Multi-Agent Pointer Transformer: Seq-to-Seq Reinforcement Learning for Multi-Vehicle Dynamic Pickup-Delivery Problems

Zengyu Zou[1, 4], **Jingyuan Wang**[1, 2, 3, 4, *], **Yixuan Huang**[1, 4], **Junjie Wu**[2, 3]
[1]School of Computer Science and Engineering, Beihang University, Beijing, China
[2]School of Economics and Management, Beihang University, Beijing, China
[3]MIIT Key Laboratory of Data Intelligence and Management, Beihang University, Beijing, China
[4]MOE Engineering Research Center of Advanced Computer Application Technology, Beihang University, China
{zzy0001213, jywang, yixuanhuang, wujj}@buaa.edu.cn

Abstract

This paper addresses the cooperative Multi-Vehicle Dynamic Pickup and Delivery Problem with Stochastic Requests (MVDPDPSR) and proposes an end-to-end centralized decision-making framework based on sequence-to-sequence, named Multi-Agent Pointer Transformer (MAPT). MVDPDPSR is an extension of the vehicle routing problem and a spatio-temporal system optimization problem, widely applied in scenarios such as on-demand delivery. Classical operations research methods face bottlenecks in computational complexity and time efficiency when handling large-scale dynamic problems. Although existing reinforcement learning methods have achieved some progress, they still encounter several challenges: 1) Independent decoding across multiple vehicles fails to model joint action distributions; 2) The feature extraction network struggles to capture inter-entity relationships; 3) The joint action space is exponentially large. To address these issues, we designed the MAPT framework, which employs a Transformer Encoder to extract entity representations, combines a Transformer Decoder with a Pointer Network to generate joint action sequences in an AutoRegressive manner, and introduces a Relation-Aware Attention module to capture inter-entity relationships. Additionally, we guide the model's decision-making using informative priors to facilitate effective exploration. Experiments on 8 datasets demonstrate that MAPT significantly outperforms existing baseline methods in terms of performance and exhibits substantial computational time advantages compared to classical operations research methods.

Code — https://github.com/Beihang-BIGSCity/MAPT

1 Introduction

The Pickup and Delivery Problem is a type of vehicle routing problem that has demonstrated its importance in many real-world applications, such as online food delivery. In real-world scenarios, the arrival time, origin, and destination of requests are often unpredictable, and there are strict requirements for service response times. Therefore, efficiently planning vehicle routes based on real-time updated information is particularly important in solving this problem. This work focuses on solving the cooperative Multi-Vehicle Dynamic Pickup and Delivery Problem with Stochastic Requests (MVDPDPSR). Unlike the traditional Pickup and Delivery Problem, in this problem, we need to route multiple vehicles, and the occurrence times of all requests are unknown in advance. Moreover, the origin and destination of each request are only revealed after the request emerges.

Existing methods for addressing dynamic vehicle routing problems fall into three categories: **heuristic methods**, **classical operations research methods**, and **reinforcement learning-based algorithms**. **Heuristic methods** have shown some success (Sheridan et al. 2013; Fikar 2018; Andersson 2021) by routing vehicles through manually designed heuristic rules, such as the nearest-distance rule. Their strengths are simplicity and efficiency, fitting real-time decision scenarios, but they have clear limits as they depend heavily on manually designed rules and struggle to ensure optimality. **Classical operations research methods** typically adopt the Rolling-Horizon paradigm, which transforms dynamic problems into static ones. These static subproblems are then solved using either exact algorithms (Lu and Dessouky 2004; Liu et al. 2018; Savelsbergh and Sol 1998) or metaheuristics (Schilde, Doerner, and Hartl 2011; Geiser, Hanne, and Dornberger 2020; Cai et al. 2022). This paradigm requires accumulating a sufficient number of requests before initiating planning, which fails to meet the needs of real-time dynamic decision-making. Also, both exact and metaheuristic algorithms have high computational complexity, greatly affecting their time efficiency in practice. With technology development, **reinforcement learning-based algorithms** have been used for multi-vehicle routing. Multi-agent reinforcement learning methods have been proposed for the static multi-vehicle pickup and delivery problem (Zong et al. 2022; Berto et al. 2024). They encode the current state with an Encoder, decode independently for each vehicle with a Decoder, and use a conflict handler in post-processing to solve vehicle conflicts.

However, these multi-agent methods still face the following issues: (1) Existing multi-agent reinforcement learning methods decode each agent independently, failing to model the joint probability distribution of agents' actions. Furthermore, the decisions generated by these methods may conflict among agents (e.g., competing for the same request). (2) Existing feature extraction networks fail to capture rela-

*Corresponding author

tionships between entities (e.g., stations, vehicles, requests), which are key components of the MVDPDPSR. (3) The joint action space is often exponentially large, which prevents reinforcement learning from exploring useful actions.

To tackle these remaining issues, we propose the Multi-Agent Pointer Transformer (MAPT), an end-to-end centralized decision-making framework based on sequence-to-sequence to solve MVDPDPSR. Specifically, we model the MVDPDPSR problem as a Markov Decision Process, use a Transformer Encoder to extract entity representations, and then combine a Transformer Decoder with a Pointer Network to convert the embedding sequence into a joint action sequence in an AutoRegressive manner. To extract relationships between entities, we propose a Relation-Aware Attention module that embeds entity relationships and adds them to the attention matrix. To address the challenge of reinforcement learning exploring a vast joint action space, we design informative priors based on load balancing and station distance, and fuse these priors with the action distribution decoded by the Decoder, enabling the model to select actions with higher potential rewards.

In summary, the main contributions of our work are as follows:

- We formalize the MVDPDPSR problem as a Markov Decision Process and design the Multi-Agent Pointer Transformer (MAPT) framework to decode joint actions and model the joint probability.
- We design a Relation-Aware Attention module to capture various relationships between entities.
- We design informative priors and integrate them into the model's decision distribution to facilitate effective exploration.
- We validate the effectiveness of MAPT on 8 datasets, and experiments show that MAPT outperforms various baselines and demonstrates significant computational time advantages over classical operations research methods.

