PREDICTING OFFERS FOR MOBILITY-ON-DEMAND SERVICES: A MACHINE LEARNING-BASED APPROACH

Chenhao Ding[*]
Chair of Traffic Engineering and Control
Technical University of Munich
Archisstraße 21
80333 Munich, Germany
Email: chenhao.ding@tum.de
ORCID: 0009-0004-0423-9614

Florian Dandl
Chair of Traffic Engineering and Control
Technical University of Munich
Archisstraße 21
80333 Munich, Germany
Email: florian.dandl@tum.de
ORCID: 0000-0003-3706-9725

Klaus Bogenberger
Chair of Traffic Engineering and Control
Technical University of Munich
Archisstraße 21
80333 Munich, Germany
Email: klaus.bogenberger@tum.de
ORCID: 0000-0003-3868-9571

[*] Corresponding Author

Word Count: 6997 words + 2 table(s) $\times$ 250 = 7497 words

Submission Date: November 12, 2024

ABSTRACT

The rapid development of mobile internet technology has driven the widespread adoption of Mobility-on-Demand services like ride-hailing and ride-pooling, which are integral to daily travel. Ensuring the quality of these services is vital for Transportation Network Companies such as Uber, Lyft, and Didi, as it directly impacts user experience and revenue. One quality aspect is providing timely and precise predictions of waiting and driving times for users, as this information affects their decision-making. However, creating these predictions is conceptually and computationally challenging in high-demand scenarios due to the dynamic nature of the services and exhaustive vehicle information processing. This paper introduces a machine learning-based model incorporating a three-dimensional encoding method based on pick-up likelihood, vehicle capacity, and detour degree to efficiently encode user requests and fleet status. A selection mechanism filters encoded information in temporal and spatial domains, reducing computational burden. The model employs an encoder-decoder architecture, with a Convolutional Neural Network encoding fleet status as an image and Fully Connected Networks decoding it for various predictions. A case study using the Manhattan taxi dataset demonstrates the model's effectiveness, showing that spatial domain information significantly impacts predictive performance. The model achieves a True Negative Rate of approximately 90% in identifying unserviceable requests and outperforms heuristic algorithms in travel time prediction accuracy and efficiency, though it lags in waiting time prediction. Future research should explore nonlinear encoding for vehicle pick-up likelihood and test various neural network configurations to improve prediction performance and transferability.

Keywords: Mobility-on-Demand, Machine learning, Offer prediction

INTRODUCTION

With the rapid development of mobile internet technology, Mobility-on-Demand (MoD) services, such as ride-hailing (RH) and ride-pooling (RP), have gained widespread attention and become integral to people's daily travel routines. According to Uber Technologies, Inc. (*1*), approximately 9.4 billion trips were completed globally in 2023. In comparison, DiDi Global Inc. (*2*) reported serving 10.8 billion trips in China alone. RH offers users on-demand door-to-door transportation services, significantly enhancing travel flexibility and alleviating parking pressure from users in urban areas. Building on this, RP allows multiple passengers with identical or similar routes to share a vehicle, partially combining their journeys. This method increases vehicle occupancy, thereby improving transportation efficiency, reducing the number of vehicles on the road, alleviating traffic congestion, and cutting carbon emissions.

For Transportation Network Companies (TNCs) such as Uber, Lyft, and Didi, providing users with offers that are produced quickly and contain accurate forecasts for trip planning is crucial for ensuring a positive user experience and increasing revenue. When users access the MoD service app to input trip details such as origin, destination, and expected departure time, they expect the system to promptly provide essential information about available services, enabling effective travel planning. Based on real-time fleet status, such as the location and onboard passenger count of each vehicle, most TNCs provide users with estimated service details, including vehicle availability, estimated pick-up time, estimated travel time, and estimated cost, before users commit to the service. This information has to be provided quickly, as lengthy computation times result in slow app response times and can frustrate users, thereby decreasing their willingness to use the services. Additionally, the predictions must be as accurate as possible to reflect the actual services provided later. Inaccurate predictions can lead to users making poor decisions, which, over time, can erode their trust in the MoD services, reducing overall acceptance and loyalty (*3*).

