Towards Efficient Ridesharing via Order-Vehicle Pre-Matching Using Attention Mechanism

Zhidan Liu[†], Jinye Lin[§], Zhiyu Xia[§], Chao Chen[‡], Kaishun Wu[†]
[†]*Hong Kong University of Science and Technology (Guangzhou),* [§]*Shenzhen University,* [‡]*Chongqing University*
Emails: {zhidanliu, wuks}@hkust-gz.edu.cn, {linjinye2021, xiazhiyu2022}@email.szu.edu.cn, cschaochen@cqu.edu.cn

***Abstract*—Dynamic ridesharing has garnered significant attention in recent years due to its numerous benefits. Existing ridesharing algorithms often employ a "*filter-and-refine*" framework, where a large set of candidate vehicles is initially selected for each ride order, followed by computationally intensive route planning for each candidate. However, this process can lead to significant response delays and limit system efficiency. To address this challenge, we propose an order-vehicle pre-matching recommendation approach (`PreMR`) that refines the candidate set before route planning. `PreMR` leverages spatial-temporal intervals and a self-attention mechanism to encode diverse order and vehicle information into uniform and informative representations, enabling it to accurately identify the most suitable vehicles for each order. Extensive experiments using real-world datasets and four representative ridesharing algorithms demonstrate that `PreMR` significantly reduces order response time (by 46.78% on average) while maintaining high service quality, with a slight trade-off in the order completion rate.**

***Index Terms*—Ridesharing, Order-vehicle pre-matching, Self-attention mechanism, Spatio-temporal**

I. Introduction

Online ridesharing platforms such as DiDi Chuxing [1] and Uber [4] facilitate the sharing of a single vehicle among multiple riders with similar itineraries and time schedules. As a promising mode of transportation, ridesharing not only significantly reduces travel costs for riders compared to traditional taxi services but also effectively complements public transportation by providing additional capacity, particularly during rush hours and in areas with inadequate transit services [16]. With the continuous increase in urban mobility demands, ridesharing has emerged as a critical strategy for mitigating traffic congestion and enhancing transportation efficiency.

Different from static carpooling, which addresses pre-known travel demands [8], dynamic ridesharing is characterized by the unpredictable nature of riders' departure locations and request times. Consequently, a central challenge for ridesharing platforms is the immediate and irreversible assignments of ride orders to suitable vehicles. Existing research efforts focused on optimizing order-vehicle matching in dynamic ridesharing typically operate within the "*filter-and-refine*" framework, which involves two distinct stages. In the *filtering* stage, a set of candidate vehicles is initially selected for each incoming ride order r based on r's departure location and the current distribution of available vehicles. In the *refinement* stage, each candidate vehicle is then subjected to route planning in order to determine the most suitable vehicle for fulfilling order r. The majority of research efforts have concentrated on either enhancing the filtering stage by minimizing the number of candidate vehicles [12], [16], [26] or optimizing route planning algorithms to improve the system's response speed [28], [31], [34]. Within the "*filter-and-refine*" framework, the generation of accurate yet concise candidate sets is paramount. This is because all candidate vehicles must undergo a rigorous route planning investigation, which constitutes the most computationally intensive and time-consuming operation in dynamic ridesharing systems, particularly when dealing with large road networks. The larger the number of candidate vehicles for an order, the longer it takes to process that request.

TABLE I: Number of candidates generated by three algorithms under various vehicle quantities (average of 1000 orders).

# of all vehicles	2000	2500	3000	5000	10000
# of candidates by *mTshare*	115	155	255	416	984
# of candidates by *Prophet*	176	216	259	414	822
# of candidates by *pGreedyDP*	209	269	319	524	1044

Despite this, current ridesharing algorithms often generate relatively large candidate sets, leading to prolonged response times and diminished system efficiency. Table I presents statistics on the sizes of candidate sets for three representative ridesharing algorithms, namely *mTshare* [16], *Prophet* [27], and *pGreedyDP* [28], across different vehicle quantities using a real-world dataset (Please refer to Section V-A for details on the dataset and experimental settings). Table I reveals that all algorithms produce a substantial number of candidate vehicles for each order, and the size of the candidate set increases significantly with an increase in the number of vehicles.

Motivated by the aforementioned observation, in this paper we introduce an order-vehicle *pre-M*atching *R*ecommendation (`PreMR`) approach. At a high level, `PreMR` takes as input the information of a ride order r and its corresponding candidate set $\mathcal{C}_r$, and recommends the top κ candidate vehicles from $\mathcal{C}_r$ as the most suitable candidates for serving r. By providing a refined candidate set for each order, `PreMR` can be integrated into any ridesharing algorithms based on the "*filter-and-refine*" framework, enhancing their computational efficiency without compromising service quality.

Despite its potential advantages, developing an effective order-vehicle pre-matching approach faces at least two challenges: *(1) Multi-modal data representation*: Ride orders and candidate vehicles encompass diverse attributes, including location, time, sequences of pick-up and drop-off events, and travel routes, which are presented in various data modalities. Encoding this multi-modal data into a uniform and informative

2024 IEEE International Conference on Data Mining (ICDM) | 979-8-3315-0668-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDM59182.2024.00033

2374-8486/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDM59182.2024.00033

representation poses a significant challenge. *(2) Learning matching decision patterns*: Existing ridesharing algorithms employ diverse strategies for matching orders and vehicles, resulting in complex matching decision patterns. Learning and simulating this decision process is crucial for identifying suitable vehicles for an order, but it presents a formidable task.

To address the challenges, we design and implement PreMR. We employ embedding techniques to transform the location and time information of orders and vehicles into uniform representations. Furthermore, for each candidate vehicle w, we introduce a spatial-temporal relation matrix constructed using the geographic and temporal intervals between pick-up and drop-off events within w's schedule. This matrix is then incorporated into the self-attention mechanism to further enhance the representations of orders and vehicles. The rationale behind is that when inserting an order into a vehicle's schedule, the time and distance constraints are primarily determined by these spatial-temporal intervals. Based on these updated representations, we compute the matching score between order r and each candidate vehicle w, which represents the matching suitability between r and w. After calculating the matching scores for all candidates, PreMR returns the top κ candidates with the highest scores as recommendations for serving order r. To train the PreMR model, we conduct ridesharing simulations using a given ridesharing algorithm $\mathcal{A}$ and collect all execution details as training data. This allows PreMR to learn the matching strategy employed by algorithm $\mathcal{A}$.

This paper makes the following main contributions:

- We formally define and formulate the problem of order-vehicle pre-matching in dynamic ridesharing. The study of this problem has the potential to improve the performance of "*filter-and-refine*" based ridesharing algorithms.
- We introduce PreMR, a novel solution to the pre-matching problem. By leveraging spatial-temporal intervals and a self-attention mechanism, PreMR effectively refines the candidate sets for orders, thereby enhancing the efficiency of ridesharing algorithms.
- We integrate PreMR with four representative ridesharing algorithms and conduct extensive experiments using real-world datasets. The results show that PreMR significantly reduces order response time, achieving an average reduction of 46.78% while maintaining high service quality, with slight trade-off in order completion rate.

The rest of this paper is organized as follows. We discuss related works in Section II. The problem statement and solution overview are presented in Section III. We elaborate and evaluate the design of PreMR in Section IV and Section V, respectively. Finally, Section VI concludes this paper.