2 Preliminaries

2.1 Problem Formulation

In this section, we formally define the Multi-Vehicle Dynamic Pickup and Delivery Problem with Stochastic Requests(MVDPDPSR). A typical MVDPDPSR scenario involves a fleet of vehicles (Multi-Vehicle) tasked with picking up and delivering cargos among a set of stations. The pickup and delivery requests arise dynamically over time, reflecting the "Dynamic" nature of the problem. Accordingly, stations, vehicles, and requests constitute the three fundamental components of a MVDPDPSR. We provide their formal definitions below.

Definition 1 (Stations)**.** The stations in MVDPDPSR are represented as a fully connected weighted graph $\mathcal{G} = \{N, \boldsymbol{E}\}$. Here, $N = \{n_1, \ldots, n_i, \ldots, n_I\}$ denotes the set of stations, where n_i represents the i-th station. $\boldsymbol{E} \in \mathbb{R}^{|N|\times|N|}$ is the distance matrix, where each element e_{n_i,n_j} indicates the travel time from station n_i to station n_j. In the station graph $\mathcal{G}$, each station can serve as either an origin (pickup station) or a destination (delivery station) for cargos.

Definition 2 (Vehicles)**.** A MVDPDPSR scenario involves K vehicles, where each vehicle v_k is represented by a tuple of four state variables:

$$v_k = \langle\, cap_k,\ spa_k,\ to_k,\ dist_k \,\rangle. \tag{1}$$

The meanings of these variables are as follows:

- cap_k (capacity): the total capacity of vehicle v_k.
- spa_k^t (space): the remaining available space of vehicle v_k at time t. For notational brevity, we omit the superscript time index t when unambiguous (e.g. spa_k).
- to_k (destination): the current destination station of the vehicle.
- $dist_k$ (distance): the remaining distance (travel time) to reach the current destination.

If $dist_k \neq 0$, the vehicle is en route; otherwise, it is located at a station. Notably, the destination of a vehicle cannot be changed while it is en route.

Definition 3 (Requests)**.** The objective of MVDPDPSR is to fulfill M cargo delivery requests, where each request r_m is represented by a tuple of six state variables:

$$r_m = \langle\, from_m,\ to_m,\ val_m,\ vol_m,\ time_m,\ state_m \,\rangle. \tag{2}$$

The meanings of these variables are as follows:

- $from_m$ (origin): the origin station of request r_m.
- to_m (destination): the destination station of request r_m.
- val_m (value): the profit associated with fulfilling the request.
- vol_m (volume): the cargo volume required for the request, which consumes vehicle space.
- $time_m$ (appearance time): the time at which the request becomes visible to the system.
- $state_m$ (state): the current state of request r_m.

A request becomes visible only when the current time reaches its appearance time $time_m$. The state variable $state_m \in \{\text{unassigned}, \text{picked}, \text{delivered}\}$ indicates whether the request has not been picked up, has been picked up by a vehicle, or has been delivered.

In the MVDPDPSR scenario, the planning horizon is discretized into time slices $t = 0, \ldots, T$. At the initial time slice $t = 0$, all vehicles are located at their initial origin stations, and their available space equals their full capacity. At each time slice, a scheduler performs the following actions:

1. For every visible request in the unassigned state at time t (i.e. $\text{time}_m \leq t$), assign a vehicle that is currently at the same station and has sufficient space ($spa_k \geq vol_m$) to load the request's cargo. Update the state of the request to picked and reduce the vehicle's available space accordingly: $spa_k := spa_k - vol_m$.
2. For each vehicle at a station ($dist_k = 0$), select its next destination station and dispatch it. Set the vehicle's travel distance $dist_k$ to the distance between the current station and the selected destination.
3. For every vehicle in transit ($dist_k \neq 0$), decrease its remaining travel distance by one unit: $dist_k := dist_k - 1$.

4. For each vehicle arriving at its destination ($dist_k = 0$), unload all cargos whose destination matches the current station. Update the state of each delivered request to $\mathrm{delivered}$ and restore the vehicle's available space: $spa_k := spa_k + vol_m$ for each delivered request.

In this scenario, cargo delivery requests emerge dynamically over time, characterizing MVDPDPSR as a *dynamic* pickup-and-delivery problem.

A MVDPDPSR instance terminates when the planning horizon reaches the upper limit T. The goal of scheduling is to maximize the total profit of requests completed within the time limit while minimizing the aggregate travel cost.

2.2 Markov Decision Process (MDP)

MVDPDPSR is a sequential decision-making problem over time, and we model its decision process as a Markov decision process, defined as follows:

Definition 4 (Observation/State)**.** Since we have a centralized decision system, the system can fully observe all state information at the current time step. The state at time step t includes the distance between stations e_{n_i,n_j}, all vehicle information v_k, and all appeared requests r_m where $time_m \leq t$. Notably, within a centralized decision-making system, the relationship $rel_{v_k,r_m} \in \{\mathrm{unassigned}, \mathrm{picked}, \mathrm{delivered}\}$ between each vehicle v_k and request r_m at time t must be explicitly tracked. This relationship indicates the request has not been picked up, has been picked up by the vehicle, or has been delivered by the vehicle.

Definition 5 (Action)**.** The decision includes decisions for requests and decisions for vehicles.

- Request decisions. For the set of $\mathrm{unassigned}$ requests $\mathcal{R}^t = \{r_m \mid state_m = \mathrm{unassigned} \wedge time_m \leq t\}$, we need to decide which vehicle to assign them to. The assigned vehicle's current location must be the same as the request's origin, i.e.,
$$A^t_{r_m} \in \{v_k \mid dist_k = 0 \wedge to_k = from_m\} \cup \{\tau\}, \quad (3)$$
where τ denotes the action of temporarily deferring the request assignment. The assignment must ensure that the vehicle's capacity meets the requirements.
- Vehicle decisions. For the set of vehicles that have reached their destinations $\mathcal{V}^t = \{v_k \mid dist_k = 0\}$, we need to decide their next destination station, i.e.,
$$A^t_{v_k} \in \{n_1, \ldots, n_I\}. \quad (4)$$

The final joint action space is the Cartesian product of all sub-action spaces:

$$A^t = \prod \left(\left\{ A^t_{r_m} \mid r_m \in \mathcal{R}^t \right\} \cup \left\{ A^t_{v_k} \mid v_k \in \mathcal{V}^t \right\} \right) \quad (5)$$

Definition 6 (Transition)**.** For vehicles $v_k \in \mathcal{V}^t$, we need to update their destination station to_k and remaining time to reach the new destination station $dist_k$ based on $A^t_{v_k}$, and update their remaining space based on unloaded and newly loaded goods. For vehicles $v_k \notin \mathcal{V}^t$, the remaining time to reach their destination stations decreases by one unit. For all requests assigned to vehicles, their $state_m$ and rel_{v_k,r_m} changes to picked, and for all requests that have reached their destinations, their $state_m$ and rel_{v_k,r_m} changes to $\mathrm{delivered}$.

