an MHA function, a feed-forward function (FF), and two skip connections with trainable parameters α_1 and α_2 (Rezero normalization (Bachlechner et al., 2021)), i.e.,

$$\begin{aligned}\hat{X}_c^{(l)} &= \alpha_1 * \text{MHA}(H_c^{(l-1)}, H_c^{(l-1)})) + H_c^{(l-1)},\\ X_c^{(l)} &= \alpha_2 * \text{FF}(\hat{X}_c^{(l)}) \qquad\qquad\quad + \hat{X}_c^{(l)}.\end{aligned} \tag{9}$$

Partition Part. The following partition part is designed to process representations related to the customer partition task. Agent features $H_a^{(l-1)}$ and customer features from the navigation part $X_c^{(l)}$ are fused through two cascaded attention blocks. In this part, agents are supposed to integrate features from their assigned customers (i.e., sub-graph $\mathcal{G}_i$ for i-th agent), while customers can derive representations about their sub-graph assignments as well. Moreover, according to Eq. (3), each customer can only be assigned to one sub-graph, so each customer should only integrate the embeddings of one among the M agents. To meet this requirement, we remove the Softmax temperature in the Attn function (Eq. (6)) for fast convergence and design a multi-head sharp attention function (denoted as MHSA) as

$$\begin{aligned} &\text{MHSA}(X, C) = \text{Concat}(S_1, \ldots, S_H)W_P, \\ &\text{where} \quad S_i = \text{Softmax}\left(XW_Q(CW_K)^\intercal\right)CW_V. \end{aligned} \tag{10}$$

The total function of the partition part employs four residual links with parameters $\alpha_3, \ldots, \alpha_6$, i.e.,

$$\hat{H}_a^{(l)} = \alpha_3 * \text{MHA}(H_a^{(l-1)}, X_c^{(l)}) + H_a^{(l-1)}, \tag{11}$$

$$H_a^{(l)} = \alpha_4 * \text{FF}(\hat{H}_a^{(l)}) + \hat{H}_a^{(l)}, \tag{12}$$

$$\hat{H}_c^{(l)} = \alpha_5 * \text{MHSA}(X_c^{(l)}, H_a^{(l)}) + X_c^{(l)}, \tag{13}$$

$$H_c^{(l)} = \alpha_6 * \text{FF}(\hat{H}_c^{(l)}) + \hat{H}_c^{(l)}. \tag{14}$$

3.2. APS-Loss

Leveraging symmetry can effectively promote the training process of neural combination optimization solvers (Kwon et al., 2020; 2021; Kim et al., 2022b). Correspondingly, as another important contribution, DPN develops a novel agent-permutation-symmetry (APS) in solving min-max VRPs. For an instance $\mathcal{G}$, and an index order of agents $\boldsymbol{o} = (o_1, o_2, \ldots, o_M)$ (i.e., permutation) for generating the M routes in solution $\mathcal{T}$, the APS maintains an equivalence optimal objective functions for any permutations $\boldsymbol{o}$. The APS holds because, for all the $\boldsymbol{o} = (o_1, \ldots, o_M)$, and $\{\mathcal{G}^i | i \in \{1, \ldots, M\}\}$ being optimal sub-graph partitions of $\mathcal{G}$, the set of routes in the optimal solution $\mathcal{T}^* = \{\boldsymbol{\tau}^{*i} \sim \pi_\theta(\cdot|\mathcal{G}^{o_i})$ for $i \in \{1 \ldots, M\}\}$ keeps unordered. Hence, the objective function (i.e., the longest route length) is identical for any permutations. $\pi_\theta(\cdot|\mathcal{G}^{o_i})$ represents the policy derived from Eq. (4).

As shown in Figure 1(c), DPN leverages the APS for a better baseline with K sampled permutations $\boldsymbol{o}^{(1)}$ to $\boldsymbol{o}^{(K)}$. The APS-equipped baseline $b(\mathcal{G})$ is the average objective function of the K generated solutions estimated as

$$b(\mathcal{G}) = \frac{1}{K}\sum_{k=1}^{K} \max_{i \in \{1,\ldots,M\}} L(\boldsymbol{\tau}^i \sim p_\theta(\cdot|\mathcal{G}^{o_i^{(k)}})). \tag{15}$$

APS loss function $\mathcal{L}(\mathcal{G})$ (i.e., APS-Loss) is processed by the REINFORCE (Williams, 1992) algorithm as:

$$\pi'_\theta(\mathcal{T}|\mathcal{G}, M, \boldsymbol{o}^{(k)}) = \prod_{i=1}^{M} \pi_\theta(\boldsymbol{\tau}^i|\mathcal{G}^{o_i^{(k)}}), \tag{16}$$

$$\begin{aligned} \nabla_\theta \mathcal{L}(\mathcal{G}) = -\frac{1}{K}\sum_{k=1}^{K} &\mathbb{E}_{\pi'_\theta(\mathcal{T}|\mathcal{G},M,\boldsymbol{o}^{(k)})} \\ &\left[(f(\mathcal{T}) - b(\mathcal{G}))\nabla_\theta \log \pi'_\theta(\mathcal{T}|\mathcal{G}, M, \boldsymbol{o}^{(k)})\right], \end{aligned} \tag{17}$$

where features of the sub-graph $\mathcal{G}^{o_i^{(k)}}$ are supposed to be represented in the $o_i^{(k)}$-th agent embedding in DPN. The APS-Loss can promote explorations in learning representations and alleviate the bias in embeddings caused by predetermined decoding orders.

3.3. Rotation-based PE

In sequential planning methods (Son et al., 2024), different agents starting from the same depot are distinguished by positional encodings. Existing sequential planning methods adopt the sinusoidal PE (Vaswani et al., 2017) for agent distinction. The d-dimensional sinusoidal PE (denoted as SPE $\in \mathcal{R}^{M\times d}$) in the case is defined as

$$\text{SPE}(m, q) = \begin{cases} \sin(m/10{,}000^{\frac{\lfloor q/2 \rfloor}{d}}), q \equiv 0 (mod\ 2) \\ \cos(m/10{,}000^{\frac{\lfloor q/2 \rfloor}{d}}), q \equiv 1 (mod\ 2) \end{cases}. \tag{18}$$

In Appendix C.5, we illustrate how a PE conveys the angle-based information when representing the sub-graphs in the partition part of the P&N Encoder. The angle-based interval between sub-graphs is highly correlated with the depot location (França et al., 1995), so SPE excluding the depot coordinates will harm the partition optimality. To introduce the depot coordinates to PEs, we refer to the concept of Rotary PE (Su et al., 2024) and design the Rotation-based PE $\in \mathcal{R}^{M\times d}$ for a depot-aware agent distinction as

$$\text{PE}(m, q) = \text{Re}\left[(\boldsymbol{x}_d W_a + b_a)\exp(im/1{,}000^{\frac{\lfloor q/2 \rfloor}{d}})\right], \tag{19}$$

where i is an imaginary unit (i.e., $i^2 = -1$) and $\boldsymbol{x}_d W_a + b_a$ is a trainable linear projection of the depot coordinate $\boldsymbol{x}_d \in \mathcal{R}^2$. PE is calculated in the complex vector space, and Re represents the real part of a complex number. Additionally, Appendix C.6 provides an intuitive way of calculating PE.

4. Experiments: Single-depot Min-max VRPs

To evaluate the effectiveness of the proposed DPN method on single-depot min-max VRPs, we implement it on min-max mTSP and min-max mPDP. Detailed formulations of the two problems are provided in Appendix B.

Training Settings. For both the min-max mTSP and min-max mPDP, we conduct training from scratch on 50-scale

Table 1. The average objective function values(i.e., Obj.), gaps to the best algorithm (i.e., Gap), and testing times (i.e., Time) obtained by each algorithm on 12 datasets. Each set contains 100 randomly generated medium-scale instances. The overall best results are highlighted in bold, and the algorithm that produces the best results among all five learning-based results is highlighted in a gray background.

Min-max mTSP50(N=49,D=1)									
M=		3			5			7	
Methods	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
HGA	**2.4184**	-	7m	**2.0154**	-	6m	**1.9386**	-	4m
LKH3	2.4301	0.4816%	2m	2.0182	0.1351%	4m	1.9408	0.1157%	7m
OR-Tools(600s)	2.5878	7.0030%	98s	2.1584	7.0938%	3m	2.0992	8.2849%	3m
DAN	2.9899	23.629%	22s	2.3225	15.238%	23s	2.1492	10.864%	30s
Equity-Transformer	2.5643	6.0301%	<1s	2.0798	3.1935%	<1s	1.9618	1.1964%	<1s
Equity-Transformer-×8aug	2.4901	2.9641%	<1s	2.0399	1.2139%	<1s	1.9465	0.4076%	<1s
DPN	2.5149	3.9888%	<1s	2.0545	1.9405%	<1s	1.9549	0.8413%	<1s
DPN-×8aug	2.4654	1.9433%	<1s	2.0337	0.9087%	<1s	1.9460	0.3804%	<1s
DPN-×8aug-×16per	2.4637	1.8728%	1s	2.0324	0.8416%	1s	1.9454	0.3503%	1s

Min-max mTSP100(N=99,D=1)									
M=		5			7			10	
Methods	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
HGA	**2.1893**	-	20m	1.9963	0.1240%	16m	1.9507	0.0273%	14m
LKH3	2.1924	0.1410%	16m	**1.9939**	-	17m	**1.9502**	-	17m
OR-Tools(600s)	2.3477	7.2346%	5m	2.1627	8.4671%	6m	2.1465	10.068%	7m
DAN	2.6995	23.305%	40s	2.3115	15.930%	42s	2.1556	10.534%	46s
Equity-Transformer	2.3042	5.2456%	<1s	2.0487	2.7480%	<1s	1.9583	0.4153%	<1s
Equity-Transformer-×8aug	2.2563	3.0577%	<1s	2.0225	1.4345%	<1s	1.9534	0.1652%	<1s
DPN	2.2704	3.7017%	<1s	2.0335	1.9860%	<1s	1.9587	0.4377%	<1s
DPN-×8aug	2.2346	2.0703%	<1s	2.0143	1.0223%	<1s	1.9534	0.1671%	<1s
DPN-×8aug-×16per	2.2314	1.9240%	1s	2.0126	0.9388%	1s	1.9532	0.1542%	1s

Min-max mPDP50(N=50,D=1)									
M=		3			5			7	
Methods	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
OR-Tools(600s)	3.6796	6.1250%	20m	2.9924	7.8586%	23m	2.7867	10.663%	29m
Equity-Transformer	5.1494	48.515%	<1s	3.7159	33.937%	<1s	3.1586	25.431%	<1s
Equity-Transformer-×8aug	4.5857	32.258%	1s	3.3566	20.986%	1s	2.8764	14.227%	1s
DPN	3.6247	4.5413%	<1s	2.8946	4.3347%	<1s	2.5889	2.8081%	<1s
DPN-×8aug	3.4744	0.2082%	1s	2.7803	0.2141%	1s	2.5223	0.1651%	1s
DPN-×8aug-×16per	**3.4672**	-	1s	**2.7744**	-	1s	**2.5182**	-	1s

Min-max mPDP100(N=100,D=1)									
M=		5			7			10	
Methods	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
OR-Tools(600s)	14.315	309.48%	4h	14.486	378.96%	5h	14.500	438.07%	5h
Equity-Transformer	5.5571	58.958%	<1s	4.4831	48.234%	<1s	3.7483	39.092%	<1s
Equity-Transformer-×8aug	5.0735	45.123%	1s	4.1152	36.069%	1s	3.4411	27.690%	1s
DPN	3.6500	4.4054%	<1s	3.1404	3.8364%	<1s	2.7998	3.8933%	<1s
DPN-×8aug	3.5123	0.4673%	1s	3.0430	0.6165%	1s	2.7101	0.5647%	1s
DPN-×8aug-×16per	**3.4960**	-	2s	**3.0244**	-	2s	**2.6949**	-	2s

Table 2. The average objective function values (i.e., Obj.) on 100 randomly generated larger-scale min-max mTSP or min-max mPDP instances. The overall best results and the best learning-based results are highlighted in bold and gray backgrounds, respectively.

	Min-max mTSP200			Min-max mTSP500			Min-max mTSP1,000		
M=	10	15	20	30	40	50	50	75	100
Methods	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.
HGA	1.9861	**1.9628**	**1.9627**	**2.0061**	2.0061	2.0061	**2.0448**	2.0448	2.0448
LKH3	**1.9817**	1.9628	1.9628	2.0061	2.0061	2.0061	2.0448	2.0448	2.0448
OR-Tools(600s)	2.3711	2.3687	2.3687	8.9338	8.9356	8.9308	16.436	16.436	16.436
NCE*	2.07	1.97	1.96	2.07	2.01	2.01	2.13	2.07	2.05
DAN	2.3586	2.1732	2.1151	2.2345	2.1610	2.1465	2.3390	2.2544	2.2394
ScheduleNet*	2.35	2.13	2.07	2.16	2.12	2.09	2.26	2.17	2.16
Equity-Transformer-F-×8aug	2.0500	1.9688	1.9631	2.0165	2.0084	2.0068	2.0634	2.0531	2.0488
DPN-F-×8aug	2.0030	1.9647	1.9628	2.0065	2.0061	2.0061	2.0452	2.0448	2.0448
DPN-F-×8aug-×16per	1.9993	1.9640	1.9628	2.0061	**2.0061**	**2.0061**	2.0450	**2.0448**	**2.0448**

	Min-max mPDP200			Min-max mPDP500			Min-max mPDP1,000		
M=	10	15	20	30	40	50	50	75	100
Methods	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.
OR-Tools(600s)	45.299	45.387	45.131	140.85	140.92	140.79	280.22	280.19	280.14
Equity-Transformer-F-×8aug	4.9143	3.8186	3.3417	4.4619	3.7723	3.4455	4.9328	3.9198	3.5241
Equity-Transformer-F-sample*	4.68	3.65	3.18	4.11	3.52	3.23	4.73	3.77	3.38
DPN-F-×8aug	3.3227	2.8630	2.6735	3.1615	3.0264	2.9379	3.2802	3.0673	3.0000
DPN-F-×8aug-×16per	**3.2959**	**2.8363**	**2.6519**	**3.0878**	**2.9510**	**2.8690**	**3.2263**	**2.9811**	**2.9114**

(i.e., min-max mTSP50 and min-max mPDP50) and 100-scale problems respectively, and then conduct fine-tuning for larger problem scales (200-, 500-, and 1,000-scale). For all the scales involved, the DPN utilizes a 6-layer P&N Encoder, a uniform sampling of the number of agents M from 2 to 10, and a fixed number of permutations K to 60. The MHA function comprises 8 heads $H = 8$, with the embedding dimension $d = 128$ and the hidden dimension $d_h = 512$. The Adam optimizer (Kingma & Ba, 2014) is employed with weight decay $\beta = 1$, and the learning rate $\alpha = 1e^{-4}$ in training models from scratch and being $\alpha = 1e^{-5}$ for the fine-tuning. Moreover, the hyperparameters for various problems and scales are provided in Appendix D.1. With utilizing an NVIDIA Tesla V100S GPU, the training duration for the DPN targeting min-max mTSP100 and min-max mPDP100 span three and five days, respectively, and the fine-tuning for larger scale is completed within a few hours.

4.1. Performance Evaluation

Baselines. We select representative heuristic and learning-based methods as baselines. For heuristic methods, the Lin-Kernighan-Helsgaun3 algorithm (LKH3) (Lin & Kernighan, 1973) and the hybrid genetic algorithm (HGA) (Mahmoudinazlou & Kwon, 2024) are adopted, and these algorithms conduct 10 runs for each instance. As an outstanding operations research (OR) solver, OR-Tools has demonstrated excellent generalization ability. We report its search results within 600 seconds. For learning-based methods, we adopt a parallel planning method DAN (Cao et al., 2021) and a sequential planning method Equity-Transformer (Son et al., 2024). For efficient yet unavailable neural methods such as ScheduleNet (Park et al., 2021) and NCE (Kim et al., 2022a), we list the reported results as well. The min-max mPDP has also been investigated, but heuristic methods can hardly be implemented on Euclidean coordinates (Lau & Sengupta, 2022). Therefore, our comparative algorithm only considers OR-Tools and the learning-based method Equity-Transformer.