This study contributes to the research on MoD offer prediction by:

1. Describing the MoD offer creation problem.
2. Proposing a machine learning (ML)-based offer prediction approach that employs an encoder-decoder architecture to enhance flexibility for different prediction tasks.
3. Developing a three-dimensional encoding method based on pick-up likelihood, vehicle capacity, and detour degree to achieve efficient encoding of vehicle status.
4. Introducing a selection mechanism that improves prediction efficiency by reducing the amount of information processed in both temporal and spatial domains.
5. Conducting a case study using the publicly available New York City taxi dataset (*4*) to validate the performance of the proposed approach in terms of prediction accuracy and efficiency.

The remainder of this paper is organized as follows. The **LITERATURE REVIEW** section summarizes and discusses the research progress of existing MoD simulation frameworks and the state-of-the-art ML applications in the MoD field. The **METHODOLOGY** section discusses the MoD offer creation problem and introduces the proposed approach, providing a detailed explanation of the design concepts and functionalities of the individual modules. The **CASE STUDY** section describes the experiments based on the Manhattan taxi dataset, presents the results, evaluates the prediction accuracy and efficiency of the proposed approach, and discusses the reasons behind the model's performance. Finally, the **CONCLUSION AND FUTURE WORK** section summarizes the research and outlines potential directions for future work.

LITERATURE REVIEW

Modeling of MoD services

Research on Mobility-on-Demand services has become a key focus in transportation studies. A scientometric analysis by Guo and Li (*5*) reviews the development and research directions of MoD systems. Similarly, Shaheen and Cohen (*6*) examine methodologies for studying MoD, evaluating the impacts of shared modes and autonomous MoD (AMoD) on public transportation. For instance, Paparella et al. (*7*) propose a time-invariant network flow model for two-person ride-sharing MoD systems, utilizing a Poisson process to describe trip requests and analyze ride-sharing in the spatiotemporal dimension. Pavone et al. (*8*) employ a fluid model and linear programming to devise an optimal real-time rebalancing policy, validated through simulations and hardware experiments.

In addition to mathematical models, comprehensive simulation frameworks have been developed to simulate MoD fleet operations. Commercial solutions such as Aimsun (*9*) and PTV (*10*) have been launched, while researchers have increasingly focused on open-source frameworks. Ruch et al. (*11*) introduce AMoDeus, a software package within MATSim (*12*) for the quantitative analysis of AMoD systems. Auld et al. (*13*) develop POLARIS, integrating dynamic simulation of travel demand, network supply, and network operations. Kucharski and Cats (*14*) introduce MaaSSim, an agent-based simulator for two-sided mobility platforms, modeling urban mobility dynamics with agents such as travelers, drivers, and an intermediary platform matching demand with supply. FleetPy, developed by Engelhardt et al. (*15*), focuses on user-operator interactions and supports multiple operators within the transportation system, offering a modular structure for customization and transferability.

Vehicle assignment is a core challenge for MoD services and is closely tied to offer creation. This problem can be addressed using rules, heuristics, or optimization algorithms. Fagnant and Kockelman (*16*) tackle this problem by iteratively matching travelers with the nearest vehicles within a 5-minute travel time, expanding the search zone if necessary. Zhang and Guhathakurta (*17*) assign non-busy vehicles to travelers by selecting the vehicle that offers the lowest combined cost of time and fare. Alonso-Mora et al. (*18*) present a scalable mathematical model that dynamically optimizes routes, starting with a greedy assignment and improving through constrained optimization for quick, high-quality solutions. Kucharski and Cats (*19*) use a demand-driven algorithm that matches trips into shared rides via a directed shareability multi-graph, narrowing the search space for efficient graph searches. Engelhardt et al. (*20*) introduce a speed-up algorithm to find the system's optimal assignments at each decision time step. Hyland and Mahmassani (*21*) define and model assignment strategies for RH, finding that optimization-based strategies and those incorporating en-route pickups and drop-offs reduce fleet miles and wait times, particularly during peak periods.

Another crucial aspect of offer creation is the interaction between users and operators (*22*). To simplify this process, some models, like POLARIS (*13*), treat user on-demand requests as service bookings, assuming that users will invariably accept the operator's offer. This approach reduces the complexity of the offer creation problem, transforming it into a straightforward vehicle routing issue and allowing researchers to focus more specifically on vehicle assignment. Conversely, models like SimMobility (*23*) also simulate the operator's ability to reject requests. In these models, operators decide whether to fulfill a user's request based on the current fleet status. However, in reality, the decision-making process between users and operators is bidirectional. Operators assess the feasibility of fulfilling user requests in real-time while users evaluate their options among various travel modes (*24*). The offers provided by operators play a crucial role in users' decisions to use MoD services. Models such as MaaSSim (*14*) and FleetPy (*15*) effectively simulate this bidirectional decision-making process.