Definition 7 (Objective/Reward)**.** The objective is to optimize the overall routing solution quality by maximizing the profit from completed requests while accounting for vehicle travel costs. Formally, we define the objective at time T as:

$$obj_T = \sum_{t<T} \sum_k \left(\sum_{m \in D^t_k} val_m - cost \cdot e_{to^t_k, to^{t+1}_k} \right), \quad (6)$$

where D^t_k denotes the set of requests delivered by vehicle v_k by time t, and $cost$ represents the cost per unit distance. The single-step reward at each decision point is then defined as the incremental change in the objective value:

$$rwd_t = obj_t - obj_{t-1}. \quad (7)$$

3 Methodology

3.1 System Overview

We propose the Multi-Agent Pointer Transformer (MAPT), which is built on the Transformer (Vaswani et al. 2017) Encoder-Decoder architecture. At each time step of the MDP, the current state is fed into MAPT for decision-making. The Encoder processes the states of all entities and incorporates a Relation-Aware Attention module to capture inter-entity relationships. The Decoder generates actions for each entity in an AutoRegressive manner. To enhance exploration during reinforcement learning, we integrate informative priors into the action selection process. The overall architecture is illustrated in Figure 1.

3.2 Encoder

Feature Representation with Dense Vectors First, we transform the raw input into dense vectors for unified representation. We compute some augmented inputs to enhance the model's perception capabilities, namely ori_i and $dest_i$, which represent the number of requests originating from station n_i, and the number of requests destined for station n_i, respectively. Next, we concatenate the raw inputs of stations, requests, and vehicles and pass them through a linear layer to transform them into dense vectors:

$$\boldsymbol{d}_{n_i} = [ori_i \| dest_i]\, \boldsymbol{W}^n, \quad \boldsymbol{d}_{n_i} \in \mathbb{R}^{hs} \quad (8)$$

$$\boldsymbol{d}_{v_k} = [cap_k \| spa_k \| dist_k]\, \boldsymbol{W}^v, \quad \boldsymbol{d}_{v_k} \in \mathbb{R}^{hs} \quad (9)$$

$$\boldsymbol{d}_{r_m} = [val_m \| vol_m]\, \boldsymbol{W}^r, \quad \boldsymbol{d}_{r_m} \in \mathbb{R}^{hs} \quad (10)$$

where $\|$ denotes the concatenation operation for scalars, and hs is the model's hidden size. Additionally, we transform global information into a dense vector, where m_t represents the current number of requests, as follows:

$$\boldsymbol{d}_g = [t \| m_t]\, \boldsymbol{W}^g, \quad \boldsymbol{d}_g \in \mathbb{R}^{hs}. \quad (11)$$

Relation-Aware Attention Although there are some neural network models for solving the Vehicle Routing Problem, these models cannot effectively capture the relationships between entities. To overcome these limitations, this study designs a Relation-Aware Attention module aimed at incorporating relationship information between entities into the embeddings. For any two entities (stations, requests,

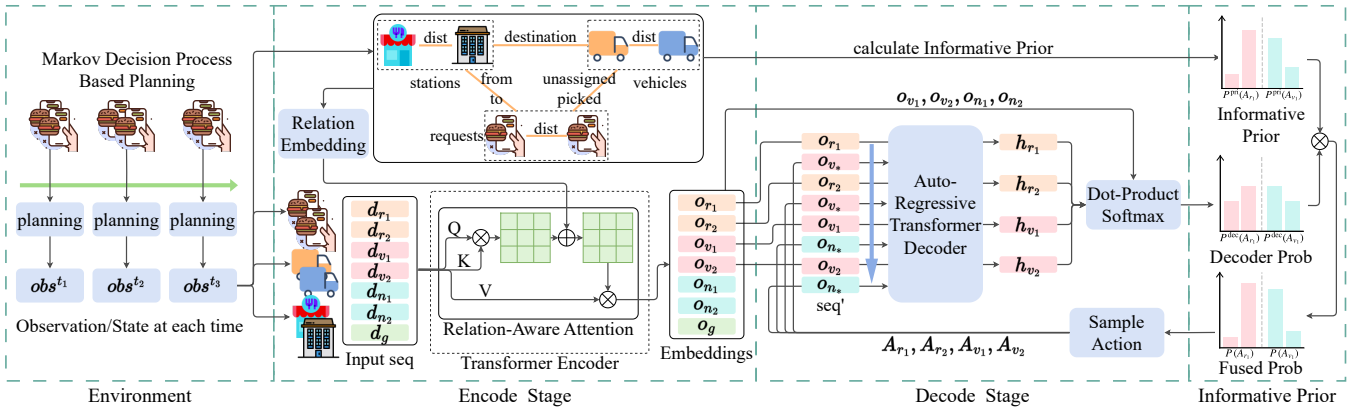


Figure 1: The framework of MAPT. The blue arrow indicates that the elements in the sequence are generated in an AutoRegressive manner. The elements marked with * are the actions that need to be decoded.

