2D-Ptr: 2D Array Pointer Network for Solving the Heterogeneous Capacitated Vehicle Routing Problem

Qidong Liu
Zhengzhou University
National Supercomputing Center in Zhengzhou
Zhengzhou, China
ieqdliu@zzu.edu.cn

Chaoyue Liu*
Zhengzhou University
Zhengzhou, China
chaoyueliu@gs.zzu.edu.cn

Shaoyao Niu
Zhengzhou University
Zhengzhou, China
ieniusy@gs.zzu.edu.cn

Cheng Long
Nanyang Technological University
Singapore, Singapore
c.long@ntu.edu.sg

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

Mingliang Xu*
Zhengzhou University
National Supercomputing Center in Zhengzhou
Zhengzhou, China
iexumingliang@zzu.edu.cn

ABSTRACT

The heterogeneous capacitated vehicle routing problem (HCVRP) aims to optimize the routes of heterogeneous vehicles with capacity constraints to serve a set of customers with demands. Existing learning-based methods for solving HCVRP have the problem of weak generalization ability, which means that well-trained model cannot adapt well to new scenarios with different vehicle or customer numbers. To address this issue, by modeling the simultaneous decision-making of multiple agents as a sequence of consecutive actions in real time, we propose a pointer network extension model, which includes a static encoder and a dynamic encoder to map the current situation to node embeddings and vehicle embeddings, respectively. For each element in the consecutive actions sequence, the decoder of our model uses the probability distribution obtained from node embeddings and vehicle embeddings as a 2D array pointer to select a tuple from the combinations of vehicles and nodes (customers and depot). We call this architecture a 2D Array Pointer network (2D-Ptr). Instead of planning paths based on the priority order of vehicles, 2D-Ptr plans paths based on the priority order of actions. In addition, 2D-Ptr consists of a series of carefully designed attention modules, entitling the model to be generalizable in the scenarios where additional vehicles (or customers) are introduced or existing vehicles (or customers) are removed. We empirically test 2D-Ptr and show its capability for producing near-optimal solutions through cooperative actions. 2D-Ptr delivers competitive performance against the state-of-the-art baselines, and can solve arbitrary instances of the HCVRP without requiring re-training.

*Corresponding authors.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

KEYWORDS

Attention Mechanism; Combinatorial Optimization; Heterogeneous CVRP; Pointer Network; Reinforcement Learning

ACM Reference Format:
Qidong Liu, Chaoyue Liu, Shaoyao Niu, Cheng Long, Jie Zhang, and Mingliang Xu. 2024. 2D-Ptr: 2D Array Pointer Network for Solving the Heterogeneous Capacitated Vehicle Routing Problem. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

The **H**eterogeneous **C**apacitated **V**ehicle **R**outing **P**roblem (HCVRP) is a classical combinatorial optimization problem, which aims to optimize the routes for a fleet of vehicles with different capacity (or speed) constraints to serve a set of customers with demands [7, 16]. To solve this problem, one option is to use Branch and Bound algorithms to explore the entire search space of possible solutions and provide the optimal solution [1, 8]. However, for NP-hard problems like HCVRP, exact solutions can only be computed in a reasonable amount of time for small input sizes. Therefore, some heuristic methods, such as simulated annealing [28] and genetic algorithm [18], have been proposed to generate approximate solutions through manually customized heuristic rules to balance the solution quality and calculation speed.

Recently, deep reinforcement learning (DRL) has shown its success in tackling complex combinatorial optimization problems [15, 16, 25, 27]. More specifically, the HCVRP can be modeled as a Markov decision process, and an action $a^t = (v_m, u_n)$ is selected at each step t, which means that vehicle v_m is selected to visit node u_n. This process is repeated until the task is completed. The advantage of these DRL methods is that they can automatically learn general patterns from the data, without the need for experts to manually formulate heuristic rules, and can often achieve similar solution quality and higher running speed compared with heuristic algorithms.

However, most of existing DRL works mainly focus on solving the combinatorial optimization problem of single vehicle or homogeneous fleet. Although there are a few DRL algorithms that can handle HCVRPs [16, 19, 21], they have issues with validity and generalizability due to the following reasons: 1) For each time step, each action a^t consists of two subactions that are executed sequentially: selecting a vehicle v_m from the fleet in a predetermined order [21] or using a vehicle selection decoder [16], and then selecting a node u_n for v_m through a node selection decoder. This two-step action selection method planing paths based on the priority order of vehicles can easily lead to suboptimal solutions. 2) Existing works usually assume that the fleet is fixed and unchanging, so a well-trained model for one fleet may generalize well to problems with different customer sizes, but cannot solve problems with another fleet without being re-optimized from scratch. This makes it difficult to deploy the model in the real-world where fleets may update or change.

To address the above issues, we propose a novel policy network called 2D-Ptr that consists of three components: node encoder, vehicle encoder and decoder. The node encoder is executed only once at the first step, which converts the raw features of nodes to node embeddings. The vehicle encoder is updated for each step, which embeds the current situation of vehicles into vehicle embeddings. Unlike the literature [16], which plans paths based on the priority order of vehicles, the decoder of our model plans paths based on the priority order of actions. More specifically, with the node embeddings $\mathbf{U}$ and vehicle embeddings $\mathbf{V}^t$ at each step t, we have a probability matrix P^t, where the value $P^t_{m,n}$ is positively related with the dot product of $\mathbf{U}_n$ and $\mathbf{V}^t_m$. Then the decoder selects a vehicle as well as its visited node based on the matrix P^t at each route construction step. This process is repeated until all customers are served.

The 2D-Ptr based on the priority order of actions planning paths can simultaneously solve vehicle selection and node selection problems within a single action, enabling it to search in a more rational and broader action space. In addition, we carefully design a series of attention modules in 2D-Ptr, entitling the model to be generalizable in the scenarios where additional vehicles (or customers) are introduced or existing vehicles (or customers) are removed. To our knowledge, this is the first attempt to address both node generalization and fleet generalization of HCVRP, while previous works only involve the former. Experimental results on 12 generated datasets indicate that 2D-Ptr surpasses existing state-of-the-art techniques concerning the trade-off between cost and runtime, positioning it as a valuable tool for various delivery service industries aiming to enhance efficiency and reduce costs.

2 RELATED WORK

In this section, we mainly review the DRL-based methods for solving VRPs with a single vehicle, a homogeneous fleet and a heterogeneous fleet, respectively.