II. Related Work

Ridesharing. Ridesharing has emerged as a promising solution for sustainable and efficient urban transportation in recent years [23]. Existing works can be broadly categorized into two modes based on the timing of ride order processing: *static ridesharing* and *dynamic ridesharing*. Unlike static ridesharing , also known as carpooling [8], which involves pre-known orders and pre-planned travel routes, dynamic ridesharing aligns with the requirements of smart mobility by enabling real-time matching riders' requests to vehicles are en route [30], [32]. However, due to its NP-hard nature, existing solutions to the dynamic ridesharing problem generally adopt the "*filter-and-refine*" framework, primarily optimizing two stages: *candidate vehicle search* [12], [16], [17], [21], [22], [26], [35] and *vehicle routing planning* [7], [27], [28], [31], [34]. For example, Ma *et al*. [21], [22] introduce *Tshare* system, which utilizes spatio-temporal indexing to search for candidate vehicles and minimizes additional driving costs when inserting trips into vehicle's schedule. In addition, Liu *et al*. [16], [17] propose the mobility-aware ridesharing algorithm called *mTshare*, aiming to efficiently serve both online and offline orders by leveraging historical mobility data to optimize order-vehicle matching and address the limitations of previous algorithms by considering geographic locations and travel directions. Hailem *et al*. [7] introduce a model for dynamic order assignments and route planning, which considers travel demands, pricing, and vehicle locations to generate optimal routes. To improve the refinement efficiency, Tong *et al*. [27], [28], [34] propose several dynamic programming based insertion operators, *i.e.*, *pGreedyDP* [28] and *Prophet* [27], and new equivalent optimization objectives to reduce the time complexity of order insertions.

These works, however, still face huge computational overheads due to the extensive investigations of a large number of candidate vehicles for each order. To address this challenge, we introduce PreMR, which can refine candidate vehicle sets, thereby accelerating the order-vehicle matching and improving the efficiency of ridesharing systems.

Attention mechanism. Attention mechanisms empower models to selectively focus on specific parts of the input data, enabling them to dynamically weigh different elements or features [5]. This powerful capability has led to widespread adoption of attention mechanisms across diverse domains [14]. Researchers have also leveraged attention mechanisms to address tasks in intelligent transportation, *e.g.*, travel demand prediction [33] and vehicle trajectory prediction [13].

Recently, the *Transformer* network [29] has gained significant recognition for its substantial improvements in temporal modeling through the use of *self-attention* mechanisms. By processing dependencies within a sequence, this mechanism can learn sequence patterns and intrinsic correlations. Numerous works have successfully applied self-attention mechanisms to analyze spatial and mobility data. For instance, Jiang *et al*. [10] propose a framework to learn feature representations for trajectories based on graph attention networks. *MTrajRec* [24] utilizes an attention module to account for global information of trajectories, enhancing trajectory recovery.

There are a few works that apply attention mechanisms for optimizing ridesharing systems. For instance, Shen *et al*. [25] propose a gated attention recurrent network for ridesharing order prediction. Additionally, Huang *et al*. [9] present a ride-hailing demand prediction method that combines multi-head spatial attention and bidirectional attention mechanisms. Our

work distinguishes itself by being the first to leverage the self-attention mechanism to score the matchings between an order and its candidate vehicles. This approach enables PreMR to recommend the most suitable candidate vehicles, thereby reducing computational costs involved in the refinement stage.

III. Problem Statement and Solution Overview

In this section, we introduce some definitions and problem statement of order-vehicle pre-matching in dynamic ridesharing, and then present the overview of our PreMR solution.

A. Notations and Definitions

***Definition** 1:* **(Road Network)** *A road network is denoted by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ represents a geo-location (e.g., road intersection), and each edge $(u, v) \in \mathcal{E}$ represents a road segment, which is associated with a weight $cost(u, v)$, indicating the travel cost from vertex u to vertex v on the graph $\mathcal{G}$.*

Noting that the travel cost can be measured by either travel distance or travel time. Since they can be easily converted to each other when the vehicle's travel speed is known, we do not differentiate between them, but use travel cost consistently.

***Definition** 2:* **(Ride Order)** *A ride order is denoted by $r = \langle o_r, t_r^o, d_r, t_r^d \rangle$, where $o_r \in \mathcal{V}$ and $d_r \in \mathcal{V}$ represent the origin and destination of a trip, respectively. In addition, t_r^o indicates when the riders release order r, and this order should be completed before a deadline time t_r^d by delivering the riders from origin o_r to destination d_r.*

The deadline t_r^d can be estimated from the trip start time and travel cost between the origin and destination, complemented with a detour cost. Following the typical setting in previous works [16], [27], [28], we set the deadline t_r^d as

$$t_r^d = t_r^o + cost(o_r, d_r) \times \rho, \tag{1}$$

where $cost(o_r, d_r)$ estimates the travel cost for order r, and ρ is the *flexible factor* that indicates the acceptable travel cost by riders compared to the shortest path [6].

The ridesharing platform continuously receives ride orders, generates a candidate vehicle set for each order r, and then determines the most suitable one to serve r.

***Definition** 3:* **(Candidate Vehicle Set)** *A candidate vehicle set $\mathcal{C}_r$ for order r contains a set of available vehicles that can serve r given the information of both r and all vehicles.*

In principle, each vehicle w can simultaneously serve at most c_w riders, and should pick-up and deliver assigned orders following the planned schedule.

***Definition** 4:* **(Ridesharing Vehicle)** *A ridesharing vehicle $w = \langle o_w, c_w \rangle$ has an initial location o_w and a capacity c_w.*

***Definition** 5:* **(Ridesharing Event)** *A ridesharing event s represents a combination of location ℓ and time t, indicating the occurrence of a pick-up or drop-off of riders at a specific location ℓ during a particular time t.*

***Definition** 6:* **(Vehicle Schedule)** *A valid schedule $\mathcal{S}_w = (s_1, s_2, \cdots, s_m)$ for ridesharing vehicle w is a sequence of events, where each event corresponds to pick-up or drop-off the riders of an order at designated location and time, e.g., o_r or d_r of an order r, and o_r should appear ahead of d_r.*

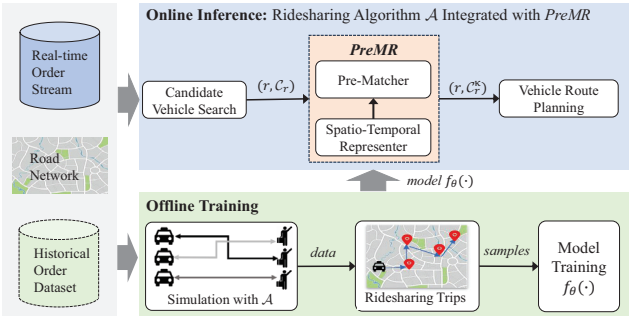


Fig. 1: Ridesharing framework integrated with PreMR.

B. Problem Statement

***Definition** 7:* **(Problem of Order-Vehicle Pre-Matching in Dynamic Ridesharing)** *Given a road network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and historical ridesharing travel orders and trajectories, order-vehicle pre-matching aims to explore spatio-temporal relationship among the ride orders and candidate vehicles from massive historical data to derive an abstract function $f_\theta(\cdot)$ for optimizing online dynamic ridesharing. Specifically, for each order r and its corresponding candidate vehicle set $\mathcal{C}_r$, $f_\theta(\cdot)$ can return a list of κ ranked candidate vehicles, which is formally expressed as:*

$$\mathcal{C}_r^\kappa \leftarrow f_\theta(r, \mathcal{C}_r), \tag{2}$$

where $\mathcal{C}_r^\kappa$ contains κ candidate vehicles selected from $\mathcal{C}_r$ that are the most suitable to serve r in the ridesharing scenario.

A solution to the order-vehicle pre-matching problem, such as our PreMR, can be integrated with any existing "*filter-and-refine*" framework based ridesharing algorithms, and significantly enhance their computational efficiency by providing accurate and reduced candidate vehicle sets.

Challenges. Despite the promising advantages, realizing a solution like PreMR faces two non-trivial challenges.

(1) **Multi-faceted matching criteria.** The matching process between orders and vehicles in existing ridesharing algorithms considers a wide range of factors. On one hand, factors associated with each ride order r, including the riders' pick-up and drop-off locations, estimated travel cost, and the time of order placement, play a crucial role in order-vehicle matching. On the other hand, the instantaneous location, and the planned schedule and travel route of each candidate vehicle are equally pivotal. Consequently, the expected function $f_\theta(\cdot)$ must effectively capture the intricate relationships among these factors by jointly characterizing this information as input.