Relation	Vehicle	Request	Station
Vehicle	distance	unassigned picked delivered	is destination isn't destination
Request	unassigned picked delivered	distance	from to
Station	is destination isn't destination	from to	distance

Table 1: Relation Types between Vehicle, Request and Station.

vehicles), there are some relationships between them, as shown in Table 1. We design a relation encoding network Relation-Embedding$(u, v) \in \mathbb{R}$. If the relationship between entities u and v is a distance relationship, we use a linear layer to project it; for other relationships, we use a learnable parameter as their embedding. We denote the relationship matrix between entities as $\boldsymbol{R} \in \mathbb{R}^{|\boldsymbol{Q}|\times|\boldsymbol{K}|}$, which is calculated as follows:

$$\boldsymbol{R_{u,v}} = \text{Relation-Embedding}(q_u, k_v), \tag{12}$$

where q_u represents the u-th entity in Query sequence ($\boldsymbol{Q}$), and k_v represents the v-th entity in Key sequence ($\boldsymbol{K}$). Following the definition of the standard Transformer (Vaswani et al. 2017), we represent the Query sequence as $\boldsymbol{Q}$, the Key sequence as $\boldsymbol{K}$, the Value sequence as $\boldsymbol{V}$ and the head dimension as d_k. The Relation-Aware Attention is expressed as:

$$\text{Rel-Aware-Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{R}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T + \boldsymbol{R}}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{13}$$

We incorporate relationship information into the attention mechanism, enabling the model to capture relationships between entities when generating entity embeddings.

Encode with Relation-Aware Attention We use a standard Transformer Encoder and replace its attention layer with Relation-Aware Attention. We concatenate the dense vectors of all stations, requests, vehicles, and global information into a sequence and input them into the Encoder to obtain their respective embeddings:

$$[\boldsymbol{o}_{r_1}, \ldots, \boldsymbol{o}_{r_M}, \boldsymbol{o}_{v_1}, \ldots, \boldsymbol{o}_{v_K}, \boldsymbol{o}_{n_1}, \ldots, \boldsymbol{o}_{n_I}, \boldsymbol{o}_g] = \text{Encoder}([\boldsymbol{d}_{r_1}, \ldots, \boldsymbol{d}_{r_M}, \boldsymbol{d}_{v_1}, \ldots, \boldsymbol{d}_{v_K}, \boldsymbol{d}_{n_1}, \ldots, \boldsymbol{d}_{n_I}, \boldsymbol{d}_g]). \tag{14}$$

3.3 Decoder

Decoder Model We use a standard Transformer Decoder and replace its attention layer with Relation-Aware Attention. The cross-attention layer needs to perform cross-attention on the output of the Encoder to perceive all information about the current problem. Additionally, we add learnable positional encoding before the Decoder.

AutoRegressive Decoding We model the joint probability distribution A^t using chain rule decomposition, which is order-agnostic:

$$P(A^t) = \prod_m^M P(A^t_{r_m} \mid A^t_{r_1 \ldots r_{m-1}}) \times \prod_k^K P(A^t_{v_k} \mid A^t_{v_1 \ldots v_{k-1}}, A^t_{r_1 \ldots r_M}). \tag{15}$$

From this formula, we can see that each request needs to refer to the decision results of previous requests when making decisions; each vehicle needs to refer to the decision results of previous vehicles and all requests when making decisions. Due to this sequential decision-making process, we adopt an AutoRegressive approach to generate the decision sequence and model it in a sequence-to-sequence framework, as also observed by Wen et al. (2022). We combine the Transformer's ability to capture long-distance dependencies with the Pointer Network's (Vinyals, Fortunato, and Jaitly 2015) ability to handle entity selection, using the Transformer Decoder to decode actions.

We detail our decoding process, for notational convenience, we omit the superscript time index t when unambiguous. Let our sequence be seq, initially empty. When de-

coding the action for the m-th request r_m, we first append it into the sequence, which then becomes:

$$\text{seq} = [r_1, A_{r_1}, \ \ldots, r_m]. \tag{16}$$

We first convert entities in the sequence to their corresponding embeddings to obtain seq' (e.g., converting request r_m to its embedding $\boldsymbol{o}_{r_m}$ which is output of the Transformer Encoder). The $\text{Decoder}(\text{seq}')$ produces a hidden sequence of the same length as seq'. We take the last item of the hidden sequence, $\boldsymbol{h}_{r_m}$, as the query vector and use the embeddings of all vehicles as the key vectors. By calculating the attention scores between them, we can quantify the request's preference for each vehicle:

$$P^{\text{dec}}(A_{r_m} = v_k) = \text{softmax}_k(\boldsymbol{h}_{r_m}^\top [\boldsymbol{o}_{v_1}, \ldots, \boldsymbol{o}_{v_K}]), \tag{17}$$

where $\text{softmax}_k(\cdot)$ denotes the k-th element of the softmax output. We determine A_{r_m} by sampling from $P^{\text{dec}}(A_{r_m})$ or selecting the maximum probability vehicle. This decoding result is then placed at the end of seq, providing critical reference information for subsequent decisions:

$$\text{seq} = [r_1, A_{r_1}, \ \ldots, \ r_m, A_{r_m}]. \tag{18}$$

This process can be repeated to decode all requests. We decode all vehicles in the same manner:

$$P^{\text{dec}}(A_{v_k} = n_i) = \text{softmax}_i(\boldsymbol{h}_{v_k}^\top [\boldsymbol{o}_{n_1}, \ldots, \boldsymbol{o}_{n_I}]). \tag{19}$$

After all decoding is completed, the elements in the sequence are:

$$\text{seq} = [r_1, A_{r_1}, \ \ldots, \ v_1, A_{v_1}, \ \ldots, \ v_K, A_{v_K}]. \tag{20}$$

3.4 Informative Priors

The joint action space in MVDPDPSR is often exponentially large because of multiple vehicles, requests, and stations, which prevents reinforcement learning from exploring useful actions. To address this, we designed manually-computed informative priors for vehicle selection and destination selection. By multiplying the probabilities output by the Decoder with informative priors, we guide the agent to prioritize actions with higher potential rewards.