Nonetheless, most MoD simulation frameworks that offer mode choice functionality for users rely on comprehensive and precise simulation results as the basis for decision-making rather than quickly predicted offers. To generate these results, the model must evaluate all available vehicle data and determine the optimal vehicle assignments. The speed of this process is influenced by both fleet size and demand level, making rapid processing challenging in scenarios involving large fleets and high demand. Although Engelhardt et al. (*15*) employed heuristic algorithms to predict offers by integrating each passenger's pick-up and drop-off locations into fleet routing plans and optimizing them based on the operator's objectives, this approach still does not entirely mitigate the impact of fleet size and demand level. For such highly dynamic, data-intensive problems, ML offers a potentially superior solution (*25*).

ML Approaches for MoD services

ML offers innovative solutions for MoD services from a big data perspective. Chen et al. (*26*) propose UberNet, a deep learning convolutional neural network designed for short-term demand prediction for RH. Chu et al. (*27*) introduce Multi-Scale Convolutional Long Short-Term Memory (MultiConvLSTM), a deep learning model inspired by image and video processing techniques that treats travel demand as image pixel values to capture both temporal and spatial correlations. Ke et al. (*28*) propose the Fusion Convolutional Long Short-Term Memory Network (FCL-Net), integrating convolutional LSTM layers, standard LSTM layers, and convolutional layers to capture spatiotemporal correlations while ranking variable importance using spatially aggregated random forest. Wang et al. (*29*) propose a Deep Spatio-Temporal ConvLSTM framework for travel demand forecasting, capturing both temporal and spatial dependencies with separate branches for closeness, period, and trend components, and employing a linear fusion method for final predictions. On the operational side, Lei et al. (*30*) combine optimization and ML to proactively relocate vehicles and ensure fair income distribution. Guo and Xu (*31*) propose a ride-sharing vehicle dispatching and routing method, which includes vehicle routing decision-making through a Markov decision process with deep reinforcement learning, and request-vehicle assignment based on routing learning values. However, offer prediction using ML remains an underexplored area.

METHODOLOGY

This paper uses simulation to analyze the performance of two different methods to address the offer creation problem. This section formalizes the MoD offer creation problem, presents the simulation framework, briefly summarizes the insertion heuristic approach, and discusses in detail the design concepts and functionalities of the ML-based solution approach.

Formulation of the MoD Offer Creation Problem

The offer creation problem describes the task of an MoD operator to generate information about a possible trip with the service after a user sends a request with their app. In this task, the MoD operator controls a set of vehicles $\mathbf{V}$ in a street network $\mathbf{G} = (\mathbf{N}, \mathbf{E})$, where $\mathbf{N}$ are the nodes and $\mathbf{E}$ are the edges containing travel distance and travel time costs. The operator has access to the current fleet state $\mathbf{F_S}$, which includes the position and onboard passengers of each vehicle, along with their assigned routing plans and any unprocessed requests. An app request $R \subset \mathbf{R}$ includes information about earliest departure times t_{ed}, origin $p_o \in \mathbf{N}$, destination $p_d \in \mathbf{N}$, and the number of passengers n_{pax}, i.e., $R = (t_{ed}, p_o, p_d, n_{pax})$. When a new app request R is received, the MoD Offer Creation Problem must be solved, requiring the operator to generate an offer O_p containing information on the expected pick-up time t_{epu}, expected drop-off time t_{edo}, and expected cost c_e, i.e., $O_p = (t_{epu}, t_{edo}, c_e)$. This problem can be very challenging.