(2) **Learning diverse matching strategies.** Considering the dynamic mobility patterns of both riders and ridesharing vehicles within a city, existing ridesharing algorithms typically employ diverse strategies, such as heuristics or optimization theories, to match orders with suitable vehicles. Therefore, the function $f_\theta(\cdot)$ must learn and simulate the decision process underlying these strategies, given the information of the order and candidate vehicles. This learning process is crucial for investigating their matching suitability and enabling efficient candidate recommendations.

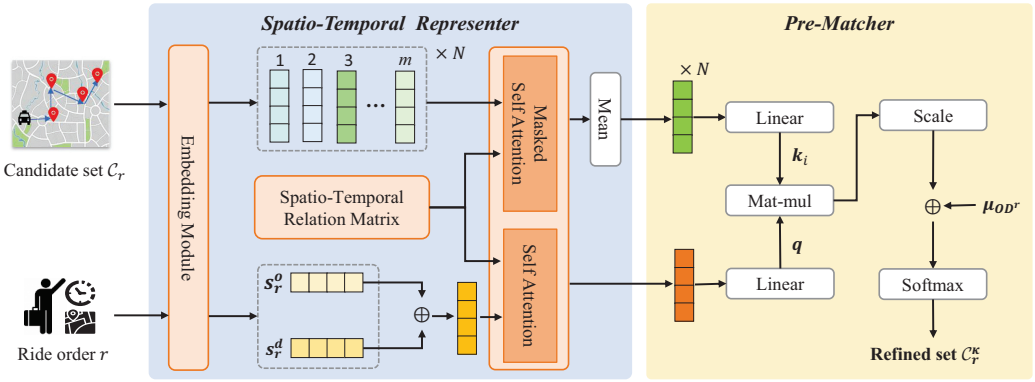


Fig. 2: The architecture of PreMR that consists of two major components, *i.e.*, *Spatio-Temporal Representer* and *Pre-Matcher*.

C. Solution Overview

To address these challenges, we present PreMR as a solution that realizes the functionality of $f_\theta(\cdot)$. PreMR can be integrated with any "*filter-and-refine*" paradigm based ridesharing algorithm $\mathcal{A}$, enhancing the efficiency by optimizing its initial candidate vehicle sets. Figure 1 presents the framework of a representative ridesharing algorithm $\mathcal{A}$ integrated with PreMR, consisting of two stages: *offline training* and *online inference*.

Offline training. To learn and capture algorithm $\mathcal{A}$'s order-vehicle matching strategies, we collect a substantial amount of training data generated during the execution of $\mathcal{A}$ in a ridesharing scenario. To this end, we conduct ridesharing simulation by running algorithm $\mathcal{A}$ with historical orders and a road network $\mathcal{G}$. During the simulation, for each ride order r, we execute $\mathcal{A}$ to process r and record all intermediate data and ridesharing trip data. This includes the order information, r's candidate set $\mathcal{C}_r$, the vehicle w assigned to serve r, and w's original schedule $\mathcal{S}_w$. All these execution details are stored, and subsequently used as samples to train PreMR, allowing it to emulate algorithm $\mathcal{A}$'s strategies for matching orders with suitable vehicles. Notably, both the ridesharing simulations and model training are performed offline, with no impact on the performance of online inference.

Online inference. In this stage, algorithm $\mathcal{A}$ collaborates with PreMR to process real-time orders appearing on the road network $\mathcal{G}$. Following the "*filter-and-refine*" paradigm, for each incoming ride order r, an initial candidate set $\mathcal{C}_r$ is generated by $\mathcal{A}$. Taking both order r and candidate set $\mathcal{C}_r$ as input, PreMR evaluates each possible matching (r, w), where candidate vehicle $w \in \mathcal{C}_r$, and outputs a matching score. Based on the scores for all candidates in set $\mathcal{C}_r$, a refined candidate set $\mathcal{C}_r^\kappa$ containing the top κ scoring candidates is fed into the *vehicle routing planning* component, which then determines the most suitable vehicle to serve order r.

IV. Design of PreMR

Figure 2 depicts the architecture of PreMR, comprising two key components, namely *spatio-temporal representer* and the *pre-matcher*. Given a ride order r and its corresponding initial candidate set $\mathcal{C}_r$ of N vehicles, provided by algorithm $\mathcal{A}$, the *spatio-temporal representer* component encodes the information of order r and candidate vehicles $\mathcal{C}_r$ into separate representations. Subsequently, the *pre-matcher* component takes these representations as input and recommends a ranked set of κ candidate vehicles for order r. Next we elaborate on the design of each component.

A. Spatio-Temporal Representer

As illustrated in Figure 2, spatio-temporal representer comprises three modules, *i.e.*, the *embedding module*, the *spatio-temporal relation matrix*, and the *self-attention layer*.

Embedding module. We design a multi-modal embedding module to learn latent representations for candidate vehicles and ride orders. To reduce computational costs and enhance representational capacity, the embedding module maps scalars to low-dimensional and dense vectors. Specifically, this module separately encodes the information of candidate vehicle ID, ride order ID, location, and time into basic representations, denoted as $\mathbf{v}^{wid} \in \mathbb{R}^n$, $\mathbf{v}^{rid} \in \mathbb{R}^n$, $\mathbf{v}^{\ell} \in \mathbb{R}^n$, and $\mathbf{v}^{t} \in \mathbb{R}^n$, respectively, where n represents the embedding dimension. The embeddings of candidate vehicles and ride orders are then composed of these basic representations.

• *Embedding for candidate vehicles.* In the ridesharing scenario, each candidate vehicle $w \in \mathcal{C}_r$ is primarily characterized by its schedule $\mathcal{S}_w$, which consists of multiple pick-up and drop-off events. The output of embedding module for each event s in schedule $\mathcal{S}_w$ is the sum of three basic representations, *i.e.*, $\mathbf{v}^s = \mathbf{v}^{wid} + \mathbf{v}^{\ell} + \mathbf{v}^{t} \in \mathbb{R}^n$, where ℓ and t are the location and time involved in event s. Therefore, the embedding for candidate vehicle w with m events, *i.e.*, $\mathcal{S}_w = (s_1, s_2, \cdots, s_m)$, is $E(\mathcal{S}_w) = [\mathbf{v}^{s_1}; \mathbf{v}^{s_2}; \cdots; \mathbf{v}^{s_m}] \in \mathbb{R}^{m \times n}$.

Considering the variable sequence lengths of candidate vehicles' schedules, we set the maximum sequence length to $m_{max} = 2 \times c_{max} + 1$, padding shorter sequences with *zeros* until they reach m_{max}. Noting that c_{max} represents the maximum capacity of all vehicles, the factor of 2 accounts for each order having pick-up and drop-off events, and the addition of 1 accounts for current location of the candidate vehicle. Consequently, the embedding for all candidate vehicles will be adjusted to $E(\mathcal{S}_w) \in \mathbb{R}^{m_{max} \times n}$.

• *Embedding for ride orders.* An order r contains one pick-up event s_r^o and one drop-off event s_r^d, each of which is encoded by the embedding module in the form of $\mathbf{v}^{rid}+\mathbf{v}^{\ell}+\mathbf{v}^{t} \in$

$\mathbb{R}^n$. We obtain $\mathbf{v}^{s_r^o}$ and $\mathbf{v}^{s_r^d}$ for the two events respectively. To derive the order's representation, we execute element-wise addition for these two embeddings as $\mathbf{v}^r = \mathbf{v}^{s_r^o} + \mathbf{v}^{s_r^d}$. We use $E(r) = \mathbf{v}^r \in \mathbb{R}^n$ to denote order r's representation.

Spatial-temporal relation matrix. To capture the spatio-temporal characteristics of orders that could share a vehicle, we construct a spatio-temporal relation matrix by exploiting the spatio-temporal intervals among events within the schedule of each candidate vehicle.