2.1 Routing a single vehicle or homogeneous fleet

The Pointer Network [22] is the first work using deep model to solve VRPs. Later, Bello et al. [2] extend it to DRL, training parameters without relying on ground-truth labels. Inspired by Transformer, some Attention-based architectures [15, 25, 27] are designed to replace LSTM-based encoder and decoder, working in parallel and achieving better performance. Different from the above works that directly construct solutions, some others combine heuristic algorithms with DRL, optimizing an initial solution iteratively, such as guiding the search process of LKH according to the output of a trainable sparse graph network [26], or using a neural network to select node pairs for the 2-opt operator [24]. In addition, there are also several works [6, 12] focusing on improving the generalization of a pre-trained model.

Later, some works focus on the VRPs with a homogeneous fleet, such as multiple traveling salesman problem (MTSP) and multiple pick-up and delivery problem (MPDP). SplitNet [17] generates a single TSP solution sequence, and then splits it into MTSP sequences by an attention-based model. Equity-Transformer (ET) sets virtual depot for each vehicle to split the path, then sequentially generates actions considering workload balancing in min-max MTSP and MPDP [20]. Jonas et al. [5] propose a model called JAMPR to solve the CVRP with time window by constructing multiple tours. In addition, some scholars tend to use multi-agent reinforcement learning for collaborative decision-making of the fleet. Zhang et al. [29] let the agents share the same policy network, and then make decisions in turn. On the contrary, in MAPDP [30], agents have independent networks, making decisions together through a fleet handler resolving possible node selection conflicts.

2.2 Routing a heterogeneous fleet

There are also a few existing DRL-based works focusing on HCVRP. For example, Vera and Abad [21] train a policy network for node selection by A2C method, and then apply it to make decisions for each vehicle alternately, which is one of the pioneers to solve the HCVRP by DRL. Qin et al. [19] adopt reinforcement learning to choose from several meta-heuristics with different characteristics to improve the performance of the heuristic framework. Moreover, Li et al. [16] consider vehicle selection to be as important as node selection and propose an effective framework with two selection decoder, selecting vehicle and node in two steps. However, these works seldom involve the discussion of fleet generalization, as they typically assume that the fleet is fixed and the input to the architecture is constrained by the number of vehicles, which means that well-trained models cannot adapt to new scenarios where additional vehicles are introduced or existing vehicles are removed.

3 METHODOLOGY

3.1 Problem Description

Given an HCVRP instance $S(\mathcal{V}, \mathcal{U})$, where $\mathcal{V} = \{v_m\}_{m=1}^{M}$ represents a heterogeneous fleet with M vehicles, and $\mathcal{U} = \{u_n\}_{n=0}^{N}$ represents a node set consisting of a depot (u_0) and N customers, the HCVRP describes a process that all fully loaded vehicles start from the depot, and sequentially visit the customers to satisfy their demands, with the constraints that each customer can be visited exactly once, and the loading amount for a vehicle during a single trip can never exceed its capacity. Each vehicle $v_m \in \mathbb{R}^2$ has two attributes: capacity (ρ_m) and speed (χ_m), and each node $u_n \in \mathbb{R}^3$ has

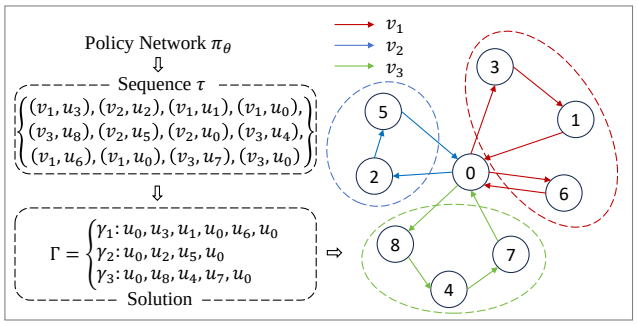


Figure 1: Constructing solution Γ to problem S from the sequence τ generated by the policy network π_θ

three attributes: x-coordinate (x_n), y-coordinate (y_n) and demand (q_n). Note that the depot's demand (q_0) is set to 0.

Let $\Gamma = \{\gamma_m\}_{m=1}^{M}$ represent a solution to the problem instance S, where γ_m denotes the construction route for each vehicle $v_m \in \mathcal{V}$. Our method aims to approach the optimal solution Γ^* that minimizes the maximum travel time of all vehicles, also known as the min-max objectives [3, 14, 23]. Let $dist(\cdot,\cdot)$ denotes the distance between two sequential nodes, we have

$$\Gamma^* = \arg\min_{\Gamma} \max_{\gamma_m \in \Gamma} \left(\sum_{j=2}^{|\gamma_m|} \frac{dist(\gamma_m^{j-1}, \gamma_m^{j})}{\chi_m} \right), \tag{1}$$

where $|\gamma_m|$ and $\gamma_m^j \in \mathcal{U}$ represent the length of γ_m and the j-th term of γ_m, respectively. It is worth noting that due to vehicle capacity limitations, each vehicle can return to the depot multiple times for replenishment, and the last station must be at the depot.

3.2 Modeling HCVRP via DRL

The solution Γ can be inferred from the sequence of tuples $\tau = \{(v^t, u^t), t = 0, 1, \cdots, T-1\}$, where each tuple represents the vehicle v^t visiting the node u^t at step t, and T is the length of the sequence, as shown in Fig.1. We are interested in finding a stochastic policy π_θ, which generates the sequence τ by minimizing the maximum travel time of all vehicles while satisfying problem constraints. Here, θ denotes the trainable parameters. To train π_θ in an unsupervised way, we adopt the REINFORCE with Greedy Rollout Baseline [15] method to minimize the objective function shown in Eq.1, where the state space $\mathcal{S}$, the action space $\mathcal{A}$, the state transition function f, and the reward $\mathcal{R}$ are defined as follows:

- **State space $\mathcal{S}$.** For each step t, we take $\mathbf{s}^t = (\mathbf{v}^t, \mathbf{u}^t)$ as the current state, where $\mathbf{v}_m^t = (\chi_m, \rho_m, \hat{\rho}_m^t, \delta_m^t, \ell_m^t)$ and $\mathbf{u}_n^t = (x_n, y_n, q_n^t)$ represent the state of vehicle v_m and node u_n, respectively. The elements of $\mathbf{s}^t$ can be classified into two categories, among which the elements that change with step t based on the state transition function f are called dynamic elements, while the elements that keep fixed throughout the decision-making process are called static elements. For a detailed introduction to these elements, please see Tab.1.
- **Action space $\mathcal{A}$.** The action a^t is defined as selecting a tuple (v_m, u_n) from all combinations of vehicles and nodes, indicating that vehicle v_m is selected to visit node u_n at step t. Instead of dividing the above operations into two steps: vehicle selection and node selection, such as in literature [16], our method can solve them simultaneously within one action.
- **State transition function f.** Given the current state $\mathbf{s}^t$ and action $a^t = (v_i, u_j)$, the next state $\mathbf{s}^{t+1}$ can be determined, i.e., $\mathbf{s}^{t+1} = f(\mathbf{s}^t, a^t)$. More specifically, the elements of $\mathbf{s}^t$ are updated according to the rules described in Tab.2.