Informative Priors for Vehicle Selection In a multi-vehicle scenario, the balance of vehicle load is crucial for overall delivery efficiency and resource utilization. Based on this, we design a load-balancing informative prior, which is inversely proportional to the current vehicle load. The smaller the current load of a vehicle, the higher its informative prior weight in vehicle selection, thereby guiding the assignment of requests toward load balancing:

$$P^{\text{pri}}(A_{r_m} = v_k) = \frac{spa_k}{cap_k}. \tag{21}$$

Although temporarily deferring request assignments may yield positive benefits (e.g., when the current vehicle load is high), we aim to avoid excessive occurrences of such cases, as this would severely affect the exploration efficiency during reinforcement learning. Therefore, we set a small probability for action τ (defined in Eq. (3)):

$$P^{\text{pri}}(A_{r_m} = \tau) = \beta. \tag{22}$$

The vehicle selection probability after informative priors' guidance is:

$$P(A_{r_m} = v_k) = P^{\text{dec}}(A_{r_m} = v_k) \cdot P^{\text{pri}}(A_{r_m} = v_k). \tag{23}$$

Informative Priors for Destination Selection For the destination station selection of vehicle v_k, the informative prior $P^{\text{pri}}(A_{v_k} = n_i)$ is determined as follows:

- If station n_i has requests to be delivered, $P^{\text{pri}}(A_{v_k} = n_i) = 1$.
- If station n_i has requests to be picked up, $P^{\text{pri}}(A_{v_k} = n_i) = \frac{0.1 \times \overline{\boldsymbol{E}}}{e_{to_k, n_i}}$, where $\overline{\boldsymbol{E}}$ denotes the average value of all elements in the distance matrix $\boldsymbol{E}$. This formulation prioritizes stations that are closer in distance. Since the number of stations with pickup requests is significantly larger than those with delivery requests, we use a smaller coefficient (0.1) to balance the vehicle's priorities between pickup and delivery tasks.
- For stations n_i that do not fall into the above two categories, $P^{\text{pri}}(A_{v_k} = n_i) = 0$.

The destination selection probability after informative priors' guidance is:

$$P(A_{v_k} = n_i) = P^{\text{dec}}(A_{v_k} = n_i) \cdot P^{\text{pri}}(A_{v_k} = n_i). \tag{24}$$

3.5 Optimization via PPO

We use Proximal Policy Optimization (Schulman et al. 2017) to train MAPT. Our value function is defined as $V(s_t) = \text{MLP}(\boldsymbol{o_g})$, where MLP transforms global information into a scalar. We use Generalized Advantage Estimation (Schulman et al. 2015) to balance the bias and variance of advantage estimation:

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l (rwd_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l})). \tag{25}$$

The Actor loss is defined as

$$\rho_t(\theta) = \frac{\pi_\theta(A^t|s_t)}{\pi_{\theta_{\text{old}}}(A^t|s_t)}, \tag{26}$$

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \Big[\min \big(\rho_t(\theta) \hat{A}_t^{\text{GAE}}, \ \text{CLIP}(\rho_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t^{\text{GAE}} \big) \Big], \tag{27}$$

where $\pi_\theta(A^t|s_t)$ is the joint probability defined in Eq. (15). The Critic loss is defined as

$$\mathcal{L}^{\text{Critic}} = \left(V(s_t) - (r_{t+1} + \gamma V(s_{t+1})) \right)^2. \tag{28}$$

The final total loss is

$$\mathcal{L} = \mathcal{L}^{\text{CLIP}} + \alpha \mathcal{L}^{\text{Critic}}. \tag{29}$$

4 Experiments

4.1 Experimental Scenarios

To validate the effectiveness of the MAPT framework, we conducted experiments on 8 datasets, including both synthetic and real-world datasets, for comprehensive evaluation. A brief description of the datasets is provided below.

- *Synthetic Dataset*: We generated synthetic datasets with different scales of stations $I \in \{20, 50, 300\}$, corresponding to dataset suffixes `-S`, `-L`, and `-XL`, respectively. For these datasets, the number of requests and vehicles were

Scenario	synth-S		synth-S-cost		synth-L		synth-L-cost		dhrd-tpe		dhrd-sg		dhrd-se		synth-XL	
Metric	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑
OR Tools	182.1	0.54	117.4	0.52	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	55.3	0.45	N/A	N/A
SA	162.5	0.52	108.1	0.52	641.1	0.28	339.3	0.29	244.9	0.31	136.7	0.30	50.4	0.41	1531.9	0.31
GA	108.7	0.30	58.3	0.29	329.4	0.14	89.5	0.12	64.1	0.08	40.9	0.13	21.3	0.17	863.8	0.17
SA*	133.0	0.42	81.2	0.42	563.1	0.25	284.8	0.24	223.3	0.28	139.6	0.30	56.6	0.46	1650.9	0.34
GA*	75.2	0.21	39.4	0.19	219.9	0.09	88.0	0.09	53.8	0.07	39.0	0.13	18.4	0.15	543.2	0.11
MAPDP*	174.9	0.53	82.0	0.49	177.6	0.08	65.2	0.08	287.5	0.36	247.2	0.35	72.9	0.56	807.0	0.16
PARCO*	178.8	0.53	86.6	0.52	177.6	0.08	65.2	0.08	391.2	0.49	267.5	0.46	64.2	0.49	1834.7	0.36
Nearest	189.2	0.57	124.1	0.57	962.6	0.41	592.3	0.41	309.2	0.39	194.5	0.40	39.8	0.32	2641.1	0.53
MAPDP	157.5	0.41	70.0	0.42	958.1	0.37	385.2	0.37	135.4	0.17	64.9	0.18	19.3	0.16	2543.1	0.50
MAPT	**275.2**	**0.83**	**192.1**	**0.84**	**1875.1**	**0.80**	**1348.6**	**0.81**	**697.2**	**0.87**	**376.1**	**0.71**	**78.9**	**0.64**	**3227.5**	**0.65**

Table 2: Overall performance comparison on synthetic datasets and real-world datasets. The best result in each column is bolded. "N/A" indicates that the algorithm cannot provide results within an acceptable time frame.