***Definition** 8:* **(Spatio-Temporal Interval)** *The spatio-temporal interval between the i-th and j-th events is denoted by st_{ij}, which consists of the geography interval Δd_{ij} and time interval Δt_{ij}.*

We model the relationship between any two events within a vehicle schedule, such as $(s_i, s_j) \in \mathcal{S}_w$, using their spatio-temporal interval st_{ij}. Specifically, we define $st_{ij} = \Delta d_{ij} + \Delta t_{ij}$, where Δt_{ij} represents the time difference between event s_i and s_j, and $\Delta d_{ij} = \texttt{Haversine}\left(\ell_{s_i}, \ell_{s_j}\right)$ that calculates the shortest distance between two locations on a sphere. Given vehicle schedule $\mathcal{S}_w$ with m events, we thus build the spatio-temporal relation matrix $\mathbf{ST}_w$ for vehicle w as follows:

$$\mathbf{ST}_w = \begin{bmatrix} st_{11} & st_{12} & \cdots & st_{1m} \\ st_{21} & st_{22} & \cdots & st_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ st_{m1} & st_{m2} & \cdots & st_{mm} \end{bmatrix}. \tag{3}$$

We will also generate embeddings for these spatio-temporal intervals. To mitigate data sparsity issue while enhancing model generalization capability [11], we first process the time and geography intervals using a normalization method, similar to [15], which divides the time and geography intervals into discrete bins. Based on the pre-defined upper and lower bounds for time and geography intervals, we then perform linear interpolation calculations for each type of intervals separately. These calculations are expressed as follows:

$$\begin{aligned} E\left(\Delta t_{ij}\right) &= \frac{\left[E(U(\Delta t))\left(U(\Delta t) - \Delta t_{ij}\right) + E(L(\Delta t))\left(\Delta t_{ij} - L(\Delta t)\right)\right]}{\left[\left(U(\Delta t) - \Delta t_{ij}\right) + \left(\Delta t_{ij} - L(\Delta t)\right)\right]}, \\ E\left(\Delta d_{ij}\right) &= \frac{\left[E(U(\Delta d))\left(U(\Delta d) - \Delta d_{ij}\right) + E(L(\Delta d))\left(\Delta d_{ij} - L(\Delta d)\right)\right]}{\left[\left(U(\Delta d) - \Delta d_{ij}\right) + \left(\Delta d_{ij} - L(\Delta d)\right)\right]}, \end{aligned} \tag{4}$$

where $U(\Delta t)$ and $L(\Delta t)$ denote the upper and lower bounds of time differences, and $U(\Delta d)$ and $L(\Delta d)$ denote the upper and lower bounds of the geographical distances. In addition, $E(\cdot)$ represents the embedding for a given interval. Based on the pre-processing on time and geography intervals, we obtain $E(st_{ij}) = E(\Delta d_{ij}) + E(\Delta t_{ij}) \in \mathbb{R}^n$. Therefore, we have $E(\mathbf{ST}_w) \in \mathbb{R}^{m \times m \times n}$. Finally, we average the spatio-temporal relation matrix representation along the last dimension to reduce the dimensionality from $m \times m \times n$ to $m \times m$, that is:

$$E(\mathbf{ST}_w) = \texttt{mean}(E(\mathbf{ST}_w)) \in \mathbb{R}^{m \times m}. \tag{5}$$

Through the averaging operation, we not only integrate feature information along the depth dimension, but also make dimensional structure for integration into the attention layer.

In the ridesharing scenario, the essential route planning involves calculating the temporal and spatial distances between the pick-up and drop-off locations/times of an order r and the event sequence of candidate vehicle w's schedule. Taking this consideration into account, we model the spatial-temporal intervals between vehicle w's events and order r's pick-up and drop-off events as their spatio-temporal relations. To this end, we separately multiply $\mathbf{v}^{s_r^o}$ and $\mathbf{v}^{s_r^d}$ with the embedding $E(s_i)$ of each event s_i in vehicle schedule $\mathcal{S}_w$. As a result, we obtain $E(\mathbf{O}^r_{\mathbf{ST}_w}) \in \mathbb{R}^{m \times n}$ and $E(\mathbf{D}^r_{\mathbf{ST}_w}) \in \mathbb{R}^{m \times n}$, which represent the spatial-temporal interval matrices of r's pick-up and drop-off events with respect to w's schedule, respectively. By summing these two matrices, we derive the overall spatio-temporal interval matrix for the matching (r, w), that is

$$E(\mathbf{OD}^r_{\mathbf{ST}_w}) = E(\mathbf{O}^r_{\mathbf{ST}_w}) + E(\mathbf{D}^r_{\mathbf{ST}_w}) \in \mathbb{R}^{m \times n}. \tag{6}$$

Furthermore, we compute the mean of this matrix to obtain an average spatial-temporal relation between r and w, *i.e.*,

$$\mu_{\mathbf{OD}^r} = \texttt{mean}(\texttt{mean}(E(\mathbf{OD}^r_{\mathbf{ST}_w}))) \in \mathbb{R}. \tag{7}$$

By performing the averaging operation, we effectively reduce the dimension of the matrix $E(\mathbf{OD}^r_{\mathbf{ST}_w})$, summarizing the spatio-temporal relationship between order r and vehicle w into a single scalar value. This not only simplifies the data but also retains essential information, making it more suitable for subsequent processing within the attention layer.

Interval-aware attention layer. Inspired by the successful applications of self-attention networks in various domains [5], we propose an extension to the self-attention mechanism that enhances the representations of both candidate vehicles and orders by considering the spatio-temporal intervals. As shown in Figure 2, we incorporate the spatio-temporal relation matrix into an interval-aware attention layer to capture the proximity between events of the vehicle schedules, thereby updating the latent representations of both candidate vehicles and orders. In particular, we design a *masked self-attention module* for enhancing candidate vehicle representation, while applying a traditional *self-attention module* for order representation.

• *Masked self-attention module.* To achieve self-association between events in a vehicle schedule, this module treats each event as a simultaneous input for the query, key, and value vectors. Additionally, we explicitly integrate the spatio-temporal relation matrix into the module, ensuring that the context of each event includes explicit spatio-temporal positioning cues. Consequently, this module takes the vehicle representation $E(\mathcal{S}_w)$ and the spatio-temporal relation matrix $E(\mathbf{ST}_w)$ as input, and produces the updated vehicle representation as output. To this end, we project $E(\mathcal{S}_w)$ into a query matrix $\mathbf{Q}_w$, a key matrix $\mathbf{K}_w$, and a value matrix $\mathbf{V}_w$ through three distinct parameter matrices $\mathbf{W}_{Q_w}, \mathbf{W}_{K_w}, \mathbf{W}_{V_w} \in \mathbb{R}^{n \times n}$, as follows:

$$\mathbf{Q}_w = E(\mathcal{S}_w)\mathbf{W}_{Q_w}, \mathbf{K}_w = E(\mathcal{S}_w)\mathbf{W}_{K_w}, \mathbf{V}_w = E(\mathcal{S}_w)\mathbf{W}_{V_w}, \tag{8}$$

where $\mathbf{Q}_w, \mathbf{K}_w, \mathbf{V}_w \in \mathbb{R}^{m \times n}$. The linear transformations enhance the model's expressive capacity for vehicle schedules. When dealing with schedule sequences of varying lengths, we employ *zero* padding to standardize all sequences to the same length. Consequently, it becomes necessary to construct a

mask matrix to ensure the model disregards padded positions, which do not contain valid information. To address this, we construct a mask matrix $\mathbf{M}_w \in \mathbb{R}^{m\times m}$, with the upper-left portion is an identity matrix corresponding to the original length of the schedule $\mathcal{S}_w$, and the remaining elements are *zeros*. The masked self-attention module then combines the attention map with the spatio-temporal relation matrix $\mathbf{ST}_w$ through element-wise addition as follows:

$$\mathbf{A}(\mathcal{S}_w) = \text{Softmax}((\frac{\mathbf{Q}_w\mathbf{K}_w^T}{\sqrt{n}} + E(\mathbf{ST}_w))\mathbf{M}_w)\mathbf{V}_w, \quad (9)$$

where $\frac{\mathbf{Q}_w{\mathbf{K}_w}^T}{\sqrt{n}} \in \mathbb{R}^{m\times m}$ represents the attention map, and $\mathbf{A}(\mathcal{S}_w) \in \mathbb{R}^{m\times n}$ is the attention result. Subsequently, we perform the mean pooling on $\mathbf{A}(\mathcal{S}_w)$ to derive the updated vehicle representation as:

$$\mathbf{A}^{'}(\mathcal{S}_w) = \texttt{mean}(\mathbf{A}(\mathcal{S}_w)). \quad (10)$$