- For large-scale services, finding a reasonable solution for a new request based on the current fleet state can be computationally expensive.
- MoD services receive new app requests dynamically, and future vehicle-user assignments can affect the service levels of previously assigned requests. Figure 1 illustrates how the dynamic mechanism works. At the simulation time step t_0, the MoD operator receives a request R_0 and decides to assign it to the vehicle v_0, generating a routing plan $V^0_{rp\_0}$ and a predicted offer $O^0_p = (t^0_{epu\_0}, t^0_{edo\_0})$ [1]. The user accepts O^0_p and decides to use the MoD service. At the simulation time step t_1, the MoD operator receives a new request R_1 and decides to combine R_1 and R_0 to form an RP. However, as R_1 is added to $V^0_{rp\_0}$, $t^0_{epu\_0}$ and $t^0_{edo\_0}$ of O^0_p need to be updated as well, resulting in $t^0_{epu\_1}$ and $t^0_{edo\_1}$. As simulation time advances, the user of R_0 is picked up at the moment t^0_{pu}. It is at this precise moment that the realized pick-up time of this user can be truly determined. Similarly, when the user is dropped off at t^0_{do}, the realized drop-off time becomes clear. Finally, it can be observed that the pick-up and drop-off time in O^0_p and the realized service $RS_0 = (t^0_{pu}, t^0_{do})$ have unavoidable deviations due to the addition of other requests.

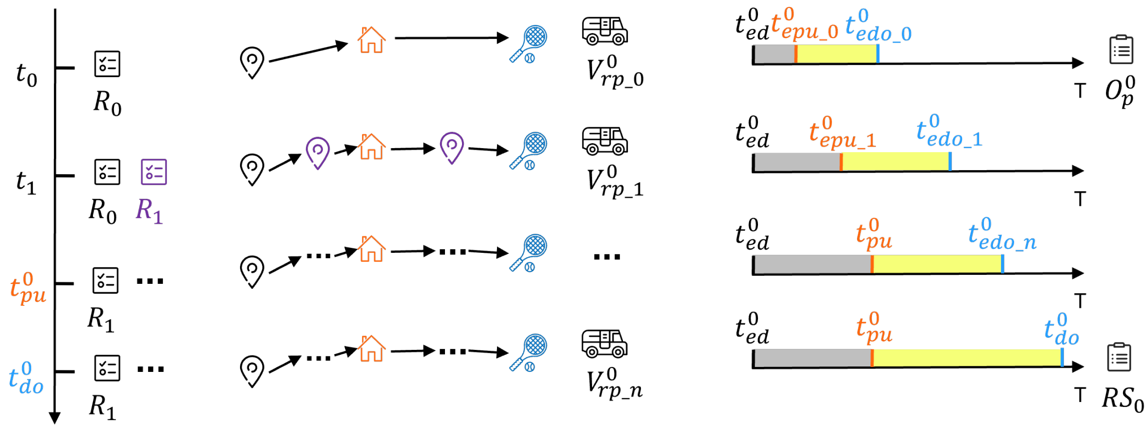

t_0
R_0
$V^0_{rp\_0}$
t^0_{ed} $t^0_{epu\_0}$ $t^0_{edo\_0}$
T
O^0_p
t_1
R_0 R_1
$V^0_{rp\_1}$
t^0_{ed} $t^0_{epu\_1}$ $t^0_{edo\_1}$
T
t^0_{pu}
R_1
t^0_{ed} t^0_{pu} $t^0_{edo\_n}$
T
t^0_{do}
R_1
$V^0_{rp\_n}$
t^0_{ed} t^0_{pu} t^0_{do}
T
RS_0

FIGURE 1: The dynamic mechanism of offer creation

This study investigates a constrained version of the general problem with the following attributes:

- Users must be picked up within the maximum waiting time t_{mw}, i.e., $(t_{epu} - t_{ed}) \leq t_{mw}$.
- The detour time factor generated by the MoD service, $f_{dt} = (t_{edo} - t_{epu} - t_{od})/t_{od}$, where t_{od} denotes the direct travel time from origin to destination, cannot exceed the maximum detour time factor f_{mdt}.
- The number of passengers n_{pax} cannot exceed the number of vehicle's available seats.

If any of these conditions are not met by a vehicle, it will be marked as unable to provide service. If no vehicles in the fleet can provide service, the request will be rejected.

[1]Pricing is not in the focus of this study and therefore omitted from now on.