Scenario	synth-S		synth-S-cost		synth-L		synth-L-cost		dhrd-tpe		dhrd-sg		dhrd-se		synth-XL	
Metric	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑	Obj.↑	Comp.↑
MAPT	**275.2**	**0.83**	**192.1**	**0.84**	**1875.1**	**0.80**	**1348.6**	**0.81**	**697.2**	**0.87**	**376.1**	**0.71**	**78.9**	**0.64**	**3227.5**	**0.65**
w/o Rel	270.9	0.82	190.7	0.83	1869.9	0.80	1338.4	0.81	692.1	0.87	365.3	0.69	77.1	0.63	3136.8	0.63
w/o AR	217.2	0.65	171.1	0.77	1604.0	0.68	1074.5	0.69	611.4	0.76	258.1	0.50	49.4	0.40	1905.8	0.38
w/o Priors	175.2	0.54	92.7	0.54	1089.5	0.47	529.1	0.48	511.6	0.64	250.9	0.49	56.6	0.46	1375.0	0.27

Table 3: Results of ablation studies. The best result in each column is bolded.

set as $(M, K) = (110, 5)$, $(550, 15)$, and $(550, 50)$, respectively. Vehicle capacity was fixed at 3. Travel cost per unit distance is set to 0 or 0.3, with the latter case marked by an additional suffix `-cost`. Requests were uniformly sampled from stations, with appearance times sampled from $\mathrm{Uniform}(1, T)$, where T was set to 58 or 128. The profit for each request was the distance between its origin and destination.

- *Real-World Dataset*: We used the DHRD dataset (Assylbekov et al. 2023), which contains food delivery requests from three cities: Taipei (`tpe`), Singapore (`sg`), and Stockholm (`se`). Each city was treated as an independent dataset with $I = 36$ stations. The corresponding numbers of requests and vehicles were $(M, K) = (800, 20)$, $(700, 15)$, and $(200, 3)$ for the three cities, respectively. Vehicle capacity was set to 6. The dataset was divided into 76 days for training/validation and 14 days for testing.

4.2 Performance Evaluation

Baselines We evaluate our MAPT against several baselines, including Rolling-Horizon algorithms, static algorithms, rule-based algorithms, and MDP-based algorithms.

- **Rolling-Horizon Algorithms:** The Rolling-Horizon paradigm addresses dynamic problems by decomposing them into static subproblems within a sliding time window. For solving these static subproblems, we employ **OR Tools**, **Simulated Annealing (SA)**, and **Genetic Algorithm (GA)**.
- **Static Algorithms:** These methods assume complete foreknowledge of all requests (though unrealistic), solving the problem statically without Rolling-Horizon. **MAPDP*** (Zong et al. 2022) is a MARL approach with Encoder-Decoder for static problems, adapted to handle distance matrix inputs. **PARCO*** (Berto et al. 2024) is a Transformer-based RL framework with parallel decision-making, modified to support distance matrix inputs. **SA*/GA*** are static versions of our Rolling-Horizon SA and GA with full request information.
- **Rule-Based Algorithms:** These methods use predefined rules for scheduling decisions. **Nearest** is a greedy approach that always selects the closest available request for pickup or delivery.
- **MDP-Based Algorithms:** At each step of the MDP, we solve the problem statically using **MAPDP** (Zong et al. 2022) as described in the Static Algorithms section, and only execute the first step of the solution to ensure applicability to dynamic problems.

Overall Comparison The performance results on synthetic and real-world datasets are summarized in Table 2, using the objective (Obj.) and request completion rate (Comp.) as metrics (higher is better). MAPT significantly outperforms all baselines across all datasets. MAPT also surpasses the state-of-the-art MAPDP, which fails to model multi-vehicle joint probabilities and lacks our informative priors. As can be seen from Table 4, the decision time of MAPT is significantly shorter than that of exact and meta-heuristic algorithms.

Scenario	synth-S	synth-S-cost	synth-L	synth-L-cost	synth-XL
OR Tools	401.727	459.143	N/A	N/A	N/A
SA	23.684	23.255	166.660	164.102	613.920
GA	16.654	16.448	103.147	99.864	354.789
Nearest	0.059	0.060	0.117	0.119	0.148
MAPDP	0.794	0.827	31.981	32.544	33.217
MAPT	1.420	1.416	8.300	7.888	27.434

Table 4: Comparison of inference times on the synthetic dataset (in seconds), which is the average running time of each instance in the dataset.

4.3 Ablation Studies

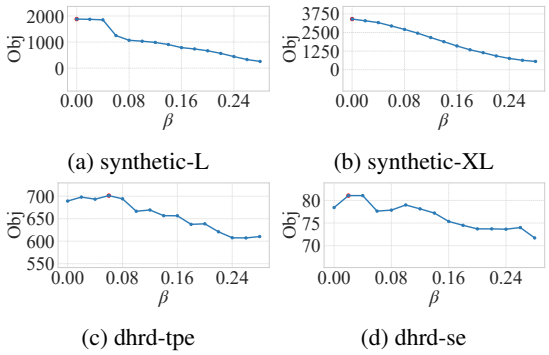
We conducted ablation studies on all datasets to validate the effectiveness of key components:

- **w/o Relation-Aware Attention (Rel)**: Replaces the Relation-Aware Attention module with the original attention mechanism.
- **w/o AutoRegressive Decoding (AR)**: Decodes each action independently and employs a conflict handler to resolve conflicts where request assignments exceed vehicle capacity.
- **w/o Informative Priors (Priors)**: Removes informative priors guidance for vehicle and destination selection.

The results are shown in Table 3. As can be seen from the table, AutoRegressive Decoding and Informative Priors significantly improve performance across all datasets, and Relation-Aware Attention also contributes to some extent. The ablation experiments demonstrate that all three proposed components enhance the final performance, validating their effectiveness.

4.4 Sensitivity Analysis

We conduct a sensitivity analysis on the hyperparameter β as defined in Eq. (22). We vary β from 0 to 0.3 in increments of 0.02 and observe the resulting changes in the final objective. The results on the 4 representative datasets are illustrated in Figure 2. As shown in the figure, when β is small, the performance is better and more stable, while larger values of β lead to a gradual decline in performance. However, in the dhrd-tpe and dhrd-se datasets, using a small but non-zero β improves performance. This indicates that slightly deferring request allocation can be beneficial in these cases.



(a) synthetic-L (b) synthetic-XL

(c) dhrd-tpe (d) dhrd-se

Figure 2: Sensitivity analysis of β across datasets.