Performance on Medium Scales. We first examine the performance of DPN on medium-scale problems, including min-max mTSP and min-max mPDP of 50- and 100-scale. For each scale, we conduct tests on three datasets distinct with various M values. Each dataset contains 100 randomly generated instances, and the distribution of M follows the settings in Kim et al. (2022a). All the involved learning-based models are trained from scratch at the testing scale. The $\times$8aug data augmentation method is available as a variant, which conducts equivalent transformations to augment the original instance and reports the best results among eight equivalent instances (Kwon et al., 2020). For DPN, we additionally provide a $\times$16per data augmentation method which reports the best solution among 16 agent permutations (i.e., $K = 16$ in testing). Table 1 exhibits the comparison results on 12 datasets, and it presents the average objective function values, the gap from the best algorithm, and the testing time for each dataset. The best result of each dataset is highlighted in bold, while the best learning-based result is highlighted in a gray background. Results demonstrate that the proposed DPN-$\times$8aug-$\times$16per version achieves the best performance among the learning methods in all the 12 datasets and significantly stands out in min-max mPDPs. Compared to heuristic methods such as HGA and LKH3, the DPN variants can achieve competitive results in a much shorter time. The proposed DPN method consistently outperforms the Equity-Transformer under the same settings, with an average reduction of 0.72% in the optimality gap of min-max mTSP and 33.7% for min-max mPDP.

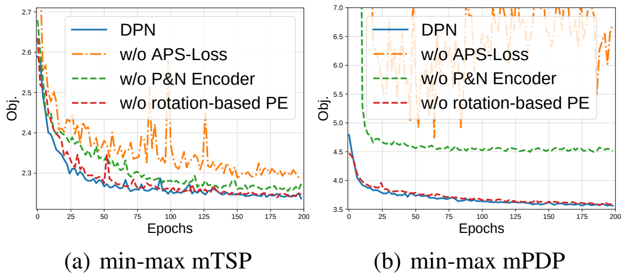


Figure 3. Training curves for ablation study (M=5).

Table 3. Ablation study on proposed components in DPN.

Min-max	mTSP100	mPDP100
M=	5	5
w/o P&N Encoder	2.2406	4.5119
w/o APS-Loss	2.2621	5.1915
w/o Rotation-based PE	2.2347	3.5158
DPN-$\times$8aug-$\times$16per	**2.2314**	**3.4960**

Performance on Larger Scales. Neural solvers of min-max VRPs also demonstrate their ability to solve larger-scale problems (Son et al., 2024). To evaluate the performance of DPN on large-scale problems, we generate datasets with 100 instances with scales of 200, 500, and 1,000, and the number of agents M follows the same setting in Son et al. (2024). Both Equity-Transformer and DPN conduct fine-tunings at scales of 200 and 500. These fine-tuned models are denoted as F in Table 2, and the 1,000-scale results are obtained using the 500-scale fine-tuned model. Results marked with "*" indicate results reported from Son et al. (2024) on the same test set. As exhibited in Table 2, the proposed DPN-F-$\times$8aug-$\times$16per acquires the best objective functions on four larger-scale min-max mTSP datasets and all nine larger-scale min-max mPDP datasets. In a total of 18 datasets, variants of DPN consistently significantly beat the Equity-Transformer and other learning-based solvers, demonstrating DPN's adaptability to large-scale scenarios.

Table 4. The average objective function values (i.e., Obj.) and testing times (i.e., Time) on 12 multi-depot min-max VRP datasets.

	MDVRP50(N=50),D=6						MDVRP100(N=100),D=8					
M=	3		5		7		5		7		10	
Methods	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
CE*	2.25	40m	1.53	17m	1.28	11m	1.85	50m	1.43	1h	1.18	1h
OR-Tools*	2.64	4m	1.68	5m	1.36	5m	2.17	6h	1.60	3h	1.29	2h
NCE*	2.25	3m	1.53	4m	1.28	5m	1.86	19m	1.43	20m	1.18	26m
DPN-×8aug-×16per	2.1491	<1s	1.4431	<1s	1.2012	<1s	1.8056	1s	1.4099	1s	1.1527	1s
DPN-F-×8aug-×16per	**2.1404**	1s	**1.4394**	1s	**1.1969**	1s	**1.7936**	2s	**1.4001**	2s	**1.1429**	2s
	FMDVRP50(N=50),D=6						FMDVRP100(N=100),D=8					
M=	3		5		7		5		7		10	
Methods	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time
CE*	2.07	35m	1.41	15m	1.19	9m	**1.74**	6h	**1.34**	4h	**1.09**	1h
OR-Tools*	2.39	4m	1.56	4m	1.27	4m	2.00	51m	1.51	54m	1.20	57m
NCE*	2.08	2m	1.40	3m	1.19	4m	1.75	11m	**1.34**	16m	**1.09**	22m
ScheduleNet*	2.61	9m	1.86	9m	1.57	10m	2.32	1h	1.86	1h	1.54	1h
DPN-×8aug-×16per	2.0471	<1s	1.3869	<1s	**1.1619**	<1s	1.7694	1s	1.3708	1s	1.1028	1s
DPN-F-×8aug-×16per	**2.0429**	**1s**	**1.3856**	1s	1.1649	1s	1.7638	2s	1.3642	2s	1.1012	2s

Experiments on both medium and large scales exhibit the effectiveness of DPN in single-depot min-max VRPs while also demonstrating the significant value of separately considering partition and navigation in sequential planning. Experiments in Appendix E further demonstrate the good generalization performance of DPN.

4.2. Analyses

To validate the necessity of the components in DPN, we conduct a series of ablation studies. As detailed in Table 3, we remove the proposed components (i.e., P&N Encoder, APS-Loss, and Rotation-based PE) and obtain the three ablation models. The 'w/o' (without) notation in Table 3 and Figure 3 indicates degrading the component by using the corresponding components proposed in Equity-Transformer. Three ablation models are evaluated with both the ×8aug and ×16per on min-max mTSP100 and min-max mPDP100 with M=5 agents. Results show that all three components of DPN make substantial contributions, with the P&N Encoder and APS-Loss playing more significant roles. The full-version DPN demonstrates notable advantages in convergence speed over the ablation models. Ablation studies on M=10 are presented in Appendix 11 where Appendix E.6 further suggests that the Rotation-based PE is less vulnerable to the depot locations.

5. Experiments: Multi-depot Min-max VRPs

This section further evaluates DPN on two representative min-max multi-depot VRPs including min-max MDVRP and min-max FMDVRP. Previous sequential planning neural solvers for min-max VRP are all limited to single-depot problems, and DPN is the first to solve multi-depot ones.

5.1. P&N Encoder for Multi-depot

In solving single-depot min-max VRP, P&N Encoder only processes the agent embeddings and customer embeddings and directly attaches the depot coordinates to agent embeddings. However, the D depots in multi-depot problems necessitate additional structures to process the embeddings of depots. To represent the depot assignment features in each route, DPN for multi-depot min-max VRPs additionally employs two additional structures in the partition part of each layer to represent depot-related partitions. The specific structure is described in detail in Appendix C.4.

5.2. Performance Evaluation

When training DPN for multi-depot min-max VRPs with both 50- and 100-scales, the number of depots D is uniformly sampled from 2 to 10. In testing, the number of depots is set to 6 for 50-scale datasets and 8 for 100-scale ones. Training on the corresponding number of depots of test sets can obtain better results, so we further provide a fine-tuned version consistently denoted as F in Table 4. We conduct experiments on 12 100-instance datasets of min-max MDVRP and min-max FMDVRP. A heuristic algorithm CROSS exchange (CE) (Taillard et al., 1997), the OR-Tools, a neural solver ScheduleNet (Park et al., 2021), and the most efficient among all existing learning-based neural solver of multi-depot min-max VRPs Neuro CE (NCE) (Kim et al., 2022a) are employed as baselines. Since some of these methods are unavailable, we provide the reported results in Kim et al. (2022a). Based on the results in Table 4, we can conclude that DPN is well applied to multi-depot min-max VRP. Especially, DPN-F-×8aug-×16per achieves significant advantages on min-max MDVRP and 50-scale min-max FMDVRP. CE and NCE outperform the proposed DPN on 100-scale min-max FMDVRP.

6. Conclusion

In this paper, we have introduced a novel Decoupling-Partition-Navigation (DPN) method for solving min-max VRPs. It is the first attempt to decouple the partition and navigation in sequential-planning-based min-max VRP solvers. DPN consists of a novel P&N Encoder, a novel APS-Loss, and a Rotation-based PE. Experimental results on four min-max VRPs have demonstrated that DPN significantly outperforms existing learning-based methods. In addition, comprehensive ablation studies have been performed to verify the effectiveness of each component of DPN.

Limitation and Future Work. Although DPN efficiently processes constraints (e.g., route number M) and representations in min-max VRP, it cannot be directly applied to general VRPs (further discussed in C.7). In the future, we plan to extend the idea of decoupled representation in DPN to general VRPs, pursuing better representations of each route.

Acknowledge

This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFA1004102), the National Natural Science Foundation of China (Grant No. 62106096), the Natural Science Foundation of Guangdong Province (Grant No. 2024A1515011759), the National Natural Science Foundation of Shenzhen (Grant No. JCYJ20220530113013031).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Arkin, E. M., Hassin, R., and Levin, A. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.

Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., and McAuley, J. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Bk9mxlSFx.

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

Bertazzi, L., Golden, B., and Wang, X. Min–max vs. min–sum vehicle routing: A worst-case analysis. *European Journal of Operational Research*, 240(2):372–381, 2015.

Cao, Y., Sun, Z., and Sartoretti, G. Dan: Decentralized attention-based neural network to solve the min-max multiple traveling salesman problem. *arXiv preprint arXiv:2109.04205*, 2021.

Carlsson, J., Ge, D., Subramaniam, A., Wu, A., and Ye, Y. Solving min-max multi-depot vehicle routing problem. *Lectures on global optimization*, 55:31–46, 2009.

Cheikhrouhou, O. and Khoufi, I. A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy. *Computer Science Review*, 40: 100369, 2021.

David, J. and Rögnvaldsson, T. Multi-robot routing problem with min–max objective. *Robotics*, 10(4):122, 2021.

Delgado Serna, C. R. and Pacheco Bonrostro, J. Minmax vehicle routing problems: application to school transport in the province of burgos. In *Computer-aided scheduling of public transport*, pp. 297–317. Springer, 2001.

Drakulic, D., Michel, S., Mai, F., Sors, A., and Andreoli, J.-M. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

Faigl, J. et al. An application of self-organizing map for multirobot multigoal path planning with minmax objective. *Computational intelligence and neuroscience*, 2016, 2016.

França, P. M., Gendreau, M., Laporte, G., and Müller, F. M. The m-traveling salesman problem with minmax objective. *Transportation Science*, 29(3):267–275, 1995.

Gao, C., Shang, H., Xue, K., Li, D., and Qian, C. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. *arXiv preprint arXiv:2308.14104*, 2023a.

Gao, H., Zhou, X., Xu, X., Lan, Y., and Xiao, Y. Amarl: An attention-based multiagent reinforcement learning approach to the min-max multiple traveling salesmen problem. *IEEE Transactions on Neural Networks and Learning Systems*, 2023b.

Hou, Q., Yang, J., Su, Y., Wang, X., and Deng, Y. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6ZajpxqTlQ.

Hu, Y., Yao, Y., and Lee, W. S. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowledge-Based Systems*, 204: 106244, 2020.

Jiang, Y., Wu, Y., Cao, Z., and Zhang, J. Learning to solve routing problems via distributionally robust optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9786–9794, 2022. doi: 10.1609/aaai.v36i9.21214. URL https://ojs.aaai.org/index.php/AAAI/article/view/21214.

Jin, Y., Ding, Y., Pan, X., He, K., Zhao, L., Qin, T., Song, L., and Bian, J. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8132–8140, 2023.

Kaempfer, Y. and Wolf, L. Learning the multiple traveling salesmen problem with permutation invariant pooling networks. *arXiv preprint arXiv:1803.09621*, 2018.

Kim, M., Park, J., and Park, J. Learning to cross exchange to solve min-max vehicle routing problems. In *The Eleventh International Conference on Learning Representations*, 2022a.

Kim, M., Park, J., and Park, J. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *arXiv preprint arXiv:2205.13209*, 2022b.

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

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Kumar, S. N. and Panneerselvam, R. A survey on the vehicle routing problem and its variants. *Scientific Research Publishing*, 2012.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., and Min, S. Pomo: Policy optimization with multiple optima for reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/f231f2107df69eab0a3862d50018a9b2-Paper.pdf.

Kwon, Y.-D., Choo, J., Yoon, I., Park, M., Park, D., and Gwon, Y. Matrix encoding networks for neural combinatorial optimization. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 5138–5149. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/29539ed932d32f1c56324cded92c07c2-Paper.pdf.

Lau, T. T.-K. and Sengupta, B. The multi-agent pickup and delivery problem: Mapf, marl and its warehouse applications. *arXiv preprint arXiv:2203.07092*, 2022.

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

Liang, H., Ma, Y., Cao, Z., Liu, T., Ni, F., Li, Z., and Hao, J. Splitnet: a reinforcement learning based sequence splitting method for the minmax multiple travelling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8720–8727, 2023.

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

Liu, S., Zhang, Y., Tang, K., and Yao, X. How good is neural combinatorial optimization? a systematic evaluation on the traveling salesman problem. *IEEE Computational Intelligence Magazine*, 18(3):14–28, 2023.

Lu, Q. and Dessouky, M. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38(4):503–514, 2004. doi: 10.1287/trsc.1030.0040.

Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36, 2024.

Ma, Y., Li, J., Cao, Z., Song, W., Zhang, L., Chen, Z., and Tang, J. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 11096–11107. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/5c53292c032b6cb8510041c54274e65f-Paper.pdf.

Mahmoudinazlou, S. and Kwon, C. A hybrid genetic algorithm for the min–max multiple traveling salesman problem. *Computers & Operations Research*, 162:106455, 2024.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Narasimha, K. V., Kivelevitch, E., Sharma, B., and Kumar, M. An ant colony optimization technique for solving min–max multi-depot vehicle routing problem. *Swarm and Evolutionary Computation*, 13:63–73, 2013.

Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M. Reinforcement learning for solving the vehicle routing problem. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf.

Niu, Z., Zhong, G., and Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

Pan, X., Jin, Y., Ding, Y., Feng, M., Zhao, L., Song, L., and Bian, J. H-tsp: Hierarchically solving the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9345–9353, 2023.

Park, J., Bakhtiyar, S., and Park, J. Schedulenet: Learn to solve multi-agent scheduling problems with reinforcement learning. *arXiv preprint arXiv:2106.03051*, 2021.

Reinelt, G. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

Son, J., Kim, M., Choi, S., Kim, H., and Park, J. Equity-transformer: Solving np-hard min-max routing problems as sequential generation with equity context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20265–20273, 2024.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, pp. 127063, 2023.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Sun, R., Zheng, Z., and Wang, Z. Learning encodings for constructive neural combinatorial optimization needs to regret. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20803–20811, 2024.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186, 1997.

Toth, P. and Vigo, D. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002. doi: 10.1137/1.9780898718515. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.

Toth, P. and Vigo, D. *Vehicle routing: problems, methods, and applications*. Society for Industrial and Applied Mathematics, 2014.

Vandermeulen, I., Roderich, G., and Andreas, K. Balanced task allocation by partitioning the multiple traveling salesperson problem. In *2019 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1479–1487. ACM, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, Y., Jia, Y.-H., Chen, W.-N., and Mei, Y. Distance-aware attention reshaping: Enhance generalization of neural solver for large-scale vehicle routing problems. *arXiv preprint arXiv:2401.06979*, 2024.