Table 1: The detailed description to the elements of $\mathbf{s}^t$

Notation	Category	Description
χ_m	Static	Speed of v_m
ρ_m	Static	Capacity of v_m
$\hat{\rho}_m^t$	Dynamic	Accumulated usage capacity of v_m at step t
δ_m^t	Dynamic	Accumulated travel time of v_m at step t
ℓ_m^t	Dynamic	The last visited node of v_m
x_n	Static	X-coordinate of u_n
y_n	Static	Y-coordinate of u_n
q_n^t	Dynamic	Demand of u_n at step t

Table 2: Transition Rules

$\mathbf{s}^{t+1}$	$f(\mathbf{s}^t, a^t)$	Condition
$\hat{\rho}_m^{t+1}$	0	$m = i \wedge j = 0$
	$\hat{\rho}_m^t + q_j$	$m = i \wedge j \neq 0$
	$\hat{\rho}_m^t$	otherwise
δ_m^{t+1}	$\delta_m^t + dist(u_j, \ell_m^t)/\chi_m$	$m = i$
	δ_m^t	otherwise
ℓ_m^{t+1}	u_j	$m = i$
	ℓ_m^t	otherwise
q_n^{t+1}	0	$n = j$
	q_n^t	otherwise

- **Reward $\mathcal{R}$.** The reward is defined as the negative value of the maximum travel time of all vehicles, i.e.,

$$\mathcal{R} = -\max_{m \in \{1,\cdots,M\}} \delta_m^T,$$

where δ_m^T denotes the accumulated travel time of vehicle v_m at final step T, and M denotes the number of vehicles.

3.3 2D Array Pointer Network

In this section, we elaborate on the structure of the stochastic policy network π_θ, which consists of three components: node encoder, vehicle encoder and decoder, as shown in Fig.2. Below, we describe these three components separately.

3.3.1 Node Encoder. The function of the node encoder is to convert the raw features of nodes into high-dimensional embeddings $\mathbf{U} \in \mathbb{R}^{(N+1)\times d}$. Let $\mathbf{u}_n^0 = (x_n, y_n, q_n^0)$ denote the initial features of node

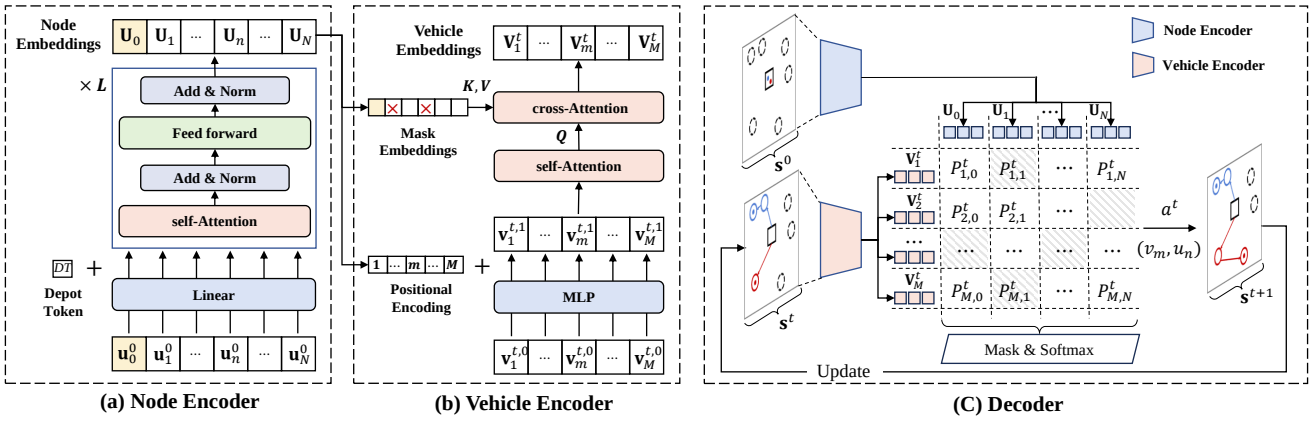


Figure 2: Architecture of Policy Network 2D-Ptr (π_θ)

u_n, we first map it to a new space through a linear layer, i.e.,

$$\bar{\mathbf{u}}_n^0 = \begin{cases} \mathbf{u}_n^0 W + DT, & \text{if } n = 0 \\ \mathbf{u}_n^0 W, & \text{otherwise.} \end{cases} \tag{2}$$

Here, $W \in \mathbb{R}^{3\times d}$ and $DT \in \mathbb{R}^d$ are trainable parameters. We name DT depot token to distinguish between depot and customer during training.

Let $H^{(0)} = \{\bar{\mathbf{u}}_0^0, \cdots, \bar{\mathbf{u}}_N^0\}$, and further transform it to $H^{(L)}$ through L identical layers, where each layer has two sublayers, namely the multi-head Attention (MHA) sublayer and the Feed Forward (FF) sublayer, with residual connection [9] and batch normalization [11] added. The layer-wise propagation rule is shown as follows:

$$\bar{H}^{(l)} = BN\left(H^{(l)} + MHA(H^{(l)}, H^{(l)}, H^{(l)})\right), \tag{3}$$

$$H^{(l+1)} = BN\left(\bar{H}^{(l)} + FF(\bar{H}^{(l)})\right). \tag{4}$$

Here, $FF(\cdot)$ is a double-layer network with ReLU activation function, $BN(\cdot)$ denotes batch normalization, and $MHA(\cdot,\cdot,\cdot)$ is an attention network with 8 heads. For the i-th head $ATT_i(\cdot,\cdot,\cdot)$, we have

$$ATT_i(Q,K,V) = softmax\left(\frac{[QW_i^Q][KW_i^K]^T}{\sqrt{d_i}}\right)VW_i^V, \tag{5}$$

$$MHA(Q,K,V) = \left(||_{i=1}^8 ATT_i(Q,K,V)\right)W^O, \tag{6}$$

where $||$ represents the concatenate operation, $W^O \in \mathbb{R}^{d\times d}$ and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d'}$ are trainable parameters, and $d' = d/8$ denotes the hidden dimension of each head,

We define the final output of the node encoder as node embeddings, i.e., $\mathbf{U} = H^{(L)}$, which will be reused for multiple times in the decoder to construct the solution to the problem instance S.