5 Related Work

5.1 Dynamic Pickup and Delivery Problems

Dynamic Pickup and Delivery Problems have been studied through exact, heuristic, metaheuristic, and learning-based methods (Cai et al. 2023). Exact methods such as MILP can yield optimal results but are restricted to small instances (Liu et al. 2018; Savelsbergh and Sol 1998). Heuristic methods (Fikar 2018; Fikar, Hirsch, and Gronalt 2018; Andersson 2021) are efficient for large-scale dynamic problems but cannot ensure optimality. Metaheuristics (Schilde, Doerner, and Hartl 2011; Tirado and Hvattum 2017; Novaes et al. 2015) improve scalability but still struggle in highly dynamic settings. Reinforcement learning methods have been explored (Li et al. 2021; Yu et al. 2025), but they mainly focus on single-vehicle operations. Hybrid approaches first learn edge costs using neural networks and then apply traditional optimization techniques (Jiang et al. 2023b; Wang, Wu, and Zhao 2021; Wang et al. 2019; Wang, Lin, and Li 2025; Wu et al. 2019). Other studies have addressed ride-on-demand problems with dynamic pricing (Guo et al. 2019, 2020, 2023), as well as dynamic metro scheduling (Wang et al. 2022). There are also entity representation learning methods that use GNNs, hyper-GNNs, contrastive learning, transfer learning, or LLM enhancement and can be applied to spatio-temporal system optimization (Wu et al. 2020; Yang et al. 2025; Han et al. 2025; Jiang et al. 2023a; Li et al. 2025; Cheng et al. 2025; Zhang et al. 2024).

5.2 RL-based Methods for Vehicle Routing Problems

Reinforcement Learning has been applied to both single and multiple Vehicle Routing Problems. For single-vehicle problems, attention-based models (Vinyals, Fortunato, and Jaitly 2015; Bello et al. 2016; Nazari et al. 2018; Kool, Van Hoof, and Welling 2018) and GNN-based variants (Fellek et al. 2023; Heydaribeni et al. 2023) have been proposed. For multi-vehicle problems, multi-agent RL methods (Zhang et al. 2020; Zong et al. 2022; Zhang, Qi, and Guan 2023) have been developed but mainly for static settings. Recent RL studies for dynamic settings (Anuar et al. 2022; Xiang et al. 2024) have emerged, but they simplify the original problem by introducing additional assumptions.

6 Conclusion

This paper presents MAPT to solve the cooperative Multi-Vehicle Dynamic Pickup and Delivery Problem with Stochastic Requests. MAPT utilizes the Transformer's Encoder-Decoder architecture combining with Pointer Network to generate joint action sequences in an AutoRegressive manner. By incorporating the Relation-Aware Attention module and informative priors, the framework achieves a better performance. Experiments show that MAPT outperforms existing baselines in solution quality across synthetic datasets and real-world datasets.

Acknowledgments

This work is supported by the National Key R&D Program of China (2023YFC3304700). Prof. Jingyuan Wang's work was partially supported by the National Natural Science Foundation of China (No. 72222022, 72171013) and the Fundamental Research Funds for the Central Universities (JKF-2025017226182). Prof. Junjie Wu's work was partially supported by the National Natural Science Foundation of China (72242101, 72031001), the Outstanding Young Scientist Program of Beijing Universities (JWZQ20240201002), and the Artificial Intelligence Technology R&D Center for Exploration and Development of China National Petroleum Corporation (2024-KFKT-22).

References

Andersson, T. 2021. A Comparative Study on a Dynamic Pickup and Delivery Problem: Improving routing and order assignment in same-day courier operations.

Anuar, W. K.; Lee, L. S.; Seow, H.-V.; and Pickl, S. 2022. A multi-depot dynamic vehicle routing problem with stochastic road capacity: An MDP model and dynamic policy for post-decision state rollout algorithm in reinforcement learning. *Mathematics*, 10(15): 2699.

Assylbekov, Y.; Bali, R.; Bovard, L.; and Klaue, C. 2023. Delivery Hero Recommendation Dataset: A Novel Dataset for Benchmarking Recommendation Algorithms. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1042–1044.

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

Berto, F.; Hua, C.; Luttmann, L.; Son, J.; Park, J.; Ahn, K.; Kwon, C.; Xie, L.; and Park, J. 2024. PARCO: Learning Parallel Autoregressive Policies for Efficient Multi-Agent Combinatorial Optimization. *arXiv preprint arXiv:2409.03811*.

Cai, J.; Zhu, Q.; Lin, Q.; Li, J.; Chen, J.; and Ming, Z. 2022. An efficient multi-objective evolutionary algorithm for a practical dynamic pickup and delivery problem. In *International conference on intelligent computing*, 27–40. Springer.

Cai, J.; Zhu, Q.; Lin, Q.; Ma, L.; Li, J.; and Ming, Z. 2023. A survey of dynamic pickup and delivery problems. *Neurocomputing*, 126631.

Cheng, J.; Wang, J.; Zhang, Y.; Ji, J.; Zhu, Y.; Zhang, Z.; and Zhao, X. 2025. Poi-enhancer: An llm-based semantic enhancement framework for poi representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, 11509–11517.

Fellek, G.; Farid, A.; Gebreyesus, G.; Fujimura, S.; and Yoshie, O. 2023. Deep Graph Representation Learning to Solve Vehicle Routing Problem. In *2023 International Conference on Machine Learning and Cybernetics (ICMLC)*, 172–180. IEEE.

Fikar, C. 2018. A decision support system to investigate food losses in e-grocery deliveries. *Computers & Industrial Engineering*, 117: 282–290.

Fikar, C.; Hirsch, P.; and Gronalt, M. 2018. A decision support system to investigate dynamic last-mile distribution facilitating cargo-bikes. *International Journal of Logistics Research and Applications*, 21(3): 300–317.

Geiser, T.; Hanne, T.; and Dornberger, R. 2020. Best-match in a set of single-vehicle dynamic pickup and delivery problem using ant colony optimization. In *Proceedings of the 2020 the 3rd International Conference on Computers in Management and Business*, 126–131.