Simulation Framework

In this study, FleetPy (*15*) is chosen to model the dynamics of realistic MoD services — including the interaction of operators with users. During the simulation, offer creation problems appear and have to be solved. FleetPy is an agent-based framework for MoD fleet simulation, as illustrated in Figure 2. Users send travel requests $\mathbf{R}$ to an MoD operator, who then provides offer predictions $\mathbf{O_P^{IR}}$ based on the current fleet state $\mathbf{F_S}$. By default, FleetPy employs an insertion heuristic in the Immediate Response (IR) module, which integrates users' pick-up and drop-off locations into each vehicle's routing plan. Offer attributes are derived from the optimal plan according to the operator's objectives. At predetermined intervals, all accepted offers are batch-optimized and reassigned to vehicles to ensure a better system-wide solution.

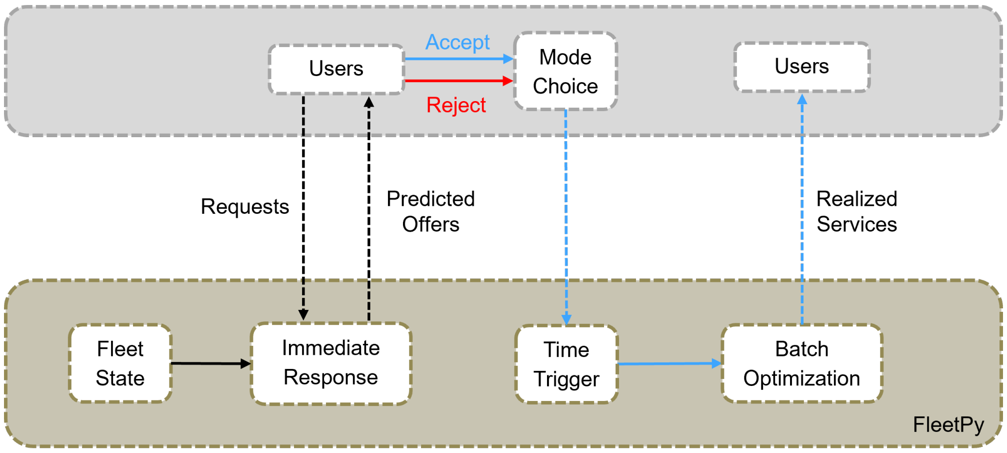


FIGURE 2: The general workflow of FleetPy

The fleet state $\mathbf{F_S}$ is described by routing plans V_{rp} for each vehicle. Besides the current position and onboard requests, V_{rp} consists of planned stops, denoted as $s_v^i = (p_{ps}^i, t_a^i, t_d^i, R_o^i, R_a^i)$. Here, p_{ps}^i denotes the position of planned stop i, t_a^i denotes the arrival time at planned stop i, t_d^i is the departure time at planned stop i, $R_o^i \subset \mathbf{R}$ is the users on board the vehicle at the planned stop i, and $R_a^i \subset \mathbf{R}$ is the users alighting from the vehicle at the planned stop i. Collectively, these individual routing plans form the overall fleet state, i.e., $V_{rp} \subset \mathbf{F_S}$.

Due to the dynamic nature of the underlying vehicle routing problem, there often is a deviation between FleetPy's $\mathbf{O_p^{IR}}$ and the realized service (**RS**) received by the users.

ML-Model Architecture

This study proposes an ML-based model to predict offers for MoD services. During the simulation, user requests $\mathbf{R}$ and the corresponding fleet status $\mathbf{F_S}$ are recorded as input data for the model. The realized services **RS** provided for each request by fleetpy are documented as target values for the ML module. The model then predicts whether a service can be provided. If it can, the model outputs specific offer attribute values, $\mathbf{O_P^{ML}}$, where $O_p^{ml} = (t_{ew}^{ml}, t_{et}^{ml})$. By directly learning from **RS**, the model can mitigate the effects of dynamic mechanisms. Figure 3 illustrates the overall workflow of the proposed model.

The idea of this study's approach is to guide and construct the input features representing vehicle availability for a user request along three dimensions: pick-up likelihood, vehicle capacity, and detour degree. In the data preprocessing phase, the model first encodes the input $\mathbf{R}$ and $\mathbf{F_S}$ in these three dimensions. Before being fed into the ML module, a selection mechanism is used to streamline the encoded data, thereby improving the processing speed and prediction accuracy.