• *Self-attention module.* We also update the representation of order r through self-attention operations to consider spatio-temporal relations between the pick-up and drop-off events. Specifically, this module takes the order representation $E(r)$ and order r's spatio-temporal intervals as input, and produces the updated order representation as follows:

$$\mathbf{A}^{'}(r) = \text{Softmax}\left(\frac{E(r)\mathbf{W}_{Q_r}(E(r)\mathbf{W}_{K_r})^T}{\sqrt{n}} + E(\Delta)\right)E(r)\mathbf{W}_{V_r}, \quad (11)$$

where $\mathbf{W}_{Q_r}, \mathbf{W}_{K_r}, \mathbf{W}_{V_r} \in \mathbb{R}^{n\times n}$ are the input projection matrices for a query, key, and value, respectively. In addition, $\Delta = \Delta d_{od} + \Delta t_{od}$ represents the spatio-temporal interval between the pick-up and drop-off events of order r, and $E(\Delta) \in \mathbb{R}^n$ is its corresponding representation.

By applying these attention operations, we obtain $\mathbf{A}^{'}(\mathcal{S}_w) \in \mathbb{R}^n$ and $\mathbf{A}^{'}(r) \in \mathbb{R}^n$, which are the updated representations for candidate vehicle w and order r, respectively.

B. Pre-Matcher

Based on the updated representations of order r and each candidate vehicle in $\mathcal{C}_r$, the *pre-matcher* component evaluates the suitability of each candidate vehicle for serving order r and recommends the top κ candidates for further investigations in the route planning stage.

Given the updated representation $\mathbf{A}^{'}(r)$ for order r, and $\mathbf{A}(\mathcal{C}_r) = \{\mathbf{A}^{'}(\mathcal{S}_{w_1}), \mathbf{A}^{'}(\mathcal{S}_{w_2}), \cdots, \mathbf{A}^{'}(\mathcal{S}_{w_N})\}$, where $N = |\mathcal{C}_r|$, as the set of updated representations for candidate vehicles in $\mathcal{C}_r$, we employ scaled dot-product attention to calculate the *matching score* of each candidate vehicle $w_i \in \mathcal{C}_r$ and r by mapping order representation $\mathbf{A}^{'}(r)$ and vehicle representation $\mathbf{A}^{'}(\mathcal{S}_{w_i})$ into a common attention space through the parameter matrices $\mathbf{W}_{Q_{rw}} \in \mathbb{R}^{n\times n}$ and $\mathbf{W}_{K_{rw}} \in \mathbb{R}^{n\times n}$, that is:

$$\begin{aligned} \mathbf{q} &= \mathbf{A}^{'}(r)\mathbf{W}_{Q_{rw}}, \\ \mathbf{k}_i &= \mathbf{A}^{'}(\mathcal{S}_{w_i})\mathbf{W}_{K_{rw}}, \text{where } w_i \in \mathcal{C}_r. \end{aligned} \quad (12)$$

Subsequently, we compute the matching score z_i between the query vector $\mathbf{q}$ and each key vector $\mathbf{k}_i$ as

$$z_i = \frac{\mathbf{q}\mathbf{k}_i^T}{\sqrt{n}} + \mu_{\mathbf{OD}^r}, \quad (13)$$

where $\mu_{\mathbf{OD}^r}$ is calculated by Eq. (7), indicating the average spatio-temporal difference between order r and vehicle w_i.

Next, we normalize the matching scores using the *Softmax* function to obtain a probability distribution $\{y_i\}_{i=1}^N$ as follows:

$$y_i = \frac{\exp(z_i)}{\sum_{j=1}^N \exp(z_j)}, \quad (14)$$

where each y_i represents the probability that the i-th candidate vehicle w_i in $\mathcal{C}_r$ is selected for serving order r.

After calculating the matching probabilities for all candidate vehicles of order r, `PreMR` ranks these candidates in descending order based on their matching probabilities. It then selects the top-κ candidates, denoted as the set $\mathcal{C}_r^\kappa$, as the refined candidate set for serving order r.

Training. Recall that we conduct ridesharing simulations with a ridesharing algorithm $\mathcal{A}$ to generate the training data. The simulation process records each ride order's details, including the order information, the available candidate vehicle set, the final vehicle selection, and the vehicle schedule and route. For a given order, the best-matched vehicle suggested by $\mathcal{A}$ is treated as a positive sample, while the unselected candidate vehicles form the negative samples. We train `PreMR` to identify the most suitable candidate vehicle by minimizing the cross-entropy loss between the predicted matching probabilities and the actual order-vehicle matching results. The loss is calculated as follows:

$$\textit{loss} = -\sum_{r\in\mathbb{O}}\left[\log(y_b) + \sum_{i=1,j\neq b}^N \log(1-y_i)\right], \quad (15)$$

where $\mathbb{O}$ is the set of ride orders in the historical trip dataset, b is the label of the best-matched vehicle for order r, and y_i is the matching probability predicted by `PreMR` model. By minimizing this loss, the model learns to distinguish the candidate vehicles most likely to serve the orders, thereby reducing candidate vehicles and improving the system efficiency.

V. Performance Evaluation

A. Experimental Setup

Dataset. We conduct data-driven experiments to validate our `PreMR` using a real-world anonymized ridesharing dataset published under the GAIA Project by DiDi Chuxing [2]. This dataset comprises 7065907 ridesharing transaction records collected in November 2016 within the downtown area of Chengdu city, China. Each transaction record includes order information such as vehicle ID, order ID, pick-up location/time, drop-off location/time, and the time when the order was generated. The location information in these records is represented by latitude and longitude coordinates.

Since the Didi dataset only contains information about these successfully served orders, unserved orders are not captured. To simulate realistic ridesharing services and augment the experimental data, we employ the method used in [19], [22] to generate additional synthetic orders that follow the distribution patterns and travel trends of orders in Didi dataset. To comprehensively evaluate the performance of `PreMR` under different settings of vehicles and orders, we adopt a growth factor α to

synthesize new orders. Specifically, we use different α values according to the total number of ridesharing vehicles. When the number of vehicles is set to 1000, 2000, and 3000, we set $\alpha = 1.0$; for 5000 vehicles, we set $\alpha = 1.5$; and for 10000 vehicles, we set $\alpha = 2.0$.

Consequently, we employ a hybrid data strategy, incorporating both synthetic data and the original Didi dataset, for the experiments. We reserve the data of the last week for testing, which compares a variety of ridesharing algorithms, while the remaining data are used for the ridesharing simulations that generate training data to train the PreMR model.

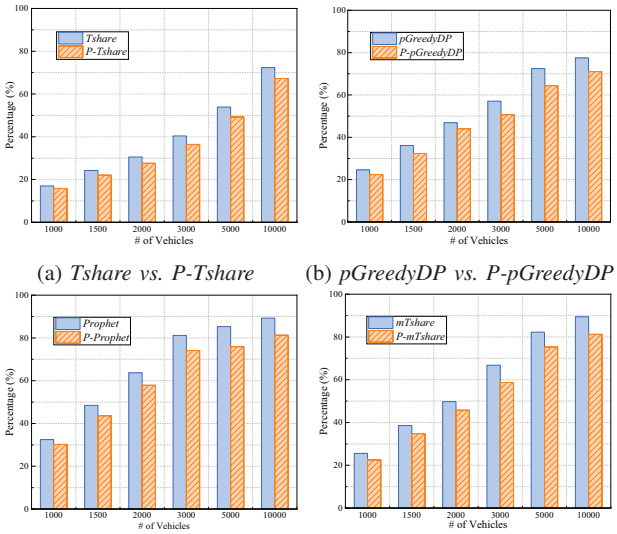
We construct the road network model $\mathcal{G}$ by acquiring road network data of Chengdu city from OpenStreetMap [3]. After data cleaning and pre-processing, the road network model $\mathcal{G}$ consists of 214440 vertices and 466330 edges. Additionally, we divide the road network into 72 uniform grids for indexing both orders and vehicles.