3.3.2 Vehicle Encoder. Unlike fixed node embeddings, vehicle embeddings change over step t during the decision-making process. Let $\mathbf{v}_m^{t,0} = (\chi_m, \rho_m, \hat{\rho}_m^t, \delta_m^t)$ denote the initial representation of vehicle v_m at step t, we have

$$\mathbf{v}_m^{t,1} = MLP(\mathbf{v}_m^{t,0}) + PE_m^t W. \tag{7}$$

Here, $MLP(\cdot)$ is a trainable Multi-layer Perceptron, which maps the initial features to d dimension. $PE_m^t = \phi(\mathbf{U}, \ell_m^t)$ denotes the positional encoding obtained by looking up the node embeddings $\mathbf{U}$ with the last visited node ℓ_m^t as the index. In such way, we can get the dynamic features of v_m at step t.

We further input $\mathbf{V}^{t,1} = \{\mathbf{v}_1^{t,1}, \cdots, \mathbf{v}_M^{t,1}\} \in \mathbb{R}^{M\times d}$ to a self-Attention layer, which allows for information exchange among vehicles, enabling better decision-making from a global perspective, i.e.,

$$\mathbf{V}^{t,2} = \mathbf{V}^{t,1} + MHA(\mathbf{V}^{t,1}, \mathbf{V}^{t,1}, \mathbf{V}^{t,1}). \tag{8}$$

Moreover, we enrich the contextual information for the vehicle encoder by adopting the cross-Attention layer. We first design the mask embeddings with $\mathbf{U}^{mask}$, in which we remove the rows of all served customers ($\{n|1 \le n \le N \wedge q_n^t = 0\}$) from $\mathbf{U}$. Then, the vehicle embeddings $\mathbf{V}^t$ at step t is obtained by:

$$\mathbf{V}^t = \mathbf{V}^{t,2} + MHA(\mathbf{V}^{t,2}, \mathbf{U}^{mask}, \mathbf{U}^{mask}). \tag{9}$$

By switching q_n^t of served customers to 0 according to the transition rules shown in Tab.2, the designed mask embeddings can help to capture the situational awareness information and highlight the dynamic features of $\mathbf{V}^t$.

3.3.3 Decoder. Given the node embeddings $\mathbf{U} \in \mathbb{R}^{(N+1)\times d}$ from node encoder and vehicle embeddings $\mathbf{V}^t \in \mathbb{R}^{M\times d}$ from vehicle encoder at step t, the decoder outputs a probability distribution $P^t \in \mathbb{R}^{M\times(N+1)}$ as the priority order of all actions, based on which we get a "2D array pointer" for identifying the t-th tuple of the output sequence.

Let $\beta_{m,n}^t$ represents the attention score between vehicle v_m and node u_n at step t, we have

$$\beta_{m,n}^t = \begin{cases} -\infty, & \text{if } q_n^t = 0 \wedge n \neq 0 \\ -\infty, & \text{if } \ell_m^t = 0 \wedge n = 0 \\ -\infty, & \text{if } \rho_m - \hat{\rho}_m^t < q_n^t \\ \lambda tanh(\frac{\mathbf{V}_m^t \mathbf{U}_n}{\sqrt{d}}), & \text{otherwise,} \end{cases} \tag{10}$$

where d is the dimension of node embeddings (or vehicle embeddings), $tanh(\cdot)$ is the activation function, and λ ($\lambda = 10$ in our manuscript) is the manually set hyper-parameter. In the action

space of $M \times (N+1)$, some actions are invalid at step t and need to be masked. For example, 1) the selected customer has been visited before step t (i.e., $q_n^t = 0 \wedge n \neq 0$); 2) the selected vehicle visits the depot twice in a row (i.e., $\ell_m^t = 0 \wedge n = 0$); and 3) the demand of the selected node exceeds the remaining capacity of corresponding vehicle (i.e., $\rho_m - \hat{\rho}_m^t < q_n^t$).

The probability of selecting a tuple (v_m, u_n) at step t is defined as follows:

$$P_{m,n}^t = \frac{exp(\beta_{m,n}^t)}{\sum_{m,n} exp(\beta_{m,n}^t)}. \tag{11}$$

The action $a^t = (v_m, u_n)$ can be selected either by retrieving the maximum probability greedily or sampling according to the probability distribution P^t.

At each step t, we select an action a^t by the policy network π_θ. We repeat the above steps until all customers have been visited, resulting in a tuple sequence of length T that can be used to construct the solution Γ for the problem instance S. It is worth noting that 2D-Ptr is composed of a series of attention modules, so its input is not constrained by the number of vehicles and customers, making the 2D-Ptr have good generalization.

4 EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed policy network through a series of experiments. First, its superiority in HCVRP is verified by comparing it with state-of-the-art methods (including heuristic algorithms and DRL based methods). Second, each component's impact on the performance of system is analyzed through ablation study. Third, its generalization ability is evaluated by deploying a well-trained model to new scenarios. To ensure fair comparisons, all computational experiments are performed on servers with Hygon C86 7185 CPUs and Vega20 A1 SERVER GPUs. Our code is available here[1], realized by the PyTorch framework.

4.1 Experimental settings

4.1.1 Dataset. Our method aims to solve the generalization problem of HCVRP, so that a well-trained model can adapt to problem instances with different numbers of vehicles (M) or customers (N). We consider 12 settings of problem sizes with $M \in \{3, 5, 7\}$ and $N \in \{40, 60, 80, 100\}$. For simplicity, we refer to the problem size with M vehicles and N customers as VM-UN, e.g., V3-U40.

For all customers and depot, we follow the existing work [16], sample their coordinates from the grid of $[0, 1] \times [0, 1]$ and demands from the set $\{1, 2, \cdots, 9\}$ according to the uniform distribution. Here, the demand of the depot is set to 0. As for fleet, existing work simply keeps it fixed. As a result, the performance of a well-trained model will drastically deteriorate when applied to another fleet (e.g., the capacity or speed of a vehicle is changed). To verify the fleet generalization, we increase the diversity of fleets in our dataset by setting each vehicle's capacity to a discrete value randomly sampled from the set $\{20, 21, \cdots, 40\}$, and the speed to a value sampled from $[0.5, 1]$ according to the uniform distribution.