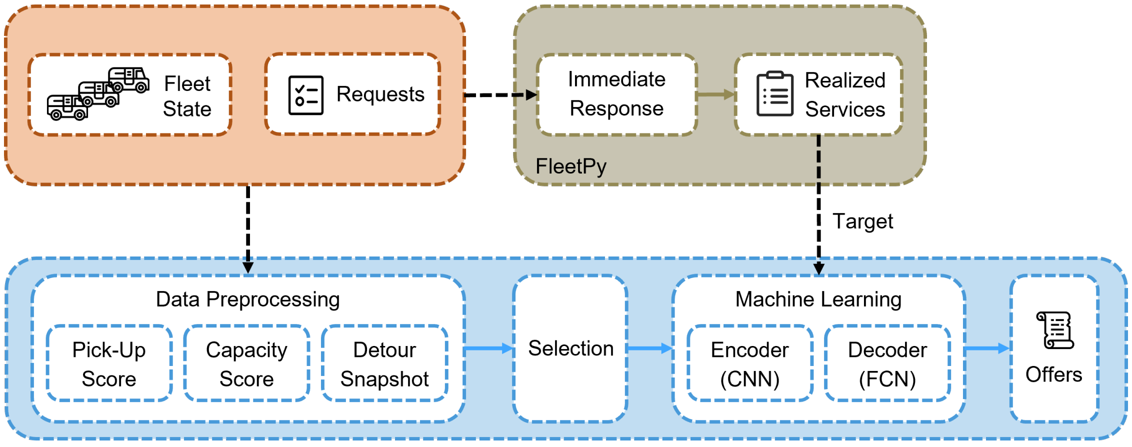


FIGURE 3: The overall workflow of the proposed model

Offer prediction involves two aspects: recognizing requests that cannot be served and predicting the offer attribute values for the requests that can be served. This approach employs an encoder-decoder architecture to accommodate the specificity of the prediction task. The encoder module comprehends and refines the encoded information to generate request embeddings (RE). For different prediction needs, different decoder modules are used to decode the RE and output the corresponding prediction results. In the following, each module is discussed in detail.

Pick-Up Score

An effective offer needs to ensure that a vehicle can reach the user's departure location within t_{mw}. The pick-up score s_{pu} is proposed in Eq. (1) to assess a vehicle's potential to fulfill this requirement. For each planned stop of the vehicle in the assigned routing plan, the vehicle's pick-up ability is scored. The time for a vehicle to travel from a planned stop to pick up a user, t_{vpu}, can be computed according to Eq. (2); it consists of two parts: t_{vo}, the travel time from the stop to the user's origin, and an additional component representing the time required for the vehicle to depart from the stop. The longer it takes a vehicle to pick up a user, the lower the s_{pu}. If the travel time exceeds t_{mw}, the vehicle's s_{pu} at that planned stop becomes 0.

Calculating the s_{pu} for each of the m planned stops across all n vehicles in the fleet yields a pick-up score table $\mathbf{S_{pu}}$ (3). This table reflects the fleet's overall potential to reach passengers on time.

$$s_{pu} = max(0, \frac{t_{mw} - t_{vpu}}{t_{mw}}) \tag{1}$$

$$t_{vpu} = t_{vo} + (t_d - t_{ed}) \tag{2}$$

$$\mathbf{S_{pu}} = \begin{bmatrix} s^i_{pu_1}, & \dots, & s^i_{pu_m}, \\ \vdots & & \vdots \\ s^n_{pu_1}, & \dots, & s^n_{pu_m} \end{bmatrix} \tag{3}$$

Capacity Score
To ensure that vehicles meet users' travel needs, they must provide sufficient seating capacity. The status information of a vehicle at each planned stop includes details about passengers boarding and alighting. Changes in the number of passengers on board directly impact the vehicle's ability to accommodate new riders. This method simplifies the impact of these changes by focusing on the number of available seats n_{fs} as the vehicle departs each stop. The seating capacity is then scored using Eq. (4), where n_{mfs} represents the vehicle's maximum seating capacity. The more empty seats a vehicle has, the higher its score. Similar to the calculation of $\mathbf{S_{pu}}$, the capacity scores for all n vehicles across all m planned stops are computed, forming a capacity score table $\mathbf{S_{cap}}$ (5), which reflects the overall seating capacity of the fleet in response to the request.