Wang, Z., Yao, S., Li, G., and Zhang, Q. Multiobjective combinatorial optimization using a single deep reinforcement learning model. *IEEE Transactions on Cybernetics*, 2023. doi: 10.1109/TCYB.2023.3312476.

Wen, M., Kuba, J., Lin, R., Zhang, W., Wen, Y., Wang, J., and Yang, Y. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022.

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

Wu, Y., Song, W., Cao, Z., Zhang, J., and Lim, A. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5057–5069, 2022. doi: 10.1109/TNNLS.2021.3068828.

Ye, H., Wang, J., Liang, H., Cao, Z., Li, Y., and Li, F. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Ye, J., Li, C., Wang, J., and Zhang, C. Towards global optimality in cooperative marl with sequential transformation. *arXiv preprint arXiv:2207.11143*, 2022.

Zha, A., Gao, R., Chang, Q., Koshimura, M., and Noda, I. Cnf encodings for the min-max multiple traveling salesmen problem. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 285–292. IEEE, 2020.

Zheng, Z., Yao, S., Li, G., Han, L., and Wang, Z. Pareto improver: Learning improvement heuristics for multi-objective route planning. *IEEE Transactions on Intelligent Transportation Systems*, 2023. doi: 10.1109/TITS.2023.3313688.

Zhou, J., Wu, Y., Song, W., Cao, Z., and Zhang, J. Towards omni-generalizable neural methods for vehicle routing problems. In *International Conference on Machine Learning*, pp. 42769–42789. PMLR, 2023.

Zong, Z., Wang, H., Wang, J., Zheng, M., and Li, Y. Rbg: Hierarchically solving large-scale routing problems in logistic systems via reinforcement learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4648–4658, 2022.

A. Related Work

A.1. Learning-based Methods for TSP and CVRP

Currently, plenty of learning-based methods have been applied to TSP and CVRP. Classified by the solution generation process, these methods can be divided into two main categories (Bengio et al., 2021). Learning improvement heuristic methods execute neural-parameter-assisted operators to modify solutions iteratively (Wu et al., 2022; Zheng et al., 2023), while the learning constructive heuristic methods directly convert the input coordinates to near-optimal solutions in an end-to-end manner (Drakulic et al., 2024). Although advanced learning improvement heuristic methods show advantages in small-scale instances with adequate running time (Ma et al., 2021; Zheng et al., 2023), learning constructive heuristic methods are considered to have better generalization ability and more application value (Liu et al., 2023; Jiang et al., 2022; Wang et al., 2023). Generally, most RL-based constructive methods including the famous Attention Model (Kool et al., 2018) employ a unified network structure including several multi-head attention encoder layers and a single-layer decoder. Advanced RL-based construction methods typically propose modifications to the original network structure (Jin et al., 2023) and decoding methods (Sun et al., 2024). LEHD (Luo et al., 2024) and ELG (Gao et al., 2023a) propose modified network structures, while POMO (Kwon et al., 2020) and Sym-NCO (Kim et al., 2022b) utilize symmetry for better reinforcement learning baselines.

Besides learning constructive heuristic methods, inspired by a similar idea of partition and navigation compared to DPN, two-stage methods (Hou et al., 2023) are also compelling in solving TSP and CVRP. To divide and conquer large-scale problems, some works such as TAM (Hou et al., 2023), RBG (Zong et al., 2022), H-TSP (Pan et al., 2023), and GLOP (Ye et al., 2024) use two sets of independent neural networks (or heuristics) to generate a large-scale solution, which are respectively responsible for problem decomposition and sub-problem solving. These two-stage solving methods have achieved high-quality solutions for large-scale problems such as TSP with 10,000 nodes. Nevertheless, we want to clarify that thought inspired by a similar idea, as a one-stage constructive decoding method, the proposed DPN is different from these two-stage solving methods.

A.2. Learning-based Methods for Min-max VRPs

The learning-based methods for solving min-max VRP can be classified into four categories: two-stage methods, learning improvement heuristics, parallel planning methods, and sequential planning methods (Son et al., 2024). The application of learning-based neural solvers begins with Kaempfer & Wolf (2018) and Hu et al. (2020), which use imitation learning and RL respectively to learn a two-stage solution generation process for min-max mTSP. SplitNet (Liang et al., 2023) also adopted the two-stage planning process. Neural solvers in these two-stage methods are only used in one stage, dividing the customers into sub-graphs, while the TSP solver is introduced in another stage for customer navigation. As a learning improvement heuristic method, NCE (Kim et al., 2022a) learns a neural-parameter-assisted CROSS exchange (CE) (Taillard et al., 1997) operator and demonstrates zero-shot generalization ability among different min-max VRPs. The following two manners implement learning constructive heuristics for min-max VRPs. As discussed in Section 2.1, parallel planning methods DAN (Cao et al., 2021) and SchedulingNet (Park et al., 2021) employ multiple networks to cooperatively process a limited number of routes (M routes), while sequential planning methods like Equity-Transformer (Son et al., 2024) process the features of both customer and agents by a unified encoder and distinguish these two parts of representations through PE (Vaswani et al., 2017). In the decoding phase, sequential planning methods sequentially generate the M routes in a solution one after another, while parallel planning methods extend the M routes simultaneously. Compared to parallel planning methods, sequential planning methods substantially reduce the decision space complexity (Son et al., 2024).

In experiments, sequential planning methods demonstrate the inherent superiority among all the learning-based methods. From our perspective, except for the sequential planning method, all the other three kinds of methods, can be explained as employing different entities or procedures to represent features for learning partition and navigation, respectively. However, the Softmax function (Vaswani et al., 2017) in the attention-based encoder for sequential planning methods (Son et al., 2024) confuses the features for learning partition and navigation in its calculation. It appeals to an efficient sequential planning method that focuses on problem-specific properties especially decoupling the representations of partition and navigation in min-max VRPs.

B. Additional Defination

B.1. Sequential Planning in Min-max VRPs

Regarding solution generations, we generally follow the sequential planning process proposed in Son et al. (2024). From the multi-agent reinforcement learning (MARL) perspective, the min-max objective function is a cooperative task for multiple routes (Gao et al., 2023b). Parallel planning methods use multiple networks to control the decoding of multiple routes separately. This framework is intuitive but has a more complex optimization space than sequential planning methods. Besides min-max VRPs, the sequential planning strategy is also proficient in general cooperative MARL tasks (Wen et al., 2022; Ye et al., 2022).

The Section 2.1 briefly displays the policy of sequential planning, the MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \boldsymbol{r}, \mathcal{P}\}$ of sequential planning is defined in detail as follows:

- **State.** The state $s_t \in \mathcal{S}$ consists of three parts, including the current partial solution, the instance $\mathcal{G}$, and the number of agents M. The partial solution in the initial state s_0 is an empty set and the terminal state s_T contains a feasible solution including a set of routes $\mathcal{T} = \{\boldsymbol{\tau}^1, \dots, \boldsymbol{\tau}^M\}$.
- **Action.** An action $a_t \in \mathcal{A}$ represents a selected index from candidate nodes in $\mathcal{V}$ which is then used to extend the current solution in s_t. Since all customers can only be accessed once, so a_t can only select from unvisited customers or the set of depots in $\mathcal{V}$. If a_t is a depot node, the following state s_{t+1} will end the current route and start a new one.
- **Reward.** Sequential planning methods only assign rewards to the final state, $r_T = -f(\mathcal{T})$.
- **Policy.** The policy of sequentially constructing the set of routes is provided in Eq. (2). With the sub-graphs constrained in Eq. (3), the policy for generating nodes in routes one by one is formulated with Eq. (4) as follows:

$$p_\theta(\mathcal{T}|\mathcal{G}, M) = \prod_{t=1}^{T-1} p_\theta(a_t|s_t) = \prod_{i=1}^{M} p(\boldsymbol{\tau}^i|\boldsymbol{\tau}^{1:i}, \mathcal{G}, \theta) = \prod_{i=1}^{M}\prod_{j=1}^{n^i} \pi_\theta(\boldsymbol{\tau}^i(j)|\boldsymbol{\tau}^i(1:j), \mathcal{G}^i), \tag{20}$$

where $\mathcal{G}^i \in P_{\theta,M}(\mathcal{G})$. Figure 4 exhibits a sketch map of the parallel planning process, the basic sequential planning method in Son et al. (2024), and the sequential planning method with agent permutation implemented in the proposed DPN. The proposed agent permutation only makes changes at the transaction to s_{t+1} when a_t is a depot node. When the construction of each route concludes, the agent for the next route's decoding is selected based on a sampled agent permutation $\boldsymbol{o}$. The detailed policy introducing a permutation $\boldsymbol{o}$ is provided in Eq. (16). The sequential planning process in Equity-Transformer can be regarded as a special situation of the implemented sequential planning with agent permutation, with $\boldsymbol{o} = (1, 2, \dots, n)$, and the sampling of agent permutations in DPN is implemented by randomly shuffling the special permutation $\boldsymbol{o} = (1, 2, \dots, n)$ for 100 times.

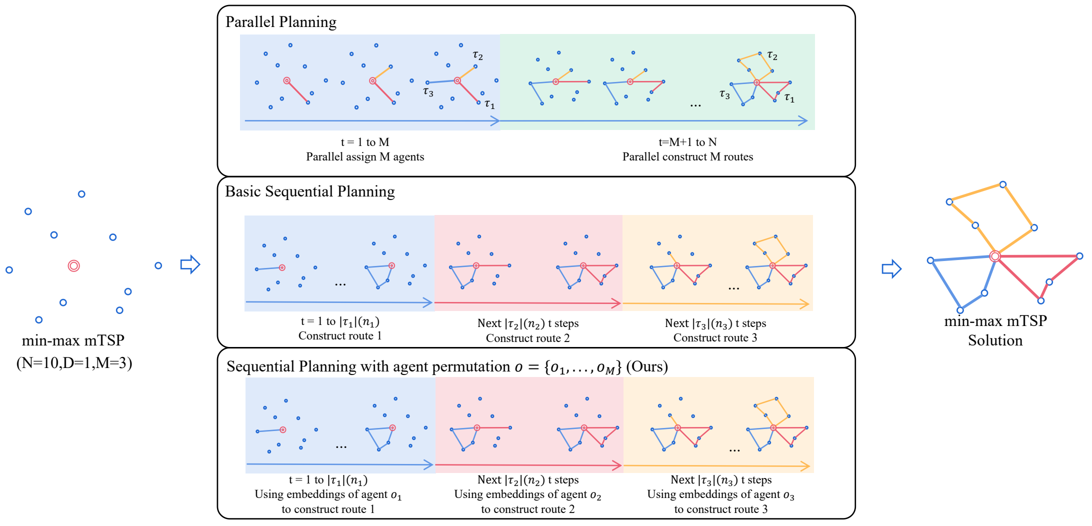

Parallel Planning
τ_2
τ_3
τ_1
t = 1 to M
Parallel assign M agents
t=M+1 to N
Parallel construct M routes
Basic Sequential Planning
t = 1 to $|\tau_1|(n_1)$
Construct route 1
Next $|\tau_2|(n_2)$ t steps
Construct route 2
Next $|\tau_3|(n_3)$ t steps
Construct route 3
Sequential Planning with agent permutation $o = \{o_1, \dots, o_M\}$ (Ours)
t = 1 to $|\tau_1|(n_1)$
Using embeddings of agent o_1
to construct route 1
Next $|\tau_2|(n_2)$ t steps
Using embeddings of agent o_2
to construct route 2
Next $|\tau_3|(n_3)$ t steps
Using embeddings of agent o_3
to construct route 3
min-max mTSP
(N=10,D=1,M=3)
min-max mTSP
Solution

Figure 4. An min-max mTSP example of constructive-based planning methods for neural solvers of min-max VRP.

B.2. Min-max Multi-agent Travel Salesman Problem

mTSP is a multi-agent extension of TSP with a single depot and a restricted number of routes. Min-max mTSP is a variant of mTSP problem with optimizing a min-max objective function. Min-max mTSP comprises the set of nodes (customers and depot) $\mathcal{V}$, the number of routes M, and the set of Depot $\mathcal{D}$ ($|\mathcal{D}| = D = 1$). We define d_{ij} as the length between node i and j, and the decision variable x_{ijm} which denotes whether the edge between node i and j are taken by agent m. The other decision variable u_{ik} is an integer representing the position (i.e., occur place) of node i in route m. For each customer node, it is positive in only one route. The MILP formulation of mTSP is given as follows (Zha et al., 2020):

$$
\begin{aligned}
&\text{minimize} && L && && (21)\\
&\text{subject to.} && \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} d_{ij}x_{ijm} \le L, && \forall m \in \{1,\dots,M\}, i \ne j, && (22)\\
& && \sum_{j\in\mathcal{V}, i\ne j} x_{ijm} = 1, && \forall m \in \{1,\dots,M\}, \forall i \in \mathcal{D}, && (23)\\
& && \sum_{i\in\mathcal{V}, j\ne i}\ \sum_{m\in\{1,\dots,M\}} x_{ijm} = 1, && \forall j \in \mathcal{V}\setminus\mathcal{D}, && (24)\\
& && \sum_{i\in\mathcal{V}, i\ne j} x_{ijm} - \sum_{h\in\mathcal{V}, h\ne j} x_{jhm} = 0, && \forall m \in \{1,\dots,M\}, \forall j \in \mathcal{V}\setminus\mathcal{D}, && (25)\\
& && u_{im} - u_{jm} + |\mathcal{V}|x_{ijm} \le |\mathcal{V}| - 1, && \forall m \in \{1,\dots,M\}, i,j \in \mathcal{V}\setminus\mathcal{D}, i \ne j, && (26)\\
& && 0 \le u_{im} \le |\mathcal{V}| - 1 && \forall m \in \{1,\dots,M\}, i \in \mathcal{V}\setminus\mathcal{D}, && (27)\\
& && x_{ijm} \in \{0,1\}, && \forall m \in \{1,\dots,M\}, i,j \in \mathcal{V}, && (28)\\
& && u_{im} \in \mathbb{Z}, && \forall m \in \{1,\dots,M\}, i \in \mathcal{V}, && (29)
\end{aligned}
$$

where L denotes the distance of the longest route (i.e., makespan) among the set of M routes. These formulations are generally adopted from Kim et al. (2022a), $u_{im} = 0$ for node i not in route $m \in \{1,\dots,M\}$.