4.1.2 Baseline. Three state-of-the-art DRL-based methods are selected as baselines: 1) **AM** [15], learning a Transformer-based model via REINFORCE for solving the standard VRPs; 2) **ET** [20], a state-of-the-art method for min-max MTSP and MPDP, generating sequential actions and considering an equitable workload; 3) **DRL**$_{Li}$ [16], a DRL-based method proposed by Li et al. for solving the HCVRP, in which each action consists of two subactions that are executed sequentially. We also employ three well known heuristic methods for comparison, which include: 4) **GA** [13], Genetic Algorithm; 5) **SA** [10], Simulated Annealing; 6) **SISR** [4], a state-of-the-art heuristic algorithm for solving CVRP and its variants.

The original ET and AM are not designed for HCVRP, and thus we modify their input layers and decoder masks to accept heterogeneous features of the vehicles as input, while keeping other network structures unchanged. In addition, AM can only construct solution for a single vehicle, therefore we follow the settings of literature [16] and assign vehicles to take turns to decide the next node. As for the heuristic methods, the hyper-parameters are found by trial and error. The configuration details of all the baselines and our method are given in Appendix.

4.1.3 Evaluation metrics. Following [16], we use 3 metrics including the average objective value, the average gap of objective value relative to SISR (the best heuristic baseline) and the average computation time. For DRL-based solvers, there are two strategies available for the decoder during testing: 1) Greedy, which constructs the solution by selecting actions with the maximum probability; 2) Sampling, which retrieves the best one from k solutions obtained by sampling actions based on probability distribution. Here, we set k to 1280 and 12800, and term them as Sample1280 and Sample12800, respectively.

4.2 Comparative Study

Tab.3 shows the experimental results of different methods, from which we can draw the following conclusions: 1) Heuristic methods have excellent performance in most problem instances, especially SISR achieving the smallest objective values. However, these methods are very time-consuming, especially on the large problem scales like V7-U100, even taking more than 1000 seconds. 2) Compared to heuristic methods, DRL-based methods exhibit significant speed improvements with comparable performance. With the advantage of offline training, DRL-based methods can learn common patterns of instances, which saves a substantial amount of online computation time; 3) Although with slightly longer computation time, both Sample1280 and Sample12800 achieve smaller objective values and gaps than Greedy, which demonstrates the effectiveness of sampling strategy in improving the solution quality; 4) 2D-Ptr outperforms all DRL-based baselines in terms of objective value and gap. We attribute it to the proposed policy network planning paths based on the priority order of actions, which can simultaneously solve vehicle selection and node selection problems within a single action, enabling it to search in a more rational and broader action space. The experimental results in Fig.3 validate our hypothesis, where we record the validation objective values of 2D-Ptr and DRL$_{Li}$ (the best DRL-based baseline) based on the Greedy strategy during the training process. From the experimental results, it can be seen that with the support of the proposed policy network, 2D-Ptr can be quickly optimized in the correct direction.

[1] https://github.com/farkguidao/2D-Ptr

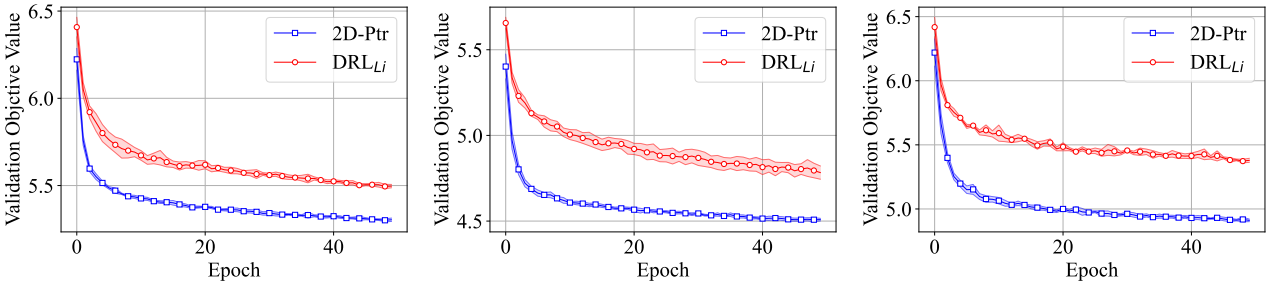
Table 3: Performance Comparison. Each test set has 1280 fixed problem instances. The notation (g.), (s.1280) and (s.12800) refer to Greedy, Sample1280 and Sample12800, respectively.