Baselines. We evaluate the effectiveness and performance of PreMR by integrating it with the following four representative "*filter-and-refine*" based ridesharing algorithms.

- *Tshare* [21], [22]: This algorithm proposes a bidirectional search method that filters candidate vehicles for order r by considering r's origin/destination and vehicle schedules with a search radius γ. Upon finding a match, the system promptly returns the result.
- *pGreedyDP* [28]: This algorithm identifies candidate vehicles within a search radius γ around the origin of order r. It utilizes dynamic programming for the strategic insertion of r into the vehicle schedule, thereby improving computation efficiency of the refinement stage.
- *Prophet* [27]: This algorithm integrates both real-time orders and future orders, which are predicted by analyzing historical data, into route planning. It further enhances the order insertion operation to improve system efficiency.
- *mTshare* [16], [17]: This algorithm utilizes dual-layer partitions and mobility clusters to index orders and vehicles. Thus, it can effectively filter candidates for order r through geographical locations and travel directions. It optimizes the route planing by leveraging the partitions.

For each baseline algorithm $\mathcal{A}$, we integrate it with PreMR, and the derived algorithm is denoted as *P-*$\mathcal{A}$. Consequently, we have four variant algorithms, namely *P-Tshare*, *P-pGreedyDP*, *P-Prophet*, and *P-mTshare*.

Performance metrics. We assess all algorithms using four widely used performance metrics, *i.e.*, *order completion rate*, *average response time*, *average waiting time*, and *average detour time*. Specifically, the order completion rate refers to the ratio of the number of served orders to the total number of orders. The response time is the time taken to process an order. The waiting time is the interval between the release time and pick-up time of an order, and the detour time is the additional time added compared to serving an order without ridesharing. In addition, we evaluate the prediction performance of the PreMR model using the top-κ accuracy, denoted as $acc@\kappa$, which refers to the percentage of testing cases where the best matching vehicle is among the top κ recommendations.



(a) *Tshare vs. P-Tshare* (b) *pGreedyDP vs. P-pGreedyDP*

(c) *Prophet vs. P-Prophet* (d) *mTshare vs. P-mTshare*

Fig. 3: The order completion rates of different algorithms.

Implementation. We implement PreMR and all baselines using Python. For each ride order r, its origin and destination are mapped to the nearest vertices in graph $\mathcal{G}$. We use the original release time of r, and set its deadline time t_r^d following Eq. (1). By default, we set flexible factor $\rho = 1.3$ to simulate a realistic ridesharing scenario. The initial location o_w for vehicle w is randomly assigned as any vertex in graph $\mathcal{G}$, and the capacity c_w is set to be 4. Similar to previous works [16], [22], [28], we fix the traveling speed of all vehicles at $15\,km/h$. Although we assume stable traffic conditions, our method can easily extend to run with real-time traffic conditions if such information can be derived from transportation agencies or be inferred by advanced traffic estimation methods [18], [20]. Additionally, we fix the search radius $\gamma = 2.5\,km$, equivalent to a maximum waiting time of 10 minutes.

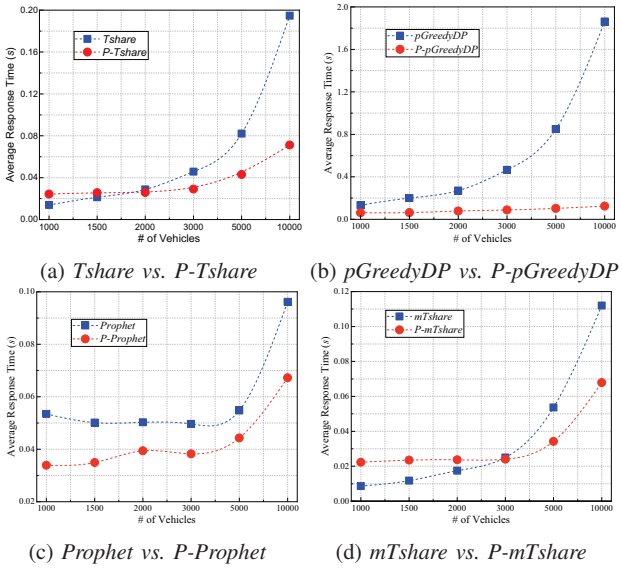
To accelerate route planning, we precompute and store in memory the shortest paths between any two vertices in graph $\mathcal{G}$. Consequently the travel cost between any two vertices can be easily retrieved by all algorithms. To train the PreMR model, we use the Adam optimizer with a learning rate of 0.001, with a training cycle of 64 epochs. By default, we set the embedding dimension n as 64 and set $\kappa = 35$, which implies that at most 35 candidate vehicles are recommended for each order.

All experiments are conducted on a server equipped with an Intel Core i9-9900K CPU@3.60GHz, NAVIDA GeForce RTX 2080 Ti GPU, and 32GB of memory. Each configuration is repeated 10 times, and the average results are reported.

B. *Effectiveness of* PreMR

We validate the effectiveness of PreMR by comparing the performance between each baseline $\mathcal{A}$ and its variant *P-*$\mathcal{A}$ that integrates with PreMR under different quantities of vehicles.

Comparisons of order completion rate. We report the order completion rates of different algorithms under various amount of vehicles in Figure 3. From these results, we observe



(a) *Tshare vs. P-Tshare* (b) *pGreedyDP vs. P-pGreedyDP*

(c) *Prophet vs. P-Prophet* (d) *mTshare vs. P-mTshare*

Fig. 4: The average response time of different algorithms.

two key trends. Firstly, with an increase in the number of vehicles, the number of orders served by each algorithm also increases. We observe that *Prophet* and *mTshare* consistently outperform *Tshare* and *pGreedyDP* by completing more orders with a given amount of available vehicles. Secondly, the variants generally serve slightly fewer orders than their respective original algorithms. This is because PreMR attempts to learn the order-vehicle matching strategy of each algorithm $\mathcal{A}$ and then accelerates the overall matching process (see the results in Figure 4 later). In fact, we find that PreMR actually performs quite well by simulating the matching patterns of each original algorithm. The order completion rate gap between each original algorithm $\mathcal{A}$ and its variant *P-*$\mathcal{A}$ is quite small. Simply put, each variant *P-*$\mathcal{A}$ can successfully fulfill about 90.78% of the orders served by the original algorithm $\mathcal{A}$.

Comparisons of response time. We vary the total number of vehicles and compare the response time of all algorithms. From Figure 4, we observe that PreMR accelerates the response speed of baselines more significantly in cases with a larger number of vehicles. This is attributed to the increased candidate sets for orders due to more available vehicles, leading to a substantial increase in the time taken to find the optimal vehicle. It is noteworthy that for all ridesharing algorithms, larger candidate sizes pose greater challenges, highlighting the advantage of utilizing PreMR. We also find that *P-Tshare* and *P-mTshare* respond a bit slower than the original *Tshare* and *mTshare* in the case of $\leq$ 2000 vehicles and $\leq$ 3000 vehicles, respectively. This is because these two algorithms have simple heuristic filtering rules that work well with a small number of vehicles, while PreMR requires considerable time to execute the deep learning model for inferences. However, with the increase in available vehicles, the variants integrated with PreMR respond to requests much faster than the baselines. For the cases of $\geq$ 3000 vehicles, PreMR helps to reduce the response time by 46.78% on average across all baselines.

TABLE II: Comparisons of the total running time (in *minutes*) required by *pGreedyDP* and its variant *P-pGreedyDP* for processing different amounts of orders.