The tasks of customer partition for different routes and customer navigation in each route of min-max mTSP are mentioned frequently in the main text. The optimal partition of min-max mTSPs assumes that the navigation task obtains a route with the optimal TSP length of any sub-graph $\mathcal{G}_i$ (Vandermeulen et al., 2019). Without affecting optimality, we ignore the impact of agent permutation and assume that $\mathcal{T} = \{\boldsymbol{\tau}^i \sim \text{TSP}(\cdot|\mathcal{G}^i)\}$ for $i \in \{1\dots,M\}$, the objective function of the partition task in min-max mTSP is as follows (Carlsson et al., 2009):

$$
\begin{aligned}
&\underset{\theta}{\text{minimize}} && \lambda\\
&\text{subject to.} && L(\text{TSP}(\cdot|\mathcal{G}^i)) \le \lambda, \forall i \in \{1,\dots,M\},\\
& && \bigcap_{i\in\{1,\dots,M\}} \mathcal{G}^i = \text{Depots}, \qquad \bigcup_{i\in\{1,\dots,M\}} \mathcal{G}^i = \mathcal{G},\\
& && \{\mathcal{G}^1,\dots,\mathcal{G}^M\} = P_{\theta,M}(\mathcal{G}),
\end{aligned}
\tag{30}
$$

where TSP$(\cdot|\mathcal{G}^i)$ represents the optimal TSP solution of the sub-graph $\mathcal{G}^i$. The navigation task optimizes the routing performance to approach the optimal TSP solution as follows:

$$
\begin{aligned}
&\underset{\theta}{\text{minimize}} && L(\boldsymbol{\tau}^i) = \sum_{i=2}^{|\boldsymbol{\tau}^i|} \|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2 + \|\boldsymbol{x}_0 - \boldsymbol{x}_{|\boldsymbol{\tau}^i|}\|_2\\
&\text{subject to.} && \boldsymbol{\tau}^i = \pi_\theta(\cdot|\mathcal{G}^i),
\end{aligned}
\tag{31}
$$

where $i \in \{1,\dots,M\}$, and $\boldsymbol{x}_i$ represents the 2-dimensional coordinate of the node with index i. $|\boldsymbol{\tau}^i|$ is adopted to indicate the number of nodes in $\boldsymbol{\tau}^i$

B.3. Min-max Multi-agent Pickup and Delivery Problem

Min-max mPDP is another single-depot min-max VRP considered in DPN. The mPDP is a multi-agent extension of the famous pickup and delivery problem (PDP). The difference between mPDP and mTSP is that the N customers involved in mPDP are classified to $\frac{N}{2}$ (i.e., $\frac{|\mathcal{V}|-1}{2}$) pairs of corresponding pickup customers and delivery customers. For $2 \leq i \leq \frac{N}{2}+1$, v_i and $v_{i+\frac{N}{2}}$ are a pair of pickup and delivery customers. There are two additional constraints defined for mPDP, pickup customers must get visited prior to their corresponding delivery customers (i.e., prior constraints), and each route must only contain paired pickup and delivery customers (i.e., pairing constraint) (Lu & Dessouky, 2004). The MILP formulation of mPDP is given as follows:

$$
\begin{aligned}
\text{minimize} \quad & L && && (32)\\
\text{subject to.} \quad & \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} d_{ij}x_{ijm} \leq L, && \forall m\in\{1,\dots,M\}, i\neq j, && (33)\\
& \sum_{j\in\mathcal{V}, i\neq j} x_{ijm} = 1, && \forall m\in\{1,\dots,M\}, \forall i\in\mathcal{D}, && (34)\\
& \sum_{i\in\mathcal{V}, j\neq i}\sum_{m\in\{1,\dots,M\}} x_{ijm} = 1, && \forall j\in\mathcal{V}\setminus\mathcal{D}, && (35)\\
& \sum_{i\in\mathcal{V}, i\neq j} x_{ijm} - \sum_{h\in\mathcal{V}, h\neq j} x_{jhm} = 0, && \forall m\in\{1,\dots,M\}, \forall j\in\mathcal{V}\setminus\mathcal{D}, && (36)\\
& u_{im} - u_{jm} + |\mathcal{V}|x_{ijm} \leq |\mathcal{V}| - 1, && \forall m\in\{1,\dots,M\}, i,j\in\mathcal{V}\setminus\mathcal{D}, i\neq j, && (37)\\
& 0 \leq u_{im} \leq |\mathcal{V}| - 1 && \forall m\in\{1,\dots,M\}, i\in\mathcal{V}\setminus\mathcal{D}, && (38)\\
& x_{ijm}\in\{0,1\}, && \forall m\in\{1,\dots,M\}, i,j\in\mathcal{V}, && (39)\\
& u_{v_i m} \leq u_{v_{i+\frac{N}{2}} m}, && \forall m\in\{1,\dots,M\},\ 2\leq i\leq \frac{N}{2}+1, && (40)\\
& (u_{v_i m} - 1)(u_{v_{i+\frac{N}{2}} m} - 1) > 0, && \forall m\in\{1,\dots,M\}, 2\leq i\leq \frac{N}{2}+1, && (41)
\end{aligned}
$$

where Eq. (40) is the prior constraint (Lu & Dessouky, 2004), which forces the pickup node to be visited before its delivery node. Eq. (41) is the pairing constraint, which ensures paired pickups and deliveries cannot occur in different routes.

In min-max mPDPs, the navigation task considers the optimization of a PDP solution to sub-graphs, and the sub-graphs in the partition task are additionally subject to a validity constraint as follows:

$$
\begin{aligned}
\underset{\theta}{\text{minimize}} \quad & \lambda \\
\text{subject to.} \quad & L(\text{PDP}(\cdot|\mathcal{G}^i)) \leq \lambda, \forall i\in\{1,\dots,M\}, \\
& \bigcap_{i\in\{1,\dots,M\}} \mathcal{G}^i = \text{Depots}, \qquad \bigcup_{i\in\{1,\dots,M\}} \mathcal{G}^i = \mathcal{G}, \qquad\qquad (42)\\
& \{\mathcal{G}^1,\dots,\mathcal{G}^M\} = P_{\theta,M}(\mathcal{G}), \\
& \text{PDP}(\cdot|\mathcal{G}^i) \neq \emptyset \quad \forall i\in\{1,\dots,M\},
\end{aligned}
$$

where the last constraint means that sub-graphs must be feasible to be solved with at least one valid PDP route.

B.4. Min-max Multi-depot Vehicle Routing Problem

Multi-depot VRP is a multi-depot extension of mTSP where the agent of each route starts from an arbitrary depot in D and must finally end at the selected starting depot. In addition, the set $\mathcal{Q}$ is additionally defined and $\mathcal{Q}_i$ indicates the set of agent (i.e., route) indexes assigned to the depot i. The MILP formulation is provided based on the description in Kim et al. (2022a), which is as follows:

$$
\begin{aligned}
\text{minimize} \quad & L && && (43)\\
\text{subject to.} \quad & \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} d_{ij}x_{ijm} \le L, && \forall k \in \{1,\dots,M\}, i \ne j, && (44)\\
& \sum_{j\in\mathcal{V} j\ne i}\ \sum_{m\in\{1,\dots,M\}} x_{ijm} = 1, && \forall i \in \mathcal{V}\setminus\mathcal{D}, && (45)\\
& \sum_{i\in\mathcal{V} j\ne i}\ \sum_{m\in\{1,\dots,M\}} x_{ijm} = 1, && \forall j \in \mathcal{V}\setminus\mathcal{D}, && (46)\\
& \sum_{i\in\mathcal{V}} x_{ijm} - \sum_{h\in\mathcal{V}} x_{jhm} = 0, && \forall m \in \{1,\dots,M\}, \forall j \in \mathcal{V}\setminus\mathcal{D}, && (47)\\
& u_{im} - u_{jm} + (|\mathcal{V}| - |\mathcal{D}| + 1)x_{ijm} \le |\mathcal{V}| - |\mathcal{D}|, && \forall m \in \{1,\dots,M\}, i,j \in \mathcal{V}\setminus\mathcal{D}, i \ne j, && (48)\\
& 0 \le u_{im} \le |\mathcal{V}| - |\mathcal{D}|, && \forall m \in \{1,\dots,M\}, i \in \mathcal{V}\setminus\mathcal{D}, && (49)\\
& x_{ijm} \in \{0,1\}, && \forall m \in \{1,\dots,M\}, i,j \in \mathcal{V}, && (50)\\
& u_{im} \in \mathbb{Z}, && \forall m \in \{1,\dots,M\}, i \in \mathcal{V}, && (51)\\
& \mathcal{Q}_i \subseteq \{1,2,\dots,M\}, && \forall i \in \mathcal{D}, && (52)\\
& \sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} \le 1, && \forall m \in \mathcal{Q}_i, \forall i \in \mathcal{D}, && (53)\\
& \sum_{i\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} \le 1, && \forall m \in \mathcal{Q}_j, \forall j \in \mathcal{D}, && (54)
\end{aligned}
$$

where Eq. (53) and Eq. (54) indicate that each vehicle starts and returns its depot at most once.

As a multi-depot extension of min-max mTSP, min-max MDVRP maintains consistent partition and navigation requirements compared to the formulations in min-max mTSP (Eq. (30) and Eq. (31)) (Carlsson et al., 2009).

B.5. Min-max Flexible Multi-depot Vehicle Routing Problem

Flexible MDVRP is an extension of MDVRP, allowing the vehicle to return to any depot in $\mathcal{D}$. Kim et al. (2022a) also provides the FMDVRP formulation by extending the MDVRP formulation (B.4). To account for the flexibility of depot returning, researchers introduce a dummy node for all depots and afterward, a depot is modeled with a start depot and an end depot. Besides $\mathcal{Q}$, min-max FMDVRP further uses $\mathcal{D}_s$ and $\mathcal{D}_e$ to represent the set of start depots and end depots and s_m represents the start node of the agent $m \in \{1, \dots, M\}$.

$$\text{minimize} \quad L \tag{55}$$

$$\text{subject to.} \quad \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} d_{ij}x_{ijm} \leq L, \qquad \forall k \in \{1,\dots,M\}, i \neq j, \tag{56}$$

$$\sum_{j\in\mathcal{V} j\neq i}\ \sum_{m\in\{1,\dots,M\},} x_{ijm} = 1, \qquad \forall i \in \mathcal{V}\setminus\mathcal{D}, \tag{57}$$

$$\sum_{i\in\mathcal{V} j\neq i}\ \sum_{m\in\{1,\dots,M\},} x_{ijm} = 1, \qquad \forall j \in \mathcal{V}\setminus\mathcal{D}, \tag{58}$$

$$\sum_{i\in\mathcal{V}} x_{ijm} - \sum_{h\in\mathcal{V}} x_{jhm} = 0, \qquad \forall m \in \{1,\dots,M\}, \forall j \in \mathcal{V}\setminus\mathcal{D}, \tag{59}$$

$$u_{im} - u_{jm} + (|\mathcal{V}| - |\mathcal{D}| + 1)x_{ijm} \leq |\mathcal{V}| - |\mathcal{D}|, \qquad \forall m \in \{1,\dots,M\}, i,j \in \mathcal{V}\setminus\mathcal{D}, i\neq j, \tag{60}$$

$$0 \leq u_{im} \leq |\mathcal{V}| - |\mathcal{D}|, \qquad \forall m \in \{1,\dots,M\}, i \in \mathcal{V}\setminus\mathcal{D}, \tag{61}$$

$$x_{ijm} \in \{0,1\}, \qquad \forall m \in \{1,\dots,M\}, i,j \in \mathcal{V}, \tag{62}$$

$$u_{im} \in \mathbb{Z}, \qquad \forall m \in \{1,\dots,M\}, i \in \mathcal{V}, \tag{63}$$

$$\mathcal{Q}_i \subseteq \{1,2,\dots,M\}, \qquad \forall i \in \mathcal{Q}, \tag{64}$$

$$\sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{s_m jm} = 1, \qquad \forall m \in \{1,\dots,M\}, \tag{65}$$

$$\sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} = 0, \qquad \forall i \in \mathcal{D}\setminus s_m, \forall m \in \{1,\dots,M\}, \tag{66}$$

$$\sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} \leq 1, \qquad \forall m \in \mathcal{Q}_i, \forall i \in \mathcal{D}_s, \tag{67}$$

$$\sum_{i\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} \leq 1, \qquad \forall m \in \mathcal{Q}_j, \forall j \in \mathcal{D}_e, \tag{68}$$

$$\sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} = 0, \qquad \forall m \in \{1,\dots,M\}, \forall i \in \mathcal{D}_e, \tag{69}$$

$$\sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} = 0, \qquad \forall m \in \{1,\dots,M\}, \forall i \in \mathcal{D}_s, \tag{70}$$

$$\sum_{i\in\mathcal{D}_s}\sum_{j\in\mathcal{V}\setminus\mathcal{D}} x_{ijm} = \sum_{i\in\mathcal{V}\setminus\mathcal{D}}\sum_{j\in\mathcal{D}_e} x_{ijm}, \qquad \forall m \in \{1,\dots,M\}, \tag{71}$$

where Eq. (65) and Eq. (66) indicate each vehicle starts at its depot. Eq. (67) to Eq. (70) indicate constraints about start and end depots. Eq. (71) indicates the balance equation of the start depots and end depots.

Min-max FMDVRP inherits the partition function in Eq. (30) but changes the calculation of length function $L(\boldsymbol{\tau^i})$ in Eq. (31) as follows due to there is no edge from the start depots to the end depots in each route:

$$L'(\boldsymbol{\tau}^i) = \sum_{i=2}^{|\boldsymbol{\tau}^i|} \|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2. \tag{72}$$

C. DPN: Details

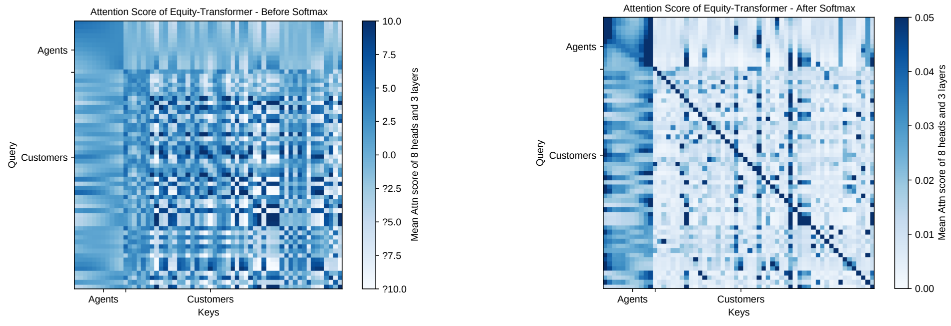
This section provides additional details of the DPN method, including the motivation of the P&N Encoder, the decoder structure for various min-max VRPs, the P&N Encoder for multi-depot min-max VRPs, and a supplemental illustration of the adopted Rotation-based PE.

C.1. Motivation of P&N Encoder

The **encoder in Equity-Transformer** calculate layers of multi-head self-attentions MHA(Concat$[H_a^{(0)}, H_c^{(0)}]$, Concat$[H_a^{(0)}, H_c^{(0)}]$) to the agent initial embedding $H_a^{(0)}$ and the customer initial embedding $H_c^{(0)}$. Ignoring the formulas of the multi-head mechanism, the attention score α in the l-th layer **encoder of Equity-Transformer** consists of the following four parts of relations (i.e., agent-agent, agent-customer, customer-agent, and customer-customer from upper left to bottom right):

$$\begin{aligned}\alpha &= \text{Concat}[H_a^{(l)}, H_c^{(l)}]W_Q\big(\text{Concat}[H_a^{(l)}, H_c^{(l)}]W_K\big)^\intercal \\ &= \begin{bmatrix} H_a^{(l)}W_Q(H_a^{(l)}W_K)^\intercal \in \mathcal{R}^{(M+1)\times(M+1)} & H_a^{(l)}W_Q(H_c^{(l)}W_K)^\intercal \in \mathcal{R}^{(M+1)\times N} \\ H_c^{(l)}W_Q(H_a^{(l)}W_K)^\intercal \in \mathcal{R}^{N\times(M+1)} & H_c^{(l)}W_Q(H_c^{(l)}W_K)^\intercal \in \mathcal{R}^{N\times N}\end{bmatrix}.\end{aligned} \tag{73}$$

In any single column of the attention score matrix, the embeddings of agents and customers are fused. Considering that agent embeddings mainly carry the features for the customer partition task and customer embeddings stand for navigation requirements. The normalization from the Softmax function may incline customers to focus on either the partition task or the navigation task thus resulting in less effective features for partition and navigation. Moreover, $M+1$ agent embeddings are inherently similar in values (i.e., only differ in PEs), so the self-attention calculation $H_a^{(l)}W_Q(H_a^{(l)}W_K)^\intercal$ might become inexplicable noise.



(a) Before Softmax (b) After Softmax

Figure 5. Attention score of min-max mTSP50 (M=10), each data is the average score of 8 heads and 3 attention layers.