	Methods	U40			U60			U80			U100		
		obj.	gap(%)	time(s)	obj.	gap(%)	time(s)	obj.	gap(%)	time(s)	obj.	gap(%)	time(s)
V3	SISR	4.77	0.00	262.62	6.57	0.00	478.38	8.52	0.00	750.26	10.29	0.00	1084.84
	GA	6.35	33.22	241.92	9.21	40.25	411.16	12.32	44.68	612.47	15.33	48.93	845.08
	SA	5.07	6.27	229.47	7.04	7.16	382.52	9.17	7.60	561.95	11.13	8.13	765.30
	AM (g.)	6.32	32.57	0.10	8.49	29.2	0.14	10.81	26.93	0.19	12.68	23.23	0.25
	AM (s.1280)	5.48	14.93	0.18	7.62	16.07	0.34	9.92	16.45	0.50	11.82	14.82	0.72
	AM (s.12800)	5.37	12.61	0.83	7.49	14.08	1.86	9.77	14.67	3.18	11.66	13.29	4.56
	ET (g.)	5.60	17.35	0.21	7.58	15.46	0.28	9.76	14.61	0.38	11.74	14.08	0.45
	ET (s.1280)	5.20	9.01	0.29	7.14	8.64	0.52	9.19	7.88	0.74	11.20	8.84	1.02
	ET (s.12800)	5.15	7.97	1.06	7.08	7.80	2.34	9.12	7.04	4.03	11.13	8.10	6.28
	DRL_{Li} (g.)	5.47	14.66	0.23	7.43	13.07	0.34	9.64	13.22	0.46	11.44	11.13	0.58
	DRL_{Li} (s.1280)	5.09	6.79	0.42	6.97	6.12	0.73	9.10	6.78	1.10	10.90	5.94	1.48
	DRL_{Li} (s.12800)	5.05	5.91	1.66	6.91	5.23	3.38	9.02	5.89	5.61	10.82	5.15	7.97
	2D-Ptr (g.)	5.29	10.92	0.13	7.20	9.63	0.20	9.24	8.53	0.27	11.12	8.04	0.31
	2D-Ptr (s.1280)	4.96	3.91	0.20	6.82	3.87	0.32	8.85	3.86	0.44	10.71	4.11	0.55
	2D-Ptr (s.12800)	4.92	3.09	0.88	6.77	3.08	1.53	8.78	3.13	2.39	10.65	3.44	3.05
V5	SISR	2.94	0.00	260.21	4.00	0.00	484.14	5.10	0.00	759.17	6.17	0.00	1098.69
	GA	4.88	66.10	342.84	6.89	72.29	564.44	8.95	75.55	820.15	10.93	77.24	1099.14
	SA	3.21	9.18	314.29	4.39	9.74	509.72	5.61	10.02	735.63	6.80	10.20	983.37
	AM (g.)	4.15	41.15	0.10	5.51	37.76	0.15	6.87	34.84	0.20	8.10	31.42	0.24
	AM (s.1280)	3.54	20.66	0.19	4.82	20.69	0.33	6.19	21.46	0.52	7.45	20.75	0.70
	AM (s.12800)	3.46	17.84	0.84	4.72	18.19	1.84	6.08	19.26	3.19	7.33	18.86	4.50
	ET (g.)	3.57	21.45	0.20	4.76	19.15	0.30	6.01	17.75	0.37	7.25	17.51	0.46
	ET (s.1280)	3.29	11.98	0.33	4.46	11.59	0.55	5.64	10.65	0.75	6.85	11.02	0.94
	ET (s.12800)	3.26	10.85	1.17	4.42	10.5	2.58	5.59	9.62	4.12	6.79	10.13	5.27
	DRL_{Li} (g.)	3.59	22.36	0.27	4.71	17.93	0.40	5.97	17.07	0.54	7.06	14.49	0.67
	DRL_{Li} (s.1280)	3.24	10.28	0.52	4.34	8.47	0.88	5.54	8.64	1.36	6.65	7.76	1.87
	DRL_{Li} (s.12800)	3.20	8.83	2.12	4.29	7.31	4.18	5.48	7.45	7.03	6.58	6.71	9.99
	2D-Ptr (g.)	3.34	13.58	0.13	4.48	12.01	0.20	5.65	10.71	0.24	6.75	9.40	0.32
	2D-Ptr (s.1280)	3.09	5.27	0.22	4.20	5.01	0.36	5.36	5.11	0.50	6.46	4.73	0.63
	2D-Ptr (s.12800)	3.06	4.26	1.07	4.16	4.02	1.85	5.32	4.21	2.94	6.41	3.92	3.75
V7	SISR	2.29	0.00	262.55	2.91	0.00	486.60	3.69	0.00	764.84	4.45	0.00	1102.31
	GA	4.35	90.36	444.18	5.98	105.29	715.44	7.58	105.30	1019.83	9.10	104.61	1362.07
	SA	2.50	9.24	397.85	3.30	13.38	638.45	4.17	13.09	908.11	5.01	12.58	1196.24
	AM (g.)	3.15	37.79	0.10	4.15	42.45	0.16	5.18	40.54	0.19	6.13	37.81	0.24
	AM (s.1280)	2.66	16.29	0.20	3.63	24.68	0.34	4.64	25.80	0.53	5.58	25.46	0.70
	AM (s.12800)	2.60	13.78	0.83	3.55	22.04	1.90	4.55	23.30	3.25	5.47	23.16	4.58
	ET (g.)	2.69	17.69	0.20	3.58	22.93	0.29	4.43	19.98	0.41	5.23	17.59	0.47
	ET (s.1280)	2.46	7.65	0.32	3.33	14.49	0.53	4.17	12.88	0.76	4.98	11.99	0.99
	ET (s.12800)	2.43	6.42	1.25	3.30	13.2	2.48	4.13	11.87	4.01	4.94	11.09	5.70
	DRL_{Li} (g.)	2.67	16.81	0.28	3.60	23.49	0.45	4.52	22.46	0.60	5.38	20.87	0.79
	DRL_{Li} (s.1280)	2.43	6.06	0.58	3.25	11.59	1.07	4.13	11.94	1.60	4.98	11.82	2.27
	DRL_{Li} (s.12800)	2.40	4.96	2.46	3.20	10.03	5.11	4.07	10.34	8.23	4.91	10.43	12.15
	2D-Ptr (g.)	2.50	9.33	0.13	3.31	13.81	0.19	4.14	12.26	0.24	4.92	10.49	0.32
	2D-Ptr (s.1280)	2.35	2.73	0.23	3.09	6.01	0.36	3.90	5.65	0.55	4.68	5.22	0.69
	2D-Ptr (s.12800)	2.33	2.02	1.24	3.05	4.86	1.98	3.86	4.58	3.43	4.64	4.30	4.42

4.3 Ablation Study

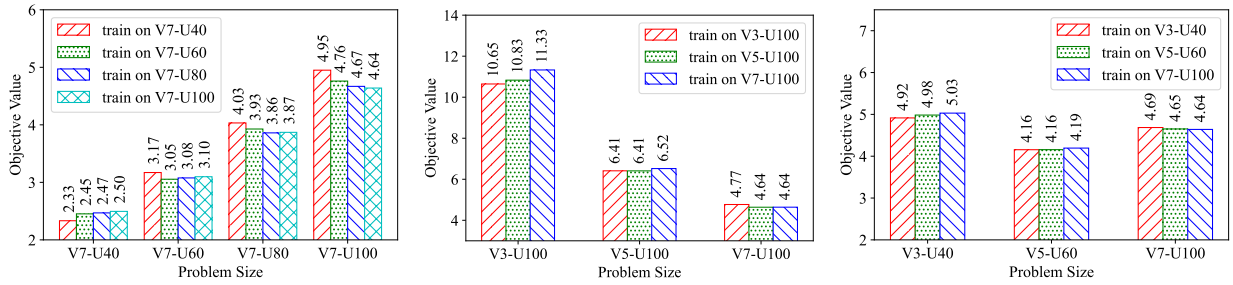
In this section, we further investigate the impact of each component of 2D-Ptr on performance improvement. We name the four variants that remove depot token, positional encoding, self-Attention layer (vehicle), and cross-Attention layer as Var^{DT}, Var^{PE}, Var^{SA} and Var^{CA}, respectively. As shown in Tab.4, all four variants of 2D-Ptr result in performance degradation on the V5-U60 problem size.

Our results demonstrate the importance of positional encoding, self-Attention, and cross-Attention for supporting the vehicle encoder. By incorporating the last visited node information of each vehicle into vehicle embeddings, the positional encoding helps to



(a) V3-U40 (Small) (b) V5-U60 (Medium) (c) V7-U100 (Large)

Figure 3: Convergence curves on different problem sizes. All results are presented in terms of the mean and standard deviation of five runs with different random seeds. 2D-Ptr consistently outperforms DRL$_{Li}$ by a big margin.



(a) Different customer sizes (b) Different fleet sizes (c) Different customer and fleet sizes

Figure 4: Generalization performance. The smaller the performance gap, the stronger the generalization.

Table 4: Ablation study for the components of 2D-Ptr.