Amount of orders	*pGreedyDP*	*P-pGreedyDP*
3 *hours*	324.32	38.03
6 *hours*	893.23	87.75
12 *hours*	1337.54	173.57
18 *hours*	$>$ 2 *days*	250.66
24 *hours*	$>$ 3 *days*	376.57

The acceleration in order response is even more pronounced for *pGreedyDP*. In particular, PreMR makes *P-pGreedyDP* 10 times faster than the original *pGreedyDP*. To better understand PreMR's advantage, we conduct experiments to compare the overall running time of *pGreedyDP* and *P-pGreedyDP* on processing orders collected within different hours. The results are presented in Table II, where more hours indicate more orders to be processed. For the order amounts of 3-6 hours, *P-pGreedyDP* runs nearly an order of magnitude faster than *pGreedyDP*. When the order amounts increase to 18 and 24 hours, *pGreedyDP* cannot process the orders within a reasonable time, while *P-pGreedyDP* can still effectively handle such a large amount of orders. It only takes 376.57 minutes for *P-pGreedyDP* to process orders of one day, demonstrating a clear advantage of PreMR in handling large-scale orders.

Comparisons of waiting time and detour time. Table III presents the average waiting time of all algorithms under various vehicle quantities. As the number of vehicles increases, the average waiting time for all algorithms generally decreases. This trend can be attributed to the increased availability of vehicles, allowing both original and variant algorithms to find nearby matches for each order efficiently, thereby reducing riders' waiting time. Table IV compares the average detour time of all algorithms. We observe similar detour performance between each original algorithm and its variant. Since PreMR is mainly designed to refine candidate sets without altering matching rules, it does not affect the performance of ridesharing algorithms in terms of waiting time and detour time.

These experimental results demonstrate that PreMR can significantly improve the response speed of existing ridesharing algorithms, while maintaining the high service quality, with a slight trade-off in the order completion rate.

C. Detailed Study

We conduct additional experiments to study the impacts of the spatio-temporal relation matrix and embedding dimension by employing *mTshare* and its variant *P-mTshare*.

Impact of spatio-temporal relation matrix. To enhance the representation of vehicle schedules and orders, we incorporate spatio-temporal intervals. We perform ablation experiments to explore the design effectiveness. Specifically, we remove the spatial information (*i.e.*, geography interval Δd) and temporal information (*i.e.*, time interval Δt) from the spatial-temporal relation matrix. This results in two variants, denoted as PreMR *-w/o S.* and PreMR *-w/o T.*, respectively. Additionally, we remove all spatial-temporal information to obtain the variant PreMR *-w/o ST.*. We evaluate these three

TABLE III: Comparisons of the average waiting time (in *minutes*) for all algorithms, where Δ_w represents the difference in average waiting time between algorithms *P-*$\mathcal{A}$ and $\mathcal{A}$. A blue Δ_w indicates performance improvement achieved by PreMR.

# vehicles	*Tshare*	*P-Tshare*	Δ_w	*pGreedyDP*	*P-pGreedyDP*	Δ_w	*Prophet*	*P-Prophet*	Δ_w	*mTshare*	*P-mTshare*	Δ_w
1000	2.34	2.29	-0.05	3.90	3.89	-0.01	3.95	3.91	-0.04	3.85	3.93	0.08
1500	2.23	2.19	-0.04	3.59	3.66	0.07	3.63	3.70	0.07	3.55	3.94	0.39
2000	2.11	2.09	-0.02	3.22	3.27	0.05	3.25	3.21	-0.03	3.20	3.13	-0.06
3000	1.84	1.81	-0.03	2.54	2.59	0.04	2.51	2.50	-0.01	2.58	2.38	-0.20
5000	1.56	1.53	-0.03	1.78	1.77	-0.01	1.82	1.80	-0.02	1.75	1.82	0.07
10000	0.95	0.94	-0.01	1.10	1.11	0.01	1.32	1.36	0.04	0.87	0.84	-0.03

TABLE IV: Comparisons of the average detour time (in *minutes*) for all algorithms, where Δ_d represents the difference in average detour time between algorithms *P-*$\mathcal{A}$ and $\mathcal{A}$. A blue Δ_d indicates performance improvement achieved by PreMR.

# vehicles	*Tshare*	*P-Tshare*	Δ_d	*pGreedyDP*	*P-pGreedyDP*	Δ_d	*Prophet*	*P-Prophet*	Δ_d	*mTshare*	*P-mTshare*	Δ_d
1000	0.53	0.51	-0.02	0.80	0.79	-0.01	1.07	1.08	0.01	0.54	0.54	0.00
1500	0.48	0.46	-0.02	0.88	0.87	-0.01	1.16	1.15	-0.01	0.49	0.46	-0.03
2000	0.59	0.56	-0.03	0.95	0.95	0.00	1.27	1.25	-0.02	0.59	0.58	-0.01
3000	0.36	0.35	-0.01	1.11	1.10	-0.01	1.48	1.47	-0.01	0.37	0.38	0.01
5000	0.30	0.29	-0.01	1.29	1.27	-0.02	1.71	1.73	0.02	0.30	0.31	0.01
10000	0.27	0.26	-0.01	1.51	1.40	-0.11	1.88	1.86	-0.02	0.27	0.27	0.00

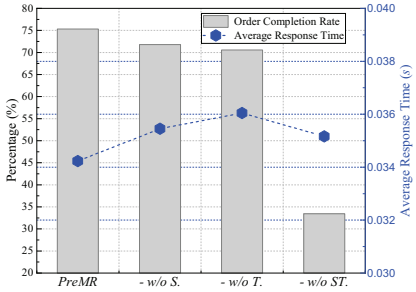


Fig. 5: Impact of spatial and temporal intervals on ridesharing performance.

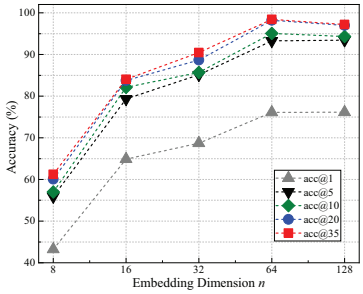


Fig. 6: Model accuracy under different embedding dimensions.

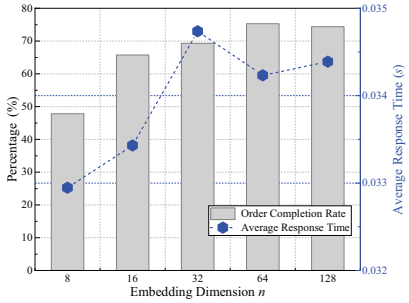


Fig. 7: Impact of embedding dimensions on ridesharing performance.

TABLE V: Impact of spatial and temporal intervals on prediction accuracy of the PreMR model.

Model	$acc@1$	$acc@5$	$acc@10$	$acc@20$	$acc@35$
PreMR	76.11%	93.26%	95.04%	98.21%	98.43%
– w/o S.	69.45%	89.46%	91.99%	94.26%	95.86%
– w/o T.	53.59%	72.99%	79.39%	87.32%	94.19%
– w/o ST.	1.41%	7.27%	14.50%	28.55%	47.81%

variant models along with the complete PreMR model using the $acc@\kappa$ metric and other ridesharing metrics when integrating each of them into *mTshare*.

Table V presents the prediction accuracy results for PreMR model and its three variants. We find that the absence of time intervals has a more significant impact than the absence of spatial intervals, as evidenced by a larger drop in accuracy compared to the complete model. This is because, in the ridesharing scenario, time factors – such as estimated travel time and delivery deadline – play a crucial role in determining the matching between orders and vehicles. Additionally, Table V indicates that spatial-temporal intervals are extremely important for representation learning. Without the spatial-temporal intervals, PreMR *-w/o ST.* exhibits a sharp decrease in accuracy, *e.g.*, the $acc@1$ metric drops from 76.11% to 1.41%, and the $acc@35$ metric drops from 98.43% to 47.81%. Ridesharing services are subject to strict spatio-temporal constraints, such as riders' expected departure and arrival times and the acceptable detour costs during the trip. Consequently, these spatio-temporal constraints directly impact the feasibility assessment of candidate vehicles. Therefore, explicitly representing temporal and spatial information can significantly enhance model performance.