Figure 5 shows the attention score α in Equity-Transformer and the attention score normalized by the Softmax function processing a min-max mTSP instance with M=10. Refer to Eq. (73), the $H_a^{(l)}W_Q(H_a^{(l)}W_K)^\intercal$ (upper left) part displays irregular noise, which also affects the calculation of the upper right part by a Softmax normalization. Moreover, the Softmax function amplifies the attention score of the $H_c^{(l)}W_Q(H_a^{(l)}W_K)^\intercal$ part (bottom left), which originally lies around 0, thereby confusing the calculation of both the customer-customer relations (bottom right) and the customer-agent relations (bottom left). To ensure the independence of the three necessary parts of relations (bottom left, bottom right, and upper right), our P&N Encoder designs a bi-part structure that calculates Softmax and the three correlations separately.

Complexity. In addition, the attention mechanisms in P&N Encoder do not increase the time complexity of each layer, slightly reducing from $\mathcal{O}\big((N+M+1)^2d\big)$ to $\mathcal{O}\big((N^2+2NM)d\big)$. The practical training and testing time generally remain the same considering the impact of additional feed-forward layers. Furthermore, the time complexity of the encoder is not the bottleneck of overall time complexity in sequential planning methods (Bello et al., 2017). The space complexity is also reduced from $\mathcal{O}\big((N+M+1)^2+(N+M+1)d\big)$ to $\mathcal{O}\big(N^2+2NM+(N+M)d\big)$, and together with the help of the APS-Loss, our DPN can adopt larger batch sizes in training compared to existing sequential planning methods.

C.2. Model Structure for Min-max mTSP

The P&N Encoder for min-max mTSP has been described in detail in the main text. The L-layer P&N Encoder processes the agent embeddings $H_a^{(0)}$ and the customer embeddings $H_c^{(0)}$ to $H_a^{(L)}$ and $H_c^{(L)}$, respectively. Sequential planning methods use contextual embedding to process the information from the instance $\mathcal{G}$, the number of routes M, and the previously generated routes. For $H_a^{(L)} = [\boldsymbol{h}_{a,1}^{(L)}, \boldsymbol{h}_{a,2}^{(L)}, \ldots, \boldsymbol{h}_{a,M}^{(L)}]$ and $H_c^{(L)} = [\boldsymbol{h}_{c,1}^{(L)}, \boldsymbol{h}_{c,2}^{(L)}, \ldots, \boldsymbol{h}_{c,N}^{(L)}]$, in decoding the m-th route ($m \leq M$) and with n customers left, if the is current agent index is o_c and the current node index is $node_c$, the contextual embedding $\boldsymbol{h}_{contex} \in \mathcal{R}^d$ is calculated as follows:

$$\begin{aligned}\boldsymbol{h}_{contex} =& \frac{W_{emb}}{N+M}\big(\sum_{i=1}^{M}\boldsymbol{h}_{a,i}^{(L)} + \sum_{i=1}^{N}\boldsymbol{h}_{m,i}^{(L)}\big) \\ &+ \text{Concat}\big[\boldsymbol{h}_{a,o_c}^{(L)}, \boldsymbol{h}_{c,node_c}^{(L)}, \frac{M-m+1}{M}, \frac{n}{N}\big]W_{step} \\ &+ \text{Concat}\big[L'(\boldsymbol{\tau}^{m+1}), \max_{i\in\{1,\ldots,N\}}\|\boldsymbol{x}_d - \boldsymbol{x}_i\|_2, \text{LD}\big]W_{length},\end{aligned} \tag{74}$$

where LD represents the maximal distance to the depot among all the n left customers. $W_{emb} \in \mathcal{R}^{d\times d}$, $W_{step} \in \mathcal{R}^{2d+2\times d}$, $W_{length} \in \mathcal{R}^{3\times d}$ are three projection matrices. When decoding the first action, $node_c$ is set to the depot in min-max mTSP. In training, DPN generates K permutations of agents and concat the K $\boldsymbol{h}_{contex}$ to $H_{contex} \in \mathcal{R}^{K\times d}$. Then the decoder of DPN contains a single 8-head glimpse attention-layer (Bello et al., 2017) which calculates the mutual attention between contextual embeddings, agent embeddings, and customer embeddings, the attention result $\boldsymbol{q}_{contex}$ for the following logit calculation is

$$\boldsymbol{q}_{contex} = \text{MHA}(H_{contex}, \text{Concat}\big[\boldsymbol{h}_{a,1}^{(L)}, \ldots, \boldsymbol{h}_{a,M}^{(L)}, \boldsymbol{h}_{c,1}^{(L)}, \ldots, \boldsymbol{h}_{c,N}^{(L)}\big]), \tag{75}$$

where $\boldsymbol{q}_{contex} \in \mathcal{R}^{K\times d}$. Afterward, given the last selected node $node_c$, the final probability $\pi_\theta(a_t|s_t, c)$ is computed as follows (Wang et al., 2024):

$$\text{dist}(node_c, j) = \alpha_d * \exp\big(\frac{\|\boldsymbol{x}_{node_c} - \boldsymbol{x}_j\|_2}{\max_{i\in\{1,\ldots,N\}}\|\boldsymbol{x}_i - \boldsymbol{x}_{node_c}\|_2}\big), \tag{76}$$

$$u_j = \begin{cases} C \cdot \tanh\Big(\frac{\boldsymbol{q}_{contex}(\boldsymbol{h}_j W^L)^\intercal}{\sqrt{d_k}} + \text{dist}(node_c, j)\Big) \odot M_a & j \in \mathcal{V} \cap j \notin s_t \\ -\infty & \text{otherwise} \end{cases}, \tag{77}$$

where $j \notin s_t$ represents the unvisited condition. W_L is a learnable square projection matrix and M_a is an agent feasibility mask, it will be $-\infty$ if the agent is not o_c, and 1 for unmasked actions. It will also mask the agent o_c when generating the last route. α_d is a trainable parameter, C is set to 50, and the above probability is calculated invariantly in all min-max VRPs. For the single-depot min-max VRPs, selecting the first M indexes corresponds to selecting the depot.

C.3. Model Structure for Min-max mPDP

Due to the special constraints introduced by the partition and navigation tasks of PDP (Appendix B.3), in the navigation part for min-max mPDP, we adopt the Heterogeneous Attention (Li et al., 2021) instead of self-attention between customers. For the partition part of the P&N Encoder, we use different linear projections to process pickup and delivery customers in the calculations of queries (i.e., Eq. (13)). In the decoder, the contextual embedding is calculated as follows:

$$\begin{aligned}\boldsymbol{h}_{contex} =& \frac{W_{emb}}{N+M}\big(\sum_{i=1}^{M}\boldsymbol{h}_{a,i}^{(L)} + \sum_{i=1}^{N}\boldsymbol{h}_{m,i}^{(L)}\big) \\ &+ \text{Concat}\big[\boldsymbol{h}_{a,o_c}^{(L)}, \boldsymbol{h}_{c,node_c}^{(L)}, \frac{M-m+1}{M}, \frac{2p}{N}\big]W_{step} \\ &+ \text{Concat}\big[L'(\boldsymbol{\tau}^{m+1}), \text{Longest-PD}, \text{Longest-P}, \text{Longest-D}, \frac{\text{Sum-PD}}{M-m}\big]W_{length},\end{aligned} \tag{78}$$

where p is the number of left pickups, Longest-PD is the longest distance of visited paired pickups and deliveries in the current route, Longest-P is the maximal distance to the depot for unvisited pickups, Longest-D is the maximal distance to the depot for unvisited deliveries, and Sum-PD represents the sum of the unvisited distance of paired pickups and deliveries.

C.4. Model Structure for Multi-depot Min-max VRPs

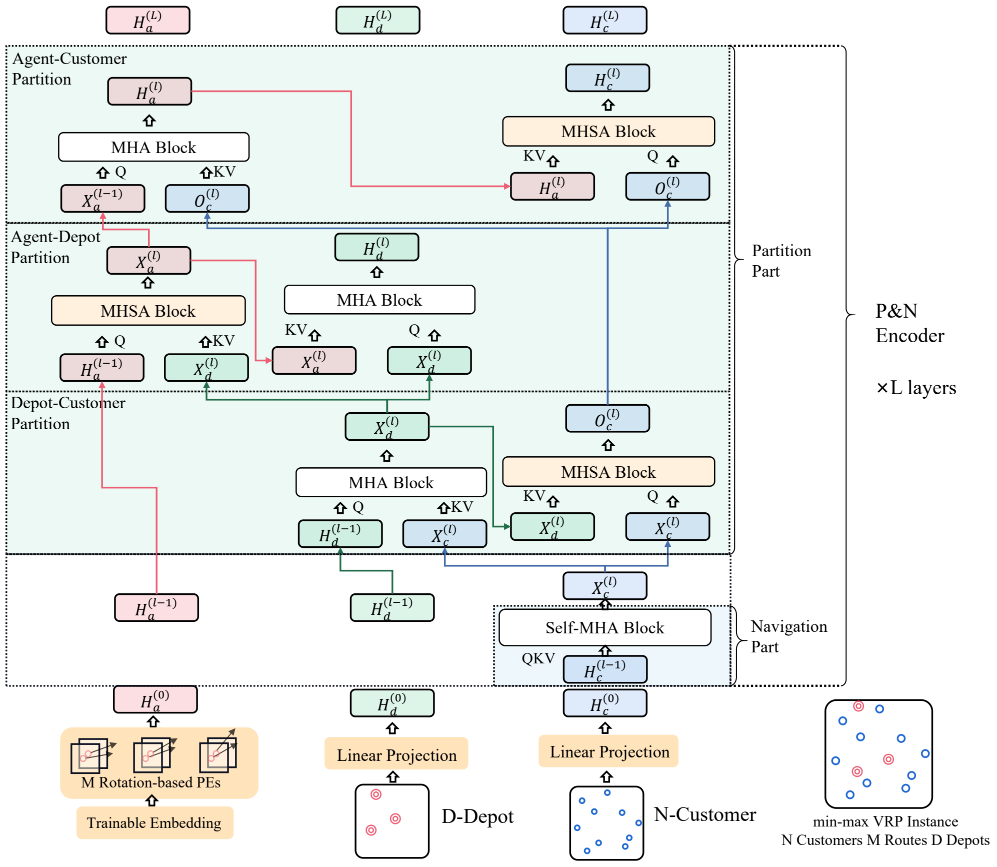


Figure 6. Model structure of the proposed P&N Encoder in min-max MDVRP and min-max FMDVRP.

Figure 6 shows that P&N Encoder designs additional partition parts to handle the D distinct depots. The input to M Rotation-based PEs is a trainable embedding representing a dummy depot. For initial embeddings, depot embeddings $H_d^{(0)}$, agent embeddings $H_a^{(0)}$, and customer embeddings $H_c^{(0)}$ are generated by linear projection.

The navigation part in the P&N Encoder for multi-depot Min-max VRPs maintains the same as what for min-max mTSP as follows:

$$\hat{X}_c^{(l)} = \alpha_1 * \text{MHA}(H_c^{(l-1)}, H_c^{(l-1)})) + H_c^{(l-1)}, \tag{79}$$
$$X_c^{(l)} = \alpha_2 * \text{FF}(\hat{X}_c^{(l)}) + \hat{X}_c^{(l)}. \tag{80}$$

The partition part consists of the depot-customer partition part, the agent-depot partition part, and the agent-customer partition part, sequentially. The depot-customer partition part is as follows:

$$\hat{X}_d^{(l)} = \alpha_3 * \text{MHA}(H_d^{(l-1)}, X_c^{(l)}) + H_d^{(l-1)}, \tag{81}$$
$$X_d^{(l)} = \alpha_4 * \text{FF}(\hat{X}_d^{(l)}) + \hat{X}_d^{(l)}, \tag{82}$$
$$\hat{O}_c^{(l)} = \alpha_5 * \text{MHSA}(X_c^{(l)}, X_d^{(l)}) + X_c^{(l)}, \tag{83}$$
$$O_c^{(l)} = \alpha_6 * \text{FF}(\hat{O}_c^{(l)}) + \hat{O}_c^{(l)}, \tag{84}$$

where MHSA is leveraged to assign only one (maybe two depots in min-max FMDVRP) depot to each customer. The agent-depot partition part is as follows:

$$\begin{aligned}\hat{X}_a^{(1)} &= \alpha_7 * \text{MHSA}(H_a^{(l-1)}, X_d^{(l)}) + H_a^{(l-1)}, &(85)\\ X_a^{(l)} &= \alpha_8 * \text{FF}(\hat{X}_a^{(l)}) \qquad + \hat{X}_a^{(l)}, &(86)\\ \hat{H}_d^{(l)} &= \alpha_9 * \text{MHA}(X_d^{(l)}, X_a^{(l)}) \qquad + X_d^{(l)}, &(87)\\ H_d^{(l)} &= \alpha_{10} * \text{FF}(\hat{H}_d^{(l)}) \qquad + \hat{H}_d^{(l)}, &(88)\end{aligned}$$

where MHSA is utilized to assign only one (maybe two depots for min-max FMDVRP) depot to each agent. The agent-customer partition part is as follows:

$$\begin{aligned}\hat{X}_d^{(l)} &= \alpha_{11} * \text{MHA}(X_d^{(l)}, O_c^{(l)}) \quad + X_d^{(l)}, &(89)\\ H_a^{(l)} &= \alpha_{12} * \text{FF}(\hat{H}_a^{(l)}) \qquad + \hat{H}_a^{(l)}, &(90)\\ \hat{H}_c^{(l)} &= \alpha_{13} * \text{MHSA}(O_c^{(l)}, H_a^{(l)}) + O_c^{(l)}, &(91)\\ H_c^{(l)} &= \alpha_{14} * \text{FF}(\hat{H}_c^{(l)}) \qquad + \hat{H}_c^{(l)}, &(92)\end{aligned}$$

where MHSA is introduced to assign only one agent to each customer. The decoder for min-max MDVRP and min-max FMDVRP only vary in the feasibility mask. If depot embeddings are $H_d^{(L)} = [\boldsymbol{h}_{d,1}^{(L)}, \boldsymbol{h}_{d,2}^{(L)}, \ldots, \boldsymbol{h}_{d,D}^{(L)}]$ and x_{d_j} represent the coordinate of the j-th depot, the contextual embedding for both is as follows:

$$\begin{aligned}\boldsymbol{h}_{contex} =& \frac{W_{emb}}{N+M+D}\Big(\sum_{i=1}^{M}\boldsymbol{h}_{a,i}^{(L)} + \sum_{i=1}^{N}\boldsymbol{h}_{m,i}^{(L)} + \sum_{i=1}^{D}\boldsymbol{h}_{d,i}^{(L)}\Big)\\ &+ \text{Concat}\big[\boldsymbol{h}_{a,o_c}^{(L)}, \boldsymbol{h}_{c,node_c}^{(L)}, \frac{M-m+1}{M}, \frac{n}{N}\big]W_{step}\\ &+ \text{Concat}\big[L'(\boldsymbol{\tau}^{m+1}), \max_{i\in\{1,\ldots,N\}}\min_{j\in\{1,\ldots,D\}}\|\boldsymbol{x}_{d_j} - \boldsymbol{x}_i\|_2, \text{LD}\big]W_{length},\end{aligned} \tag{93}$$

where LD represents the maximal distance to the nearest depot among all the n left customers. The $node_c$ in decoding the first action is set to a random depot. In decoder, the $\boldsymbol{q}_{contex}$ in multi-depot min-max VRPs is modified as follows:

$$\boldsymbol{q}_{contex} = \text{MHA}(H_{contex}, \text{Concat}\big[\boldsymbol{h}_{d,1}^{(L)}, \ldots, \boldsymbol{h}_{d,D}^{(L)}, \boldsymbol{h}_{c,1}^{(L)}, \ldots, \boldsymbol{h}_{c,N}^{(L)}\big]). \tag{94}$$

C.5. Motivation About the Rotation-base PE

The sequential planning method uses positional encoding to distinguish different agents starting from the same depot. In P&N Encoder, position encoding is first calculated in the partition part of the first layer of P&N Encoder, ignoring the formulation of the multi-head mechanism, $H_a^{(0)} = W_a x_d + b_a + PE$, the PE is involved as follows (calculating Eq. (11)):

$$\begin{aligned}\text{Attn}(H_a^{(0)}, X_c^{(1)}) =& \text{Softmax}\left(\frac{H_a^{(0)}W_Q(X_c^{(1)}W_K)^\intercal}{\sqrt{d}}\right)X_c^{(1)}W_V,\\ H_a^{(0)}W_Q(X_c^{(1)}W_K)^\intercal =& (W_a x_d + b_a + PE)W_Q(X_c^{(1)}W_K)^\intercal\\ =& (W_a x_d + b_a)W_Q(X_c^{(1)}W_K)^\intercal + PEW_Q(X_c^{(1)}W_K)^\intercal\\ =& \text{position-based score} + \text{angle-based score}.\end{aligned} \tag{95}$$

In the first layer, the attention calculation can be divided into a position-based score and an angle-based score. For the M agents, their position-based score remains the same, so the PE is used to highlight the nodes with right and predetermined angles in calculating the angle-based score (i.e., being responsible for the angle-based customer assignments). In the subsequent blocks, residual connections enable the PE to maintain the above functions.