Methods	Greedy		Sample1280		Sample12800	
	obj.	gap(%)	obj.	gap(%)	obj.	gap(%)
Var^{DT}	4.51	12.73	4.21	5.30	4.17	4.26
Var^{PE}	4.77	19.32	4.39	9.72	4.33	8.32
Var^{SA}	4.54	13.61	4.23	5.92	4.18	4.70
Var^{CA}	4.55	13.84	4.26	6.53	4.21	5.40
2D-Ptr	4.48	12.01	4.20	5.01	4.16	4.02

introduce crucial location information and create order bias among vehicles, enabling simultaneous decision-making to be modeled as a sequential process. In addition, the exchange of information among vehicles through the self-Attention layer promotes the collaboration of multiple vehicles, resulting in more robust decision-making from a global perspective. With respect to the cross-Attention layer, we believe its significant improvement is due to the rich contextual information provided by mask embedding, which enables vehicles to grasp task progress in real-time and adjust their embeddings based on remaining tasks.

As for the depot token, we find it is useful in addressing heterogeneous problems in node encoder. The depot is a place for vehicles to refill. Therefore, it allows vehicles to visit multiple times, which is different from the nature of customers. Depot token is a trainable vector that helps bridge the gap between customers and depot, thereby generating higher quality embeddings.

4.4 Generalization Study

This subsection tests the generalization ability of the well-trained model to a new scenario. To save online computation time, our model is trained offline. Since differences might exist between the training data and real scenarios, the generalization ability of well-trained model is rather critical.

Here, we conduct experiments to apply the policy learned for a problem size to different ones without any further tuning. 1) Fix fleet size to 7, and test the performance of well-trained models learned for V7-U40, V7-U60, V7-U80, and V7-U100 on all four scenarios; 2) Fix customer size to 100, and test the performance of well-trained models learned for V3-U100, V5-U100, and V7-U100 on all three scenarios; 3) Test the performance of well-trained models learned for V3-U40, V5-U60, and V7-U100 on all three scenarios. Note that the latter two settings add additional validation of models' generalization to different fleet sizes, which has not been considered in existing works. The experimental results based on the strategy of Sample12800 are shown in Fig.4.

From the experimental results, we observe that when the training and testing settings are the same, 2D-Ptr can achieve the smallest objective value. Although the performance of 2D-Ptr degrades when

applied to different settings, the change is very small. For example, in Fig.4(b), applying the policy of V3-U100 to V5-U100 results in almost the same objective value with the best settings (i.e., applying the policy of V5-U100 to V5-U100). In addition, the performance of 2d-ptr in the new scenarios is higher than that of most models (baselines) trained for this scenario. For example, in Fig.4(c), the model of 2D-Ptr trained on V3-U40 achieves an objective value of 4.69 in solving V7-U100, outperforming SA (5.01) and DRL$_{Li}$ (4.91), etc. This once again validates the generalization ability of 2D-Ptr.

5 CONCLUSION

In this paper, by factorizing HCVRP into the sequence encoder-decoder form, we propose a pointer network extension model called 2D-Ptr to learn a policy for the route construction based on the min-max objective function. Instead of planning paths based on the priority order of vehicles, 2D-Ptr plans paths based on the priority order of actions. Specifically, the policy network outputs a 2D array pointer to select a tuple from all combinations of vehicles and nodes as the element of sequence τ, which is further used to construct the solution Γ. In addition, 2D-Ptr consists of a series of carefully designed attention modules, so that it can be used to automate the task of routing a heterogeneous fleet for any configuration.

Our method surpasses existing state-of-the-art techniques concerning the trade-off between cost and runtime. Through our experiments, we have demonstrated the superior performance as well as excellent generalization ability of 2D-Ptr, positioning it as a valuable tool for various delivery service industries aiming to enhance efficiency and reduce costs. Future work will extend 2D-Ptr to dynamic scenes, and generate fast and scalable solutions by relying on centralized programming under partial observability.

A APPENDIX

A.1 Settings for 2D-Ptr

The hyper-parameter configuration for training the policy network is shared for all problem sizes. Specifically, the entire training process lasts for 50 epochs, where each epoch contains 2500 batches, and each batch contains 512 randomly generated problem instances. We adopt the Adam optimizer to train the model, with an initial learning rate of 0.0001 and a decay factor of 0.995 per epoch. In addition, the gradient clipping is applied to stabilize the training process, where the maximum L2 norm is set to 3.0. As for the hyper-parameters of the model structure, the number of heads in all multi-head attention is set to 8, while the embedding dimension d and the number L of layers in node encoder are set to 128 and 3, respectively.

As the problem size increases, the training time of the model will become longer. For example, for the smallest problem size V3-U40, the average training time of each epoch is 31min, and for the largest problem size V7-U100, each epoch lasts 70min (2 GPUs). Note that when testing, the 1280 instances of each test set are resolved serially on a single GPU for 2D-Ptr, and the same for other DRL-based baselines, to ensure a fair comparison of performance.

A.2 Settings for DRL-based baselines

All DRL-based baselines are implemented based on the source code and hyper-parameters provided in the original papers, and are all trained for 50 epochs, which is consistent with 2D-Ptr. For DRL$_{Li}$, since the features of the fleet are hard-coded and fixed in source code, we adjust it to accept different fleets as input to support our dataset. For ET, since it is originally used to solve MTSP and MPDP, we first adjust the mask in decoder to produce feasible solutions for HCVRP, and then inject vehicle features to the input layer and context encoder of ET. As for AM, we follow the adjustment for AM in DRL$_{Li}$ (using AM as the baseline), first select vehicle in turn, and then select the next node for it. Furthermore, vehicle features are considered in context vector generation to identify vehicle differences.

A.3 Settings for heuristic baselines

For the heuristic baselines, all methods tune the input and objective function to the min-max HCVRP. For SISR, we adopt an open source implementation[2] based on Java and the same hyper-parameters in the original paper. SA and GA are implemented in scikit-opt[3], which is an open source heuristic algorithm library based on Python, and their hyper-parameters are carefully tuned. We gradually increase the iteration steps with customer size to improve the quality of the solution. Finally, the hyper-parameters of all heuristic baselines are listed in the Tab.5.

Since these heuristic methods are very time-consuming, we distribute the instances to multiple identical servers, and report the average results. For example, it takes over 1000 seconds for SISR to solve a single instance of V3-U100, thus solving a test set containing 1280 instances sequentially would require approximately 15 days. However, by distributing these instances to 64 identical servers, we have managed to reduce this time to 6 hours.