Figure 5 shows the experimental results for the metrics of order completion rate and average response time for *mTshare* and its variants. When omitting the spatial and temporal intervals, there are approximately 3.54% and 4.74% reductions in the order completion rate for the variants compared to the complete model, respectively. The order completion rate even decreases to 33.43% when both intervals are excluded. This substantial decline underscores the critical role of the spatial-temporal relation matrix in the PreMR model. Additionally, Figure 5 also shows that the absence of spatial or temporal intervals affects the response time, though the impact is slight.

Impact of embedding dimension. We investigate the impact of embedding dimension n on PreMR's performance by varying n from 8 to 128 and studying its effects on prediction accuracy and ridesharing performance. The $acc@\kappa$ results in Figure 6 reveal that increasing the embedding dimension n significantly improves the prediction performance of the PreMR model. Notably, when the embedding dimension is

increased from 8 to 16, there is a substantial enhancement across different $acc@\kappa$ metrics. This is because a larger embedding dimension can capture more useful information. However, as the embedding dimension expands further beyond 64, the accuracy tends to stabilize. Therefore, we find that an embedding dimension of $n = 64$ is sufficient to capture the necessary information from ride orders, vehicle schedules, and the spatial-temporal relation matrix. We observe similar experimental results on the ridesharing performance in Figure 7, where an embedding dimension of size $n = 64$ yields optimal results in the metrics of order completion rate and average response time.

VI. Conclusion

This paper introduces PreMR, a novel framework that leverages spatial-temporal interval information and self-attention mechanisms to learn intricate matching patterns from existing ridesharing algorithms. PreMR can refine candidate sets for orders, significantly accelerating response time while maintaining high service quality. Extensive experiments on real-world datasets demonstrate the effectiveness of PreMR, particularly in handling large-scale order volumes, showcasing its potential to enhance efficiency and scalability of ridesharing platforms. Future research will focus on incorporating dynamic factors such as traffic conditions and user preferences to further optimize PreMR's performance.

Acknowledgment

We thank anonymous reviewers for their helpful comments. This work was supported in part by National Natural Science Foundations of China under Grants 62172284 and U2001207, and the grant of Guangdong Basic and Applied Basic Research Foundation (No.2022A1515010155). This work was also supported by Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No.2023B1212010007), and the Project of DEGP (No.2023KCXTD042 and No.2021ZDZX1068).

References

[1] Didi chuxing. https://www.didiglobal.com/.
[2] Gaia initiative. https://outreach.didichuxing.com/research/opendata/.
[3] Openstreetmap. http://www.openstreetmap.org/.
[4] Uber. https://www.uber.com/.
[5] G. Brauwers and F. Frasincar. A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3279–3298, 2021.
[6] P. Cheng, H. Xin, and L. Chen. Utility-aware ridesharing on road networks. In *ACM SIGMOD*, pages 1197–1210, 2017.
[7] M. Haliem, G. Mani, V. Aggarwal, and B. Bhargava. A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7931–7942, 2021.
[8] W. He, K. Hwang, and D. Li. Intelligent carpool routing for urban ridesharing by mining GPS trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2286–2296, 2014.
[9] Z. Huang, D. Wang, Y. Yin, and X. Li. A spatiotemporal bidirectional attention-based ride-hailing demand prediction model: a case study in Beijing during COVID-19. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25115–25126, 2021.
[10] J. Jiang, D. Pan, H. Ren, X. Jiang, C. Li, and J. Wang. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In *IEEE ICDE*, pages 843–855, 2023.
[11] J. Li, Y. Wang, and J. McAuley. Time interval aware self-attention for sequential recommendation. In *ACM WSDM*, pages 322–330, 2020.
[12] Y. Li, H. Gu, R. Chen, J. Xu, S. Guo, J. Xue, and M. Xu. Efficient top-k matching for publish/subscribe ride hitching. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3808–3821, 2021.
[13] L. Lin, W. Li, H. Bi, and L. Qin. Vehicle trajectory prediction using LSTMs with spatial–temporal attention mechanisms. *IEEE Intelligent Transportation Systems Magazine*, 14(2):197–208, 2021.
[14] L. Liu, L. Cai, C. Zhang, X. Zhao, J. Gao, W. Wang, Y. Lv, W. Fan, Y. Wang, M. He, et al. Linrec: linear attention mechanism for long-term sequential recommender systems. In *ACM SIGIR*, pages 289–299, 2023.
[15] Q. Liu, S. Wu, and L. Wang. Multi-behavioral sequential prediction with recurrent log-bilinear model. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1254–1267, 2017.
[16] Z. Liu, Z. Gong, J. Li, and K. Wu. Mobility-aware dynamic taxi ridesharing. In *IEEE ICDE*, pages 961–972, 2020.
[17] Z. Liu, Z. Gong, J. Li, and K. Wu. mT-Share: a mobility-aware dynamic taxi ridesharing system. *IEEE Internet of Things Journal*, 9(1):182–198, 2021.
[18] Z. Liu, Z. Li, K. Wu, and M. Li. Urban traffic prediction from mobility data using deep learning. *IEEE Network*, 32(4):40–46, 2018.
[19] Z. Liu, H. Zhang, G. Ouyang, J. Chen, and K. Wu. Data-driven pick-up location recommendation for ride-hailing services. *IEEE Transactions on Mobile Computing*, 23(2):1001–1015, 2024.
[20] Z. Liu, P. Zhou, Z. Li, and M. Li. Think like a graph: real-time traffic estimation at city-scale. *IEEE Transactions on Mobile Computing*, 18(10):2446–2459, 2018.
[21] S. Ma, Y. Zheng, and O. Wolfson. T-share: a large-scale dynamic taxi ridesharing service. In *IEEE ICDE*, pages 410–421, 2013.
[22] S. Ma, Y. Zheng, and O. Wolfson. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1782–1795, 2014.
[23] Z. T. Qin, H. Zhu, and J. Ye. Reinforcement learning for ridesharing: an extended survey. *Transportation Research Part C: Emerging Technologies*, 144:103852, 2022.
[24] H. Ren, S. Ruan, Y. Li, J. Bao, C. Meng, R. Li, and Y. Zheng. Mtrajrec: map-constrained trajectory recovery via seq2seq multi-task learning. In *ACM SIGKDD*, pages 1410–1419, 2021.
[25] J. Shen, N. Tziritas, and G. Theodoropoulos. A baselined gated attention recurrent network for request prediction in ridesharing. *IEEE Access*, 10:86423–86434, 2022.
[26] R. S. Thangaraj, K. Mukherjee, G. Raravi, A. Metrewar, N. Annamaneni, and K. Chattopadhyay. Xhare-a-ride: a search optimized dynamic ride sharing system with approximation guarantee. In *IEEE ICDE*, pages 1117–1128, 2017.
[27] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, and K. Xu. Unified route planning for shared mobility: an insertion-based framework. *ACM Transactions on Database Systems*, 47(1):1–48, 2022.
[28] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment*, 11(11):1633, 2018.
[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.
[30] J. Wang, P. Cheng, L. Zheng, L. Chen, and W. Zhang. Online ridesharing with meeting points. *Proceedings of the VLDB Endowment*, 15(13):3963–3975, 2022.
[31] J. Wang, P. Cheng, L. Zheng, C. Feng, L. Chen, X. Lin, and Z. Wang. Demand-aware route planning for shared mobility services. *Proceedings of the VLDB Endowment*, 13(7):979–991, 2020.
[32] T. Wang, H. Luo, Z. Bao, and L. Duan. Dynamic ridesharing with minimal regret: towards an enhanced engagement among three stakeholders. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3712–3726, 2023.
[33] H. Xu, T. Zou, M. Liu, Y. Qiao, J. Wang, and X. Li. Adaptive spatiotemporal dependence learning for multi-mode transportation demand prediction. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):18632–18642, 2022.
[34] Y. Xu, Y. Tong, Y. Shi, Q. Tao, K. Xu, and W. Li. An efficient insertion operator in dynamic ridesharing services. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3583–3596, 2020.
[35] S. Zhang, Q. Ma, Y. Zhang, K. Liu, T. Zhu, and Y. Liu. Qashare: towards efficient QoS-aware dispatching approach for urban taxi-sharing. In *IEEE SECON*, pages 533–541, 2015.