Given the number of routes M, sinusoidal position encoding represents a fixed angle relationship so the angle-based partition will remain the same for any instance. However, the angle-related distribution of the routes in optimal solution varies with the depot location. As shown in visual cases Figure 11(a) and Figure 11(d), and Figure 7, if the location of the depot is in the bottom left corner (e.g. the right case in Figure 7 and the instance in Figure 11(d)), it only needs to consider customer partition in an approximate 90-degree space. However, in other cases as Figure 11(a) and the left case in Figure 7, all the directions are involved. Therefore, the sinusoidal PE meets challenges in generalizing to instances with different depot locations so it is necessary to introduce the depot coordinates to the calculation of PE. Experiments in Appendix E.6 demonstrate that SPE prefers instances with corner-location depot. The proposed Rotation-based positional encoding achieves better partition optimalities.

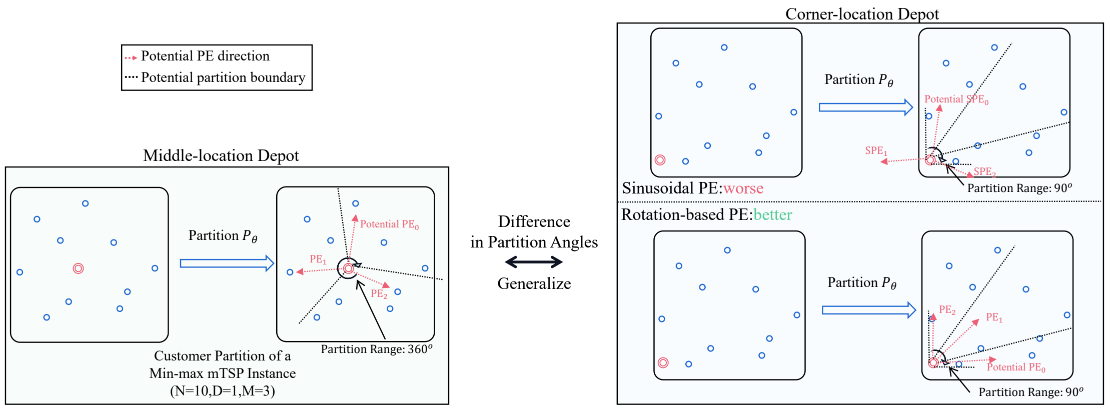


Figure 7. The PE required for the customer partition task varies to the depot locations. So the Rotation-based PE in DPN introduces the location of depots into calculation.

C.6. Rotation-base PE implementation

The main text provides a calculation method based on complex numbers for the Rotation-based PE in DPN. In practice, we adopt a real-number implementation provided in Su et al. (2023). Given $\boldsymbol{emb} = \boldsymbol{x}_d W_a + b_a, \boldsymbol{emb} \in \mathcal{R}^{M\times d}$ and $\theta_i = \frac{1}{1{,}000}^{\frac{i-1}{d}}$ for $i \in \{1, \dots, d/2\}, \theta_i \in \mathcal{R}^{d/2}$, the PE $\in \mathcal{R}^{M\times d}$ in position m PE$_m \in \mathcal{R}^d$ is as follows:

$$\mathrm{PE}_m = (\begin{pmatrix} emb_{m,1} \\ emb_{m,2} \\ emb_{m,3} \\ emb_{m,4} \\ \vdots \\ emb_{m,d-1} \\ emb_{m,d} \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2-1} \\ \cos m\theta_{d/2-1} \end{pmatrix} + \begin{pmatrix} -emb_{m,2} \\ emb_{m,1} \\ -emb_{m,4} \\ emb_{m,3} \\ \vdots \\ -emb_{m,d} \\ emb_{m,d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix})W_{PE}. \tag{96}$$

The RoPE in natural language processing usually uses 10,000 as the base (Su et al., 2023; 2024). However the number of agents in min-max VRPs usually does not exceed 100, so we adopt a smaller base of 1,000 for a shorter sequence. There is only a slight difference in the training stability between the two bases.

C.7. Discussion: Application on General VRPs

Different from general VRPs (e.g., CVRP), min-max VRPs limit the total number of vehicles and minimize the longest route. Solvers for min-max VRPs need to process these requirements with special procedures. DPN adopts the P&N Encoder and the APS-Loss to specifically process this constraint so the proposed DPN is unnecessary for VRPs without this constraint. Moreover, due to the existence of this constraint, together with min-max VRP emphasizing balancing the lengths between different routes, powerful heuristic approaches for min-sum problems are not well-generalized to the min-max case (Bertazzi et al., 2015), and existing constructive min-max VRP solving frameworks can not effectively generalize to general VRP (Son et al., 2024).

We acknowledge the enormous value of a universal and effective framework for both VRPs and min-max VRPs. Therefore, as future work, we are eager to adopt the idea of DPN to design a universal constructive solver to process decoupled representations between routes and effectively handle both VRPs and min-max VRPs.

D. Experiments Details

D.1. Hyperparameters

This paper trains 16 models on different scales and different settings. We will provide these models as pre-trained after being open-source, and their hyperparameters are listed in Table 5.

Table 5. Training settings of the proposed DPN on different scales.

Single-depot Training Settings		
Problem	mPDP50 & mTSP50	mPDP100 & mTSP100
Fintune or not	No	No
number of agents(M)	[2,10]	[2,10]
number of depots(D)	1	1
number of encoder layers (L)	6	6
learning rate	1.00E-04	1.00E-04
learning rate decay	1	1
batch size	256	256
epoches	500	500
epoch size	256000	256000
number of permutations (K)	60	60
Problem	mPDP200 & mTSP200	mPDP500 & mTSP500
Fintune or not	From 100	From 100
number of agents(M)	[10,20]	[30,50]
number of depots(D)	1	1
number of encoder layers (L)	6	6
learning rate	1.00E-05	1.00E-05
learning rate decay	1	1
batch size	64	16
epoches	20	20
epoch size	64000	16000
number of permutations (K)	60	60
Multi-depot Training Settings		
Problem	MDVRP50 & FMDVRP50	MDVRP100 & FMDVRP100
Fintune or not	No	No
number of agents(M)	[2,10]	[2,10]
number of depots(D)	[2,10]	[2,10]
number of encoder layers (L)	6	6
learning rate	1.00E-04	1.00E-04
learning rate decay	1	1
batch size	256	256
epoches	500	500
epoch size	256000	256000
number of permutations (K)	60	60
Problem	MDVRP50-F & FMDVRP50-F	MDVRP100-F & FMDVRP100-F
Fintune or not	From 50	From 100
number of agents(M)	[3,7]	[5,10]
number of depots(D)	8	8
number of encoder layers (L)	6	6
learning rate	1.00E-05	1.00E-05
learning rate decay	1	1
batch size	128	128
epoches	20	20
epoch size	128000	128000
number of permutations (K)	60	60

D.2. Datasets

There are a total of 10 100-instance datasets used in min-max mTSP and min-max mPDP in Table 1 and Table 2. We used the provided 100-instance test set in Son et al. (2024) for these datasets. Son et al. (2024) also provides the results of closed-source methods ScheduleNet and NCE with their provided datasets. So the Obj. of these two learning-based methods provided in Table 2 is accurate. We also generate a 100-instance validation size for the validation curve in Figure 3 and

Figure 8. The testing dataset for multi-depot min-max VRPs is also uniformly generated. In addition, Table 1 also shows the number of customers N and depots D corresponding to each dataset. The use of $N + D$ to represent scale in mTSP problems comes from previous works (Cao et al., 2021; Son et al., 2024; Mahmoudinazlou & Kwon, 2024).

D.3. Training Time

The detailed training time of DPN on four involved min-max VRPs are listed in Table 6. All 100-node training tasks can be done within 5 days.

Table 6. Training time of DPN on a single Nvidia Tesla V100S GPU.

Problem	min-max mTSP100	min-max mPDP100	min-max MDVRP100	min-max FMDVRP100
Epoch Time	11.0min	11.2min	12.6min	12.6min
Total Time	91.7h	93.3h	105h	105h

E. More Experiments

E.1. Generalization on Benchmark: Min-max mTSP SetI

We use the benchmark mTSP SetI with less than 500 nodes (i.e., 50- to 500-scale) to validate the generalization ability of DPN. The performances of heuristic algorithms and the best-known solution (BKS) are reported in LKH3 and HGA (Mahmoudinazlou & Kwon, 2024), We use a special fine-tuned versions on min-max mTSP200 for both the Equity-Transformer and DPN. They follow the same setting of the normal min-max mTSP200 fine-tuned model except for setting the range of agent number M to $[3, 20]$. As shown in Table 7, as neural solvers, the proposed DPN-$\times$8aug and DPN-$\times$8aug-$\times$16per versions narrow the optimal gap compared to the Equity-Transformer-$\times$8aug.

Table 7. Performances (objective function values and gaps to the BKS) of DPN on mTSP SetI datasets with less than 500 nodes.

		BKS	LKH-3	HGA	Equity-Transformer-$\times$8aug	DPN-$\times$8aug	DPN-$\times$8aug-$\times$16per	
Instances	M=	Obj.	Obj.	Obj.	Obj.	Obj.	Obj.	Gap
mtsp51	3	160	160	160	201	174	173	8.30%
	5	118	118	118	125	120	120	1.91%
	10	112	112	112	112	112	112	0.00%
mtsp100	3	8509	8509	8509	10411	9835	9759	14.69%
	5	6766	6766	6771	7929	7318	7279	7.59%
	10	6358	6358	6358	6454	6365	6359	0.00%
	20	6358	6358	6358	6358	6358	6358	0.00%
rand100	3	3032	3032	3032	3638	3196	3196	5.43%
	5	2410	2410	2410	2537	2477	2453	1.79%
	10	2299	2299	2299	2299	2299	2299	0.00%
	20	2299	2299	2299	2299	2299	2299	0.00%
mtsp150	3	13038	13038	13093	15678	14993	14929	14.50%
	5	8450	8450	8487	9993	9746	9695	14.73%
	10	5557	5557	5587	6404	6196	6196	11.50%
	20	5246	5246	5246	5263	5285	5248	0.03%
gtsp150	3	2402	2402	2402	2740	2551	2508	4.44%
	5	1741	1741	1741	1950	1823	1823	4.71%
	10	1555	1555	1555	1562	1556	1555	0.00%
	20	1555	1555	1555	1555	1555	1555	0.00%
kroA200	3	10691	10691	10700	13342	13004	12924	20.89%
	5	7412	7412	7420	8971	8755	8754	18.10%
	10	6223	6223	6223	6382	6243	6223	0.00%
	20	6223	6223	6223	6223	6223	6223	0.00%
lin318	3	15701	15701	15714	19514	16879	16879	7.50%
	5	11289	11289	11297	13763	12292	12265	8.65%
	10	9731	9731	9731	10129	9765	9738	0.07%
	20	9731	9731	9731	9731	9731	9731	0.00%
Gap to BKS		-	0.00%	0.07%	10.26%	5.67%	5.36%	

E.2. Generalization on Benchmark: Min-max mTSP Lib

We also adopt the widely used (Kim et al., 2022a) mTSP Lib (Reinelt, 1991) dataset for testing DPN. mTSP Lib contains four min-max mTSP instances (i.e., eil51, berlin52, eil76, and rat99). In testing both DPN and Equity-Transformer, We use both the model trained on 50-node and 100-node instances for validation and report the better result. As shown in Table 8, DPN variants also narrow the optimality gap on the mTSP Lib dataset.

Table 8. Performances (objective function values and gaps to the BKS) of DPN on mTSP Lib dataset.

		BKS	HGA	Equity-Transformer-×8aug	DPN-×8aug	DPN-×8aug-×16per	
Instance	M=	Obj.	Obj.	Obj.	Obj.	Obj.	Gap
eil51	2	223	223	236	234	227	2.00%
	3	160	160	164	167	165	3.09%
	5	118	118	120	120	120	1.66%
	7	112	112	112	112	112	0.00%
berlin52	2	4110	4110	4425	4421	4186	1.86%
	3	3074	3074	3263	3328	3113	1.27%
	5	2441	2441	2495	2663	2441	0.02%
	7	2441	2441	2441	2441	2441	0.00%
eil76	2	281	282	295	291	291	3.65%
	3	197	197	206	201	201	2.05%
	5	143	144	146	144	144	0.71%
	7	128	128	129	129	128	0.30%
rat99	2	666	668	795	754	754	13.16%
	3	518	518	566	572	572	10.55%
	5	450	450	474	467	467	3.71%
	7	437	437	440	442	442	1.02%
Gap to BKS		0.00%	0.06%	4.60%	2.87%	2.81%	

E.3. Generalization on Cross-distribution Datasets

The cross-distribution generalization ability of neural combinatorial optimization solvers is necessary (Jiang et al., 2022). In Table 1 and Table 2, DPN has shown effectiveness in min-max mTSP instances with a uniform distribution. In this subsection, we evaluate the DPN on min-max mTSP200 with the Gaussian distribution, the Rotation distribution, and the Explosion distribution provided in (Zhou et al., 2023). We use the provided dataset in Zhou et al. (2023), and the result is shown in Table 9. DPN demonstrates outstanding robustness on different distributions.

Table 9. The average objective function values(i.e., Obj.), gaps to the best algorithm (i.e., Gap) on 3 datasets with different distributions. HGA makes one run for these results, and the best result is in bold.

Min-max mTSP200-Gaussian						
M=	10		15		20	
Methods	Obj.	Gap.	Obj.	Gap.	Obj.	Gap.
HGA	**1.5063**	-	1.5031	0.00%	1.5031	0.00%
Equity-Transformer-F-×8aug	1.6426	9.05%	1.5458	2.84%	1.5265	1.56%
DPN-F-×8aug	1.5204	0.93%	1.5032	0.00%	1.5031	0.00%
DPN-F-×8aug-×16per	1.5181	0.78%	**1.5031**	-	**1.5031**	-
Min-max mTSP200-Explosion						
M=	10		15		20	
Methods	Obj.	Gap.	Obj.	Gap.	Obj.	Gap.
HGA	**1.7651**	-	**1.7488**	-	**1.7486**	-
Equity-Transformer-F-×8aug	1.9269	9.17%	1.8277	4.51%	1.8062	3.30%
DPN-F-×8aug	1.7924	1.55%	1.7668	1.03%	1.7663	1.01%
DPN-F-×8aug-×16per	1.7894	1.37%	1.7667	1.02%	1.7663	1.01%
Min-max mTSP200-Rotation						
M=	10		15		20	
Methods	Obj.	Gap.	Obj.	Gap.	Obj.	Gap.
HGA	1.7492	2.79%	1.7393	4.23%	1.7391	4.28%
Equity-Transformer-F-×8aug	1.8871	10.90%	1.7524	5.01%	1.7151	2.84%
DPN-F-×8aug	1.7064	0.28%	1.6697	0.05%	1.6678	0.00%
DPN-F-×8aug-×16per	**1.7017**	-	**1.6688**	-	**1.6677**	-

E.4. Extending to N=100,000

The capability of extending to very large-scale instances (Luo et al., 2024) is also necessary for neural min-max VRP solvers. The original P&N Encoder of DPN cannot manage large-scale problems such as $N = 100,000$. When generating a single solution, the space complexity of DPN is $\mathcal{O}(N(N+M))$, so the proposed DPN can only solve up to 10,000-scale problems on a single Tesla V-100S GPU.