Table 5: Hyper-parameters for heuristic baselines

Method	Settings
SISR	$\bar{c} = 10$, $L^{max} = 10$, $\alpha = 0.001$, $\beta = 0.01$, $T_0 = 100$, $T_f = 1$, iter $= 3 \times 10^5 \times N$
GA	$n = 200$, iter $= 40 \times N$, $P_m = 0.8$, $P_c = 1$
SA	$T_0 = 100$, $T_f = 1 \times 10^{-7}$, $L = 20 \times N$, $\alpha = 0.98$

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 62036010; in part by the Provincial Natural Science Foundation of Henan under Grant 232300421095 and Grant 222102210248; in part by the China Postdoctoral Science Foundation under Grant 2022T150590.

REFERENCES

[1] Roberto Baldacci and Aristide Mingozzi. 2009. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming* 120 (2009), 347–380.

[2] https://github.com/chenmingxiang110/tsp_solver
[3] https://scikit-opt.github.io/scikit-opt

[2] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2017. 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. https://openreview.net/forum?id=Bk9mxlSFx

[3] Luca Bertazzi, Bruce Golden, and Xingyin Wang. 2015. Min–Max vs. Min–Sum Vehicle Routing: A worst-case analysis. *European Journal of Operational Research* 240, 2 (2015), 372–381. https://doi.org/10.1016/j.ejor.2014.07.025

[4] Jan Christiaens and Greet Vanden Berghe. 2020. Slack induction by string removals for vehicle routing problems. *Transportation Science* 54, 2 (2020), 417–433.

[5] Jonas K. Falkner and Lars Schmidt-Thieme. 2020. Learning to Solve Vehicle Routing Problems with Time Windows through Joint Attention. *CoRR* abs/2006.09100 (2020). arXiv:2006.09100 https://arxiv.org/abs/2006.09100

[6] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. 2021. Generalize a small pretrained model to arbitrarily large TSP instances. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 7474–7482.

[7] Bruce Golden, Arjang Assad, Larry Levy, and Filip Gheysens. 1984. The fleet size and mix vehicle routing problem. *Computers & Operations Research* 11, 1 (1984), 49–66. https://doi.org/10.1016/0305-0548(84)90007-8

[8] Mordecai Haimovich and Alexander HG Rinnooy Kan. 1985. Bounds and heuristics for capacitated routing problems. *Mathematics of operations Research* 10, 4 (1985), 527–542.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[10] İLHAN İlhan. 2021. An improved simulated annealing algorithm with crossover operator for capacitated vehicle routing problem. *Swarm and Evolutionary Computation* 64 (2021), 100911.

[11] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pmlr, 448–456.

[12] Yuan Jiang, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. 2022. Learning to solve routing problems via distributionally robust optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9786–9794.

[13] Saso Karakatic and Vili Podgorelec. 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Appl. Soft Comput.* 27 (2015), 519–532. https://doi.org/10.1016/j.asoc.2014.11.005

[14] Minjun Kim, Junyoung Park, and Jinkyoo Park. 2023. Learning to CROSS exchange to solve min-max vehicle routing problems. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/forum?id=ZcnzsHC10Y

[15] Wouter Kool, Herke van Hoof, and Max Welling. 2019. Attention, Learn to Solve Routing Problems!. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ByxBFsRqYm

[16] Jingwen Li, Yining Ma, Ruize Gao, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang. 2022. Deep Reinforcement Learning for Solving the Heterogeneous Capacitated Vehicle Routing Problem. *IEEE Transactions on Cybernetics* 52, 12 (2022), 13572–13585. https://doi.org/10.1109/TCYB.2021.3111082

[17] Hebin Liang, Yi Ma, Zilin Cao, Tianyang Liu, Fei Ni, Zhigang Li, and Jianye Hao. 2023. SplitNet: A Reinforcement Learning Based Sequence Splitting Method for the MinMax Multiple Travelling Salesman Problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 8720–8727.

[18] Christian Prins. 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Eng. Appl. Artif. Intell.* 22, 6 (2009), 916–928. https://doi.org/10.1016/j.engappai.2008.10.006

[19] Wei Qin, Zilong Zhuang, Zizhao Huang, and Haozhe Huang. 2021. A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. *Comput. Ind. Eng.* 156 (2021), 107252. https://doi.org/10.1016/j.cie.2021.107252

[20] Jiwoo Son, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. 2023. Solving NP-hard Min-max Routing Problems as Sequential Generation with Equity Context. *arXiv preprint arXiv:2306.02689* (2023).

[21] José Manuel Vera and Andres G Abad. 2019. Deep reinforcement learning for routing a heterogeneous fleet of vehicles. In *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, 1–6.

[22] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems*, Vol. 28. Curran Associates, Inc., 2692–2700. https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf

[23] Xingyin Wang, Bruce Golden, Edward Wasil, and Rui Zhang. 2016. The min–max split delivery multi-depot vehicle routing problem with minimum service time requirement. *Computers & Operations Research* 71 (2016), 110–126. https://doi.org/10.1016/j.cor.2016.01.008

[24] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. 2022. Learning Improvement Heuristics for Solving Routing Problems. *IEEE Trans. Neural Networks Learn. Syst.* 33, 9 (2022), 5057–5069. https://doi.org/10.1109/TNNLS.2021.3068828

[25] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. 2021. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 12042–12049.

[26] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. 2021. NeuroLKH: Combining Deep Learning Model with Lin-Kernighan-Helsgaun Heuristic for Solving the Traveling Salesman Problem. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 7472–7483. https://proceedings.neurips.cc/paper_files/paper/2021/file/3d863b367aa379f71c7afc0c9cdca41d-Paper.pdf

[27] Yunqiu Xu, Meng Fang, Ling Chen, Gangyan Xu, Yali Du, and Chengqi Zhang. 2022. Reinforcement Learning With Multiple Relational Attention for Solving Vehicle Routing Problems. *IEEE Trans. Cybern.* 52, 10 (2022), 11107–11120. https://doi.org/10.1109/TCYB.2021.3089179

[28] Vincent F. Yu, Parida Jewpanya, A.A.N. Perwira Redi, and Yu-Chung Tsao. 2021. Adaptive neighborhood simulated annealing for the heterogeneous fleet vehicle routing problem with multiple cross-docks. *Computers & Operations Research* 129 (2021), 105205. https://doi.org/10.1016/j.cor.2020.105205

[29] Ke Zhang, Fang He, Zhengchao Zhang, Xi Lin, and Meng Li. 2020. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transportation Research Part C: Emerging Technologies* 121 (2020), 102861.

[30] Zefang Zong, Meng Zheng, Yong Li, and Depeng Jin. 2022. Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9980–9988.