Guo, S.; Chen, C.; Wang, J.; Ding, Y.; Liu, Y.; Xu, K.; Yu, Z.; and Zhang, D. 2020. A force-directed approach to seeking route recommendation in ride-on-demand service using multi-source urban data. *IEEE Transactions on Mobile Computing*, 21(6): 1909–1926.

Guo, S.; Chen, C.; Wang, J.; Liu, Y.; Xu, K.; Yu, Z.; Zhang, D.; and Chiu, D. M. 2019. Rod-revenue: Seeking strategies analysis and revenue prediction in ride-on-demand service using multi-source urban data. *IEEE Transactions on Mobile Computing*, 19(9): 2202–2220.

Guo, S.; Shen, Q.; Liu, Z.; Chen, C.; Chen, C.; Wang, J.; Li, Z.; and Xu, K. 2023. Seeking based on dynamic prices: Higher earnings and better strategies in ride-on-demand services. *IEEE Transactions on Intelligent Transportation Systems*, 24(5): 5527–5542.

Han, C.; Wang, J.; Wang, Y.; Yu, X.; Lin, H.; Li, C.; and Wu, J. 2025. Bridging traffic state and trajectory for dynamic road network and trajectory representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 11763–11771.

Heydaribeni, N.; Zhan, X.; Zhang, R.; Eliassi-Rad, T.; and Koushanfar, F. 2023. HypOp: Distributed Constrained Combinatorial Optimization leveraging Hypergraph Neural Networks. *arXiv preprint arXiv:2311.09375*.

Jiang, J.; Pan, D.; Ren, H.; Jiang, X.; Li, C.; and Wang, J. 2023a. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *2023 IEEE 39th international conference on data engineering (ICDE)*, 843–855. IEEE.

Jiang, W.; Zhao, W. X.; Wang, J.; and Jiang, J. 2023b. Continuous trajectory generation based on two-stage GAN. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 4374–4382.

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

Li, X.; Luo, W.; Yuan, M.; Wang, J.; Lu, J.; Wang, J.; Lü, J.; and Zeng, J. 2021. Learning to optimize industry-scale dynamic pickup and delivery problems. In *2021 IEEE 37th international conference on data engineering (ICDE)*, 2511–2522. IEEE.

Li, Y.; Wang, J.; Yu, X.; Wang, P.; and Huang, Q. 2025. Cross City Traffic Flow Generation via Retrieval Augmented Diffusion Model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Liu, S.; Tan, P. H.; Kurniawan, E.; Zhang, P.; and Sun, S. 2018. Dynamic scheduling for pickup and delivery with time

windows. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 767–770. IEEE.

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

Nazari, M.; Oroojlooy, A.; Snyder, L.; and Takac, M. 2018. 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.

Novaes, A. G.; Bez, E. T.; Burin, P. J.; and Aragão Jr, D. P. 2015. Dynamic milk-run OEM operations in over-congested traffic conditions. *Computers & Industrial Engineering*, 88: 326–340.

Savelsbergh, M.; and Sol, M. 1998. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4): 474–490.

Schilde, M.; Doerner, K. F.; and Hartl, R. F. 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & operations research*, 38(12): 1719–1730.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sheridan, P. K.; Gluck, E.; Guan, Q.; Pickles, T.; Balcıog, B.; Benhabib, B.; et al. 2013. The dynamic nearest neighbor policy for the multi-vehicle pick-up and delivery problem. *Transportation Research Part A: Policy and Practice*, 49: 178–194.

Tirado, G.; and Hvattum, L. M. 2017. Improved solutions to dynamic and stochastic maritime pick-up and delivery problems using local search. *Annals of Operations Research*, 253: 825–843.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. 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.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *Advances in neural information processing systems*, 28.

Wang, J.; Lin, Y.; and Li, Y. 2025. GTG: Generalizable Trajectory Generation Model for Urban Mobility. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 834–842.

Wang, J.; Wu, N.; and Zhao, W. X. 2021. Personalized route recommendation with neural network enhanced search algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 34(12): 5910–5924.

Wang, J.; Wu, N.; Zhao, W. X.; Peng, F.; and Lin, X. 2019. Empowering A* search algorithms with neural networks for personalized route recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 539–547.

Wang, Z.; Pan, Z.; Chen, S.; Ji, S.; Yi, X.; Zhang, J.; Wang, J.; Gong, Z.; Li, T.; and Zheng, Y. 2022. Shortening passengers' travel time: A dynamic metro train scheduling approach using deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(5): 5282–5295.

Wen, M.; Kuba, J.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; and Yang, Y. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 16509–16521. Curran Associates, Inc.

Wu, N.; Wang, J.; Zhao, W. X.; and Jin, Y. 2019. Learning to effectively estimate the travel time for fastest route recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1923–1932.

Wu, N.; Zhao, X. W.; Wang, J.; and Pan, D. 2020. Learning effective road network representation with hierarchical graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 6–14.

Xiang, C.; Wu, Z.; Tu, J.; and Huang, J. 2024. Centralized Deep Reinforcement Learning Method for Dynamic Multi-Vehicle Pickup and Delivery Problem With Crowdshippers. *IEEE Transactions on Intelligent Transportation Systems*.

Yang, Y.; Wang, J.; Yu, X.; and Tang, Y. 2025. HygMap: Representing All Types of Map Entities via Heterogeneous Hypergraph. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 9438–9446.

Yu, X.; Wang, J.; Yang, Y.; Huang, Q.; and Qu, K. 2025. BIGCity: A universal spatiotemporal model for unified trajectory and traffic state data analysis. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 4455–4469. IEEE.

Zhang, K.; He, F.; Zhang, Z.; Lin, X.; and Li, M. 2020. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 121: 102861.

Zhang, W.; Wang, J.; Yang, Y.; et al. 2024. VecCity: A taxonomy-guided library for map entity representation learning. *arXiv preprint arXiv:2411.00874*.

Zhang, Z.; Qi, G.; and Guan, W. 2023. Coordinated multi-agent hierarchical deep reinforcement learning to solve multi-trip vehicle routing problems with soft time windows. *IET Intelligent Transport Systems*, 17(10): 2034–2051.

Zong, Z.; Zheng, M.; Li, Y.; and Jin, D. 2022. Mapdp: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 9980–9988.