DPN-w/o Navigation Part for N=100,000. To handle large-scale data such as N=100,000, we further propose a variant of DPN. We design a variant of DPN by removing the navigation part in each P&N Encoder layer (only preserving the partition part, marked as DPN-w/o Navigation Part in Table 10) and train this variant from scratch on 100-node min-max mTSP (i.e, $N+D$=100). The space complexity of the DPN-w/o Navigation Part reduces to $\mathcal{O}(NM)$ so this model can derive feasible solutions on 100,000 scale problems. Results in Table below 10 show that this variant can effectively solve very large min-max mTSP(N=100,000) on a single GPU with considerable time.

All the heuristic baselines (like HGA and LKH) with default settings cannot generate results within 5 days on 100,000-scale problems. However, the performance of solutions can be partially reflected by the comparison to DPN and Equity-Transformer on min-max mTSP10,000 shown in Table 10 where this variant of DPN demonstrates better scaling performances.

Table 10. The average objective function values(i.e., Obj.) and the total time (i.e., Time) of DPN and DPN-w/o navigation part on min-max mTSP 10,000 and very large-scale instances.

	min-max mTSP10,000(N=9,999,D=1,10 instances)					
M=	500		750		1,000	
Method	Cost	Time	Cost	Time	Cost	Time
Equity-Transformer	4.5645	3m	2.9245	3m	2.8524	3m
DPN	2.4640	2.8m	2.4724	2.8m	2.3853	2.8m
DPN-w/o Navigation Part	2.3487	2.8m	2.3333	2.8m	2.2568	2.8m

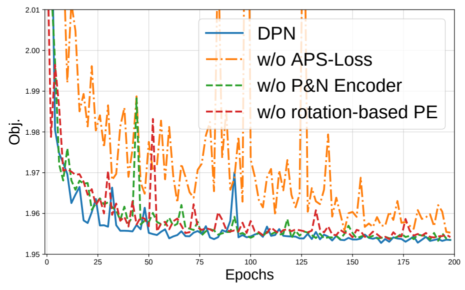
	min-max mTSP(10 instances)					
	$N+D$=50,000		$N+D$=70,000		$N+D$=100,000	
M=	500		500		500	
Method	Cost	Time	Cost	Time	Cost	Time
DPN-w/o Navigation Part	3.4183	34m	3.7832	72m	4.2078	2h

E.5. Ablation on M=10

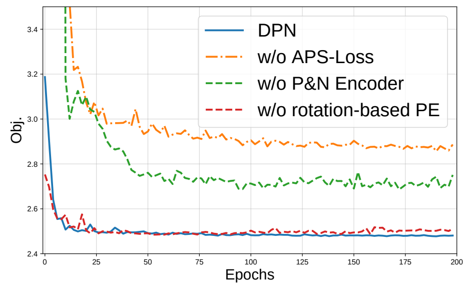
The ablation studies in Section 4.2 focus on the setting of $M = 5$, and the ablation studies on $M = 10$ are provided in this part. As shown in Table 11, the efficiency of the proposed components in DPN is even more significant in problems with $M = 10$. Due to the decision space of navigation reducing together with the average length of routes, min-max VRPs with $M = 10$ agents might focus more on partition tasks rather than navigation tasks, so the use of Rotation-based PE demonstrates more significant superiority. The training curve in Figure 8 also confirms the conclusion. The data points in both Figure 3 and Figure 8 are derived from evaluating the same validation set. These ablation models are equipped with both the $\times$8aug and $\times$16per augmentations.

Table 11. Ablation study on proposed components of DPN.

	min-max mTSP100	min-max mPDP100
M=	10	10
w/o P&N Encoder	1.9537	3.2706
w/o APS-Loss	1.9571	2.8078
w/o Rotation-based PE	1.9544	2.7123
DPN-$\times$8aug-$\times$16per	**1.9532**	**2.6949**



(a) min-max mTSP



(b) min-max mPDP

Figure 8. Training curves for ablation study (M=10).

E.6. Ablation on PE

In Appendix C.5, we have shown that the Rotation-based PE implemented in DPN helps improve convergence and optimality on both M=5 and M=10. To further underscore the advantages of using Rotation-based PE, this section presents additional comparative experiments that substantiate the efficacy of Rotation-based PE in facilitating the learning of partition strategies, especially when considering various depot locations (outlined in Section C.5). These strategies are crucial for adapting to different spatial distributions of depots, thereby underlining the practical value of Rotation-based PE in adaptive partitioning contexts.

In this subsection, we generate 10,000 mTSP100 instances, the depot coordinate of the i-th instance is as follows:

$$\mathbf{x}_d = \big(\, 0.005 + 0.01(\lfloor \frac{i-1}{100} \rfloor)\ ,\ 0.005 + 0.01(i \mod 100)\,\big), \tag{97}$$

where $(i \mod 100)$ represents the remainder.

We run the model of DPN-×8aug-×16per, DPN-w/o-Rotation-based-×8aug-×16per, and the heuristic method HGA on these instances with M=10. HGA makes one run in these instances. Figure 9 consists of 100 blocks and each block contains the average gap of the 100 instances whose depot is located in the block. The first two plots on the left in Figure 9 demonstrate that for both the DPN (left 1) and the ablation model of DPN with SPE (without Rotation-based PE) (left 2), the gap to HGA relates to depot locations and the instances with middle-location depots perform relatively worse. Relatively, Rotation-based PE enables DPN to handle instances at any depot location with an even gap level and demonstrates the contribution of Rotation-based PE to learning a more robust customer partition strategy.

The right plot shows that compared to Rotation-based PE, the original sinusoidal PE performs well in the instances with corner-location depots, but cannot generalize to the instances with middle-location depots. With the addition of the depot coordinate information, the Rotation-based PE improves the performance of these instances with middle-location depots.

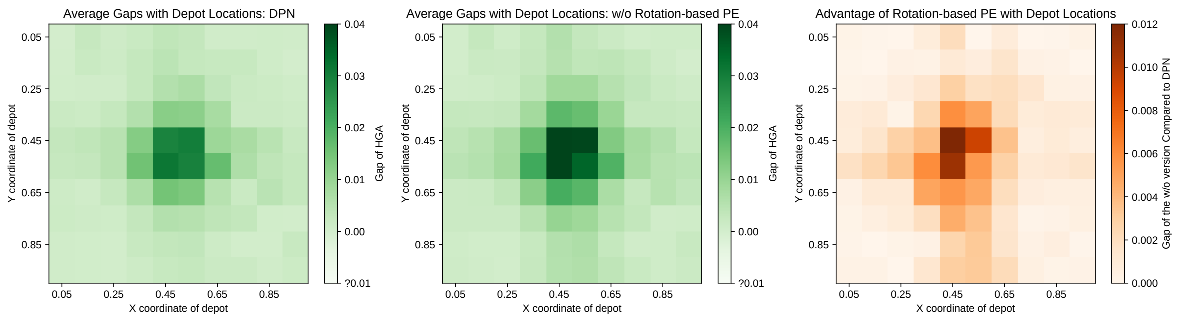
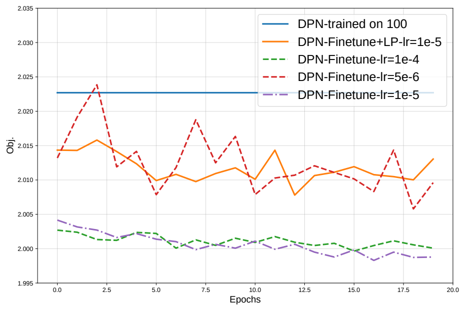


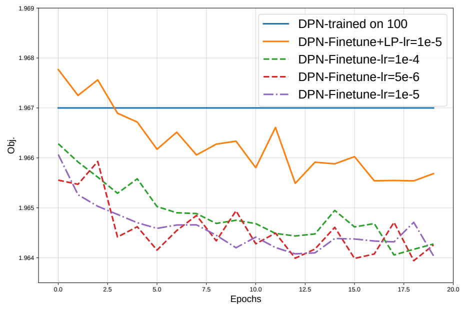
Figure 9. Plot of relations between depot locations and performances. These plots divide the xy-plate into 100 uniform blocks, and each block represents the average Gap of the instances whose depots are located in the block. The two green heatmaps on the left represent the relation between depot locations and the gaps to HGA (Mahmoudinazlou & Kwon, 2024) when testing our DPN and the w/o Rotation-based PE ablation model respectively. The red heatmap on the right is the ratio of the objective function between the two versions of models (i.e., Gap of w/o Rotation-based PE to DPN).

E.7. Ablation on the Fine-tuning Phase & Zero-shot Generalization

DPN uses a learning rate of 1e-5 in fine-tuning, and Figure 10 provides ablation studies to demonstrate the superiority of this setting. The DPN-Finetune-lr=1e-5 version represents fine-tuning with the learning rate being 1e-5 (i.e., the model for min-max mTSP in Table 2) and the DPN-Finetune-LP-lr=1e-5 is the fine-tuning setting adopted in Equity-Transformer (Son et al., 2024) which adds a 2-layer MLP for the contextual embedding $\boldsymbol{q}_{contex}$. Compared to models with other learning rates, the DPN-Finetune-lr=1e-5 consistently shows the best convergence speed among all versions.



(a) min-max mTSP200 (N=199,M=10)



(b) min-max mTSP200 (N=199,M=15)

Figure 10. Training curves on min-max mTSP200 with different learning rates and settings.

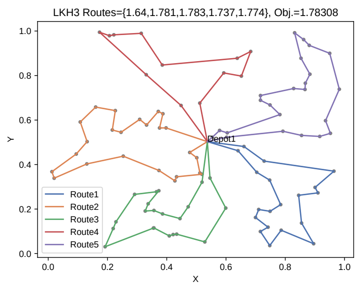
Table 12 tests the ablation fine-tuned models with different settings. It proves the above conclusions as well, and in addition, columns of "DPN-100" with ×8aug or ×16per list the results of DPN trained on 100-scale min-max mTSP and tested on 200-scale min-max mTSP datasets which can demonstrate that our DPN exhibits advantages in zero-shot generalization compared to Equity-Transformer.

Table 12. Ablation study on proposed components in DPN.

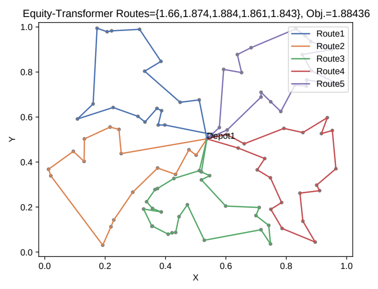
Min-max mTSP200			
M=	10	15	20
Methods	Obj.	Obj.	Obj.
HGA	1.9861	**1.9628**	**1.9627**
LKH3	**1.9817**	1.9628	1.9628
OR-Tools(600s)	2.3711	2.3687	2.3687
DAN	2.3586	2.1732	2.1151
ScheduleNet*	2.35	2.13	2.07
NCE*	2.07	1.97	1.96
Equity-Transformer-×8aug	2.0750	1.9947	1.9658
Equity-Transformer-F-×8aug	2.0500	1.9688	1.9631
DPN-100-×8aug	2.0381	1.9692	1.9660
DPN-100-×8aug-×16per	2.0227	1.9670	1.9648
DPN-F(lr=1e-4)-×8aug-×16per	2.0107	1.9643	1.9628
DPN-F(lr=5e-6)-×8aug-×16per	2.0004	1.9644	1.9628
DPN-F(lr=1e-5+LP)-×8aug-×16per	2.1064	1.9673	1.9645
DPN-F(lr=1e-5)-×8aug-×16per	1.9993	1.9640	1.9628

F. Visualization

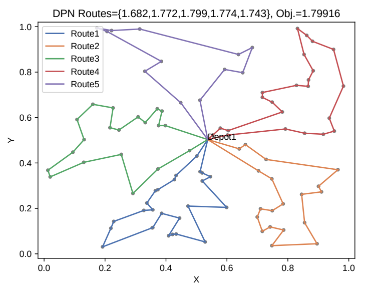
This section provides visualization of some random instances of min-max mTSP (Figure 11, Figure 12), min-max mPDP (Figure 13, Figure 14), and min-max MDVRP (Figure 15, Figure 16, Figure 19), min-max FMDVRP (Figure 17, Figure 18, Figure 20) to display the advantage of the proposed DPN.



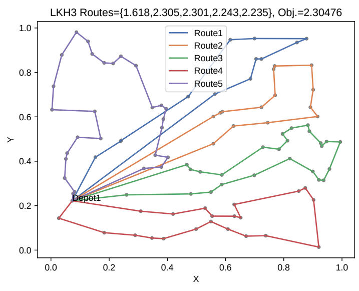
(a) LKH3



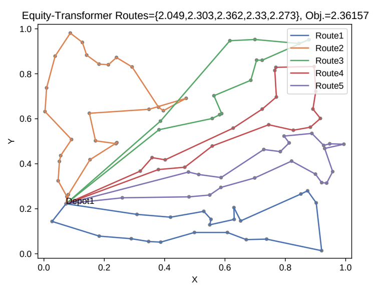
(b) Equity-Transformer-×8aug



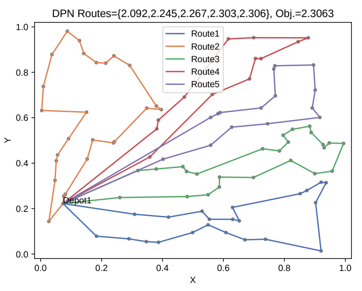
(c) DPN-×8aug



(d) LKH3

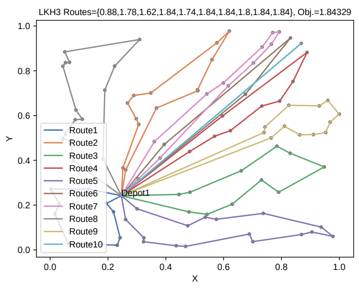


(e) Equity-Transformer-×8aug

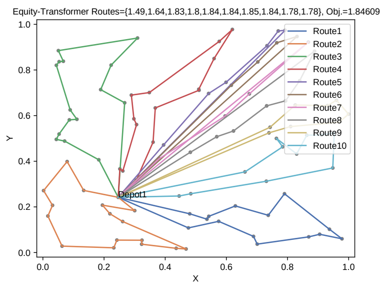


(f) DPN-×8aug

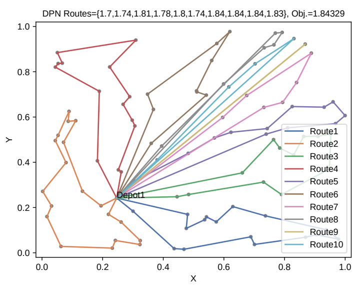
Figure 11. Min-max mTSP instances (M=5), solving by LKH3, Equity-Transformer, and ours DPN.



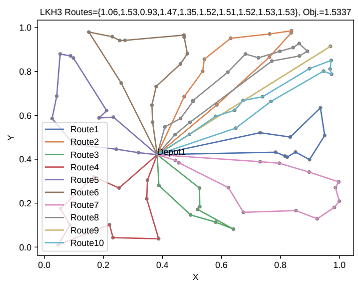
(a) LKH3



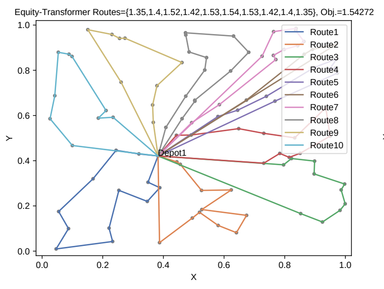
(b) Equity-Transformer-×8aug



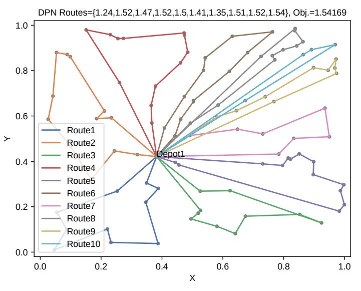
(c) DPN-×8aug



(d) LKH3

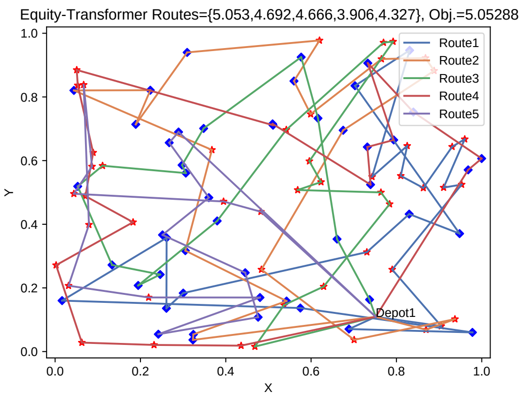


(e) Equity-Transformer-×8aug

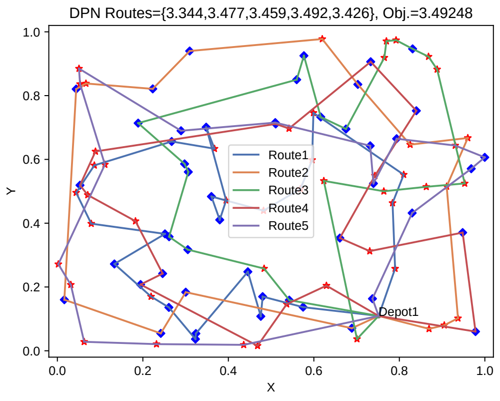


(f) DPN-×8aug

Figure 12. Min-max mTSP instances (M=10), solving by LKH3, Equity-Transformer, and ours DPN.

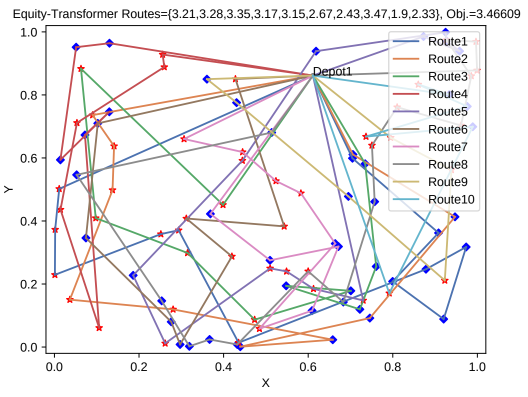


(a) Equity-Transformer-×8aug

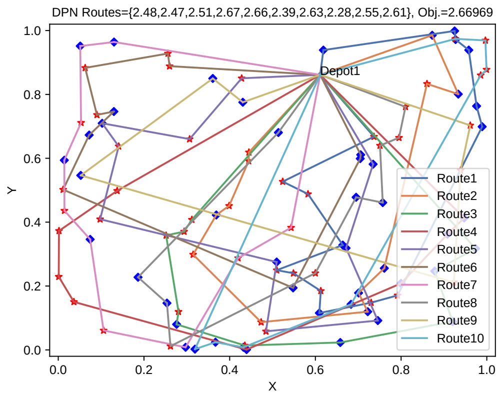


(b) DPN-×8aug

Figure 13. Min-max mPDP instances (M=5), solving by Equity-Transformer and ours DPN. Different colors and shapes (blue diamonds for pickups and red stars for deliveries) distinguish customers.

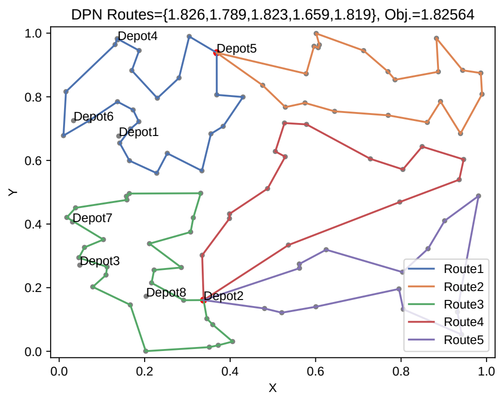


(a) Equity-Transformer-×8aug

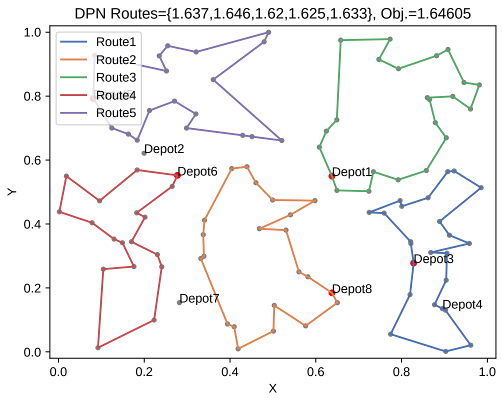


(b) DPN-×8aug

Figure 14. Min-max mPDP instances (M=10), solving by Equity-Transformer and ours DPN. Different colors and shapes (blue diamonds for pickups and red stars for deliveries) distinguish customers.

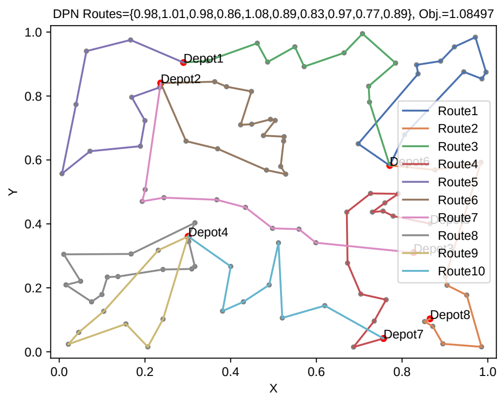


(a) DPN-×8aug-×16per

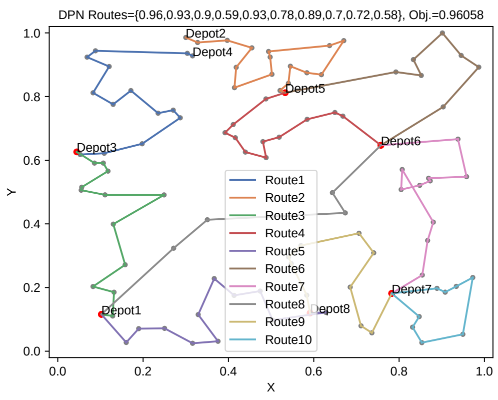


(b) DPN-×8aug-×16per

Figure 15. Min-max MDVRP instances (D=8,M=5) solving by ours DPN. Selected depots are highlighted by red.

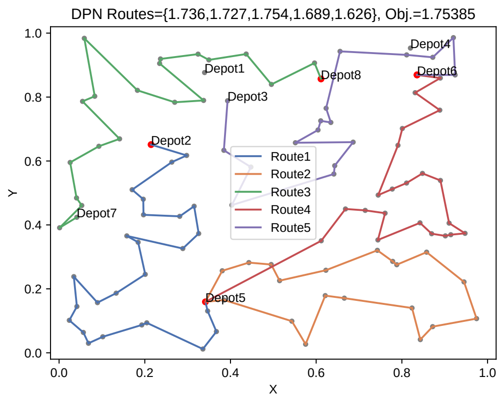


(a) DPN-×8aug-×16per

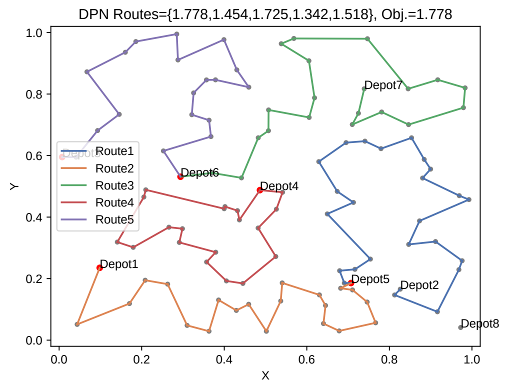


(b) DPN-×8aug-×16per

Figure 16. Min-max FMDVRP instances (D=8,M=10) solving by ours DPN. Selected depots are highlighted by red.

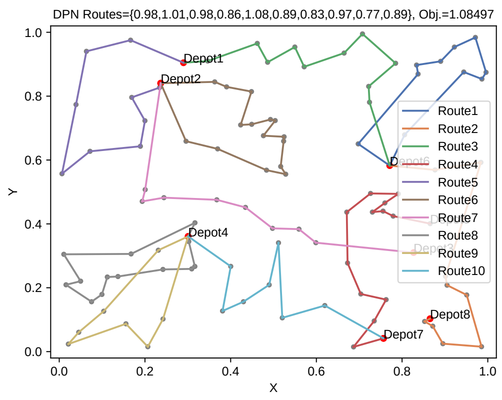


(a) DPN-×8aug-×16per

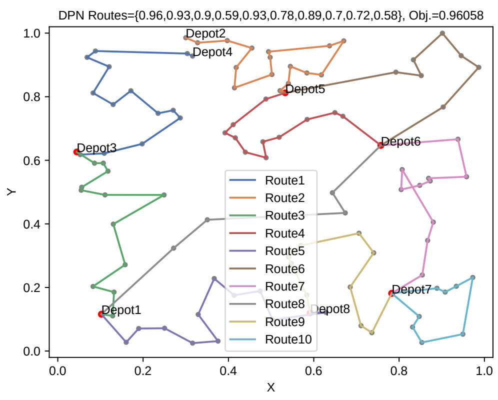


(b) DPN-×8aug-×16per

Figure 17. Min-max FMDVRP instances (D=8,M=5) solving by ours DPN. Selected depots are highlighted by red.

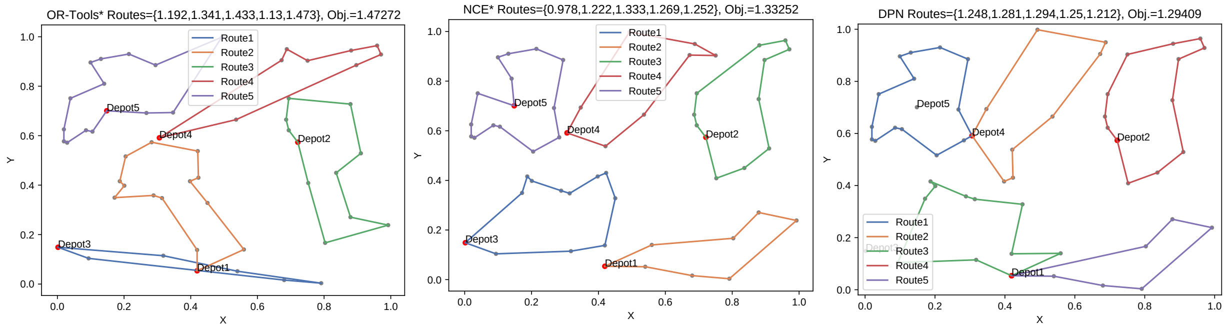


(a) DPN-×8aug-×16per



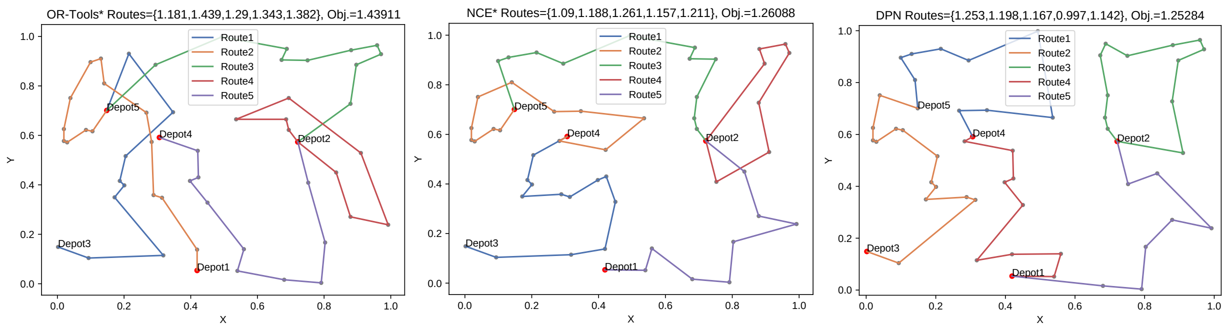
(b) DPN-×8aug-×16per

Figure 18. Min-max FMDVRP instances (D=8,M=10) solving by ours DPN. Selected depots are highlighted by red.



(a) OR-Tools* (b) NCE* (c) DPN-×8aug-×16per

Figure 19. Min-max MDVRP instances (M=5, D=5, N=50), solving by OR-Tools, NCE, and ours DPN. The solution of OR-Tools and NCE is reported in Kim et al. (2022a).



(a) OR-Tools* (b) NCE* (c) DPN-×8aug-×16per

Figure 20. Min-max FMDVRP instances (M=5, D=5, N=50), solving by OR-Tools, NCE, and ours DPN. The solution of OR-Tools and NCE is reported in Kim et al. (2022a).

The solutions of OR-Tools and NCE in Figure 19 and Figure 20 are reported in Kim et al. (2022a). Due to their code being unavailable, we crawl the data and optimal solution through pixel coordinates. Therefore, the value of their objective function may be inconsistent. In these instances, DPN still outperforms other neural solvers.

G. Baseline & License

The implement HGA algorithm is given in `https://github.com/Sasanm88/m-TSP`. For all uniform datasets with 100 instances, we set HGA to run 10 times. For the dataset with uniform depot location (adopted in Appendix E.6) consisting of 10,000 instances and cross-distribution experiments (in Appendix E.3), we only make HGA run once.

We implement the approximate min-max constraint and the pickup and delivery constraints for OR-Tools based on official tutorials `https://or-tools.github.io/docs/pdoc/ortools.html`. In solving min-max mTSP and min-max mPDP, we enable them to run 600s. For multi-depot min-max VRPs, due to various implementation alternations available, we use the results reported in Kim et al. (2022a).

As to LKH3, We adopt the commonly used setting in the NCO methods (Kool et al., 2018) and implement the LKH3 in min-max mTSP based on the code from `https://github.com/wouterkool/attention-learn-to-route`. We do not set runtime limits, but instead, specify its maximal according to commonly used settings and set the MAX_TRAILS parameter to 10,000, MAX_CANDIDATES to 6, and RUNS to 10. So, the solution optimality of LKH3 in this paper may be different from the report results in Son et al. (2024).

As mentioned in Appendix D.2 in the supplementary File, in testing mTSP, we use the dataset provided by Son et al. (2024). Due to the learning methods ScheduleNet and NCE in mTSP are provided in Son et al. (2024) with 2-digital results. So although they are not available, the results are accurate. However, with testing on a different dataset, the baseline results of min-max MDVRP and min-max FMDVRP in Table 4 are not accurate.

For Equity-Transformer, we use the provided pre-trained model for experiments on 50-scale min-max VRPs and 200-scale min-max VRPs. We further train 100-scale models for the min-max mTSP100 and the min-max mPDP100 datasets for a fair comparison. Moreover, we train a min-max mPDP500 fine-tuned model based on the min-max mPDP100 model. In Table 2, we provide an "Equity-Transformer-F-sample*" version, which is the reported result in Son et al. (2024) without illustrations about specific sample settings.

We use the provided pre-trained model for every run with a greedy decoding setting for another open-source neural solver DAN (Cao et al., 2021).

Table 13. A summary of licenses.

Resources	Type	License
HGA	Code	Available online
OR-Tools	Code	Apache License, Version 2.0
LKH3	Code	Available for academic research use
DAN	Code	MIT License
Equity-Transformer	Code	Available online
mTSP SetI	Dataset	Available for any non-commercial use
mTSP Lib	Dataset	Available for any non-commercial use

The licenses for codes and datasets used in this work are listed in Table 13.