$$s_{cap} = \frac{n_{fs}}{n_{mfs} - n_{pax}} \tag{4}$$

$$\mathbf{S_{cap}} = \begin{bmatrix} s^{i}_{cap_1}, & \dots, & s^{i}_{cap_m}, \\ \vdots & & \vdots \\ s^{n}_{cap_1}, & \dots, & s^{n}_{cap_m} \end{bmatrix} \tag{5}$$

Detour Snapshot
Another crucial factor in evaluating a vehicle's service capability is detour time. This issue does not need consideration for RH, where vehicles and users are uniquely paired. However, in RP, where users share the vehicle with others, direct transportation from origin to destination is often not feasible. Users may need to stop at other passengers' locations along the route, resulting in detours. Although a certain amount of detour is acceptable, as the reduced trip cost from sharing compensates for the increased travel time, excessively long detours can negatively impact the user experience. Therefore, it's essential to factor in detour time for each vehicle and ensure it remains within a reasonable limit to maintain service quality.

When constructing a new routing plan for each vehicle, the new planned stops N_p for the vehicle include two types of key points: the planned stops in the vehicle's current routing plan and the origin and destination points of the new user. By connecting these points and using the travel time between them as the edge lengths, we create a network that reflects the connectivity relationships along with travel times. This network represents all potential routes for the vehicle after incorporating the new user, forming what is known as a detour snapshot S_{dt}:

$$S_{dt} = \{t^{p_i}_{p_j} | p_i, p_j \in N_p, p_i \neq p_j\} \tag{6}$$

Here, $t^{p_i}_{p_j}$ denotes the travel time between nodes p_i and p_j. By calculating detour snapshots for all vehicles, we compile a detour snapshot table $\mathbf{S_{dt}}$, which reflects the fleet's overall detour degree.

Selection Mechanism
Offer prediction needs to be accomplished in a very short time, so it is necessary to improve the computational efficiency of the model. This approach reduces the computational burden of the model by designing a selection mechanism that streamlines information in both temporal and spatial domains. In the temporal domain, the model reduces the feature dimensions of each vehicle, while in the spatial domain, it filters the number of vehicles considered for offer prediction.

When assigning a vehicle to a user request, priority is given to nearby vehicles with sufficient capacity. By combining the $\mathbf{S_{pu}}$ and $\mathbf{S_{cap}}$, we infer that a suitable vehicle has a non-zero

s_{pu} and s_{cap} at least at one of its planned stops. Therefore, vehicles with a s_{pu} or s_{cap} of 0 at all planned stops are filtered out from subsequent calculations. Additionally, each vehicle's maximum s_{pu} is sorted, and only the top n_{mv} vehicles are selected as candidates, retaining those closest to the user's departure point.

Each vehicle may have multiple planned stops in its routing plan. Since these stops are visited sequentially, the likelihood of a vehicle being able to arrive at these stops and then reach the user's departure point within t_{mw} diminishes for later stops. Therefore, this approach only considers the first n_{mps} stops of each vehicle, ignoring the subsequent stops to reduce the feature dimensions of each vehicle.

The Encoder-Decoder Architecture

The encoder-decoder architecture is considered one of the most rational neural network designs and is widely used in deep learning, particularly in models like the Transformer (*32*). The encoder is responsible for understanding and processing the input information to create information embeddings. The decoder, on the other hand, provides the ML model with the flexibility to handle various tasks. Depending on the task requirements, different decoder networks, tailored to the specific task scenario, can be employed and combined with the encoder to form a complete neural network prediction module.

The encoded information of the user's request and the corresponding fleet state is stored in a matrix, where each row represents a vehicle, and the columns contain encoded information in three dimensions: pick-up likelihood, vehicle capacity, and detour degree. This matrix serves as a snapshot of the current fleet state, akin to an image that can be processed using machine vision techniques. In this study, a Convolutional Neural Network (CNN) (*33*) is selected for extracting and embedding the encoded information due to its effectiveness in handling such image-like data.

For each request, the offer prediction needs to be performed at most twice. First, it must be determined whether the request can be adequately served with the current fleet state. If it can, the specific offer attribute values need to be predicted. To meet these two-aspect prediction requirements, two different Fully Connected Networks (FCN) (*34*) are employed: one for identifying rejected requests and the other for predicting the values of specific offer attributes.

CASE STUDY

This section outlines the experimental design of the study, presents the results, and concludes with a brief discussion of the findings.

Experimental Design

FleetPy

To train the ML model, we configure FleetPy's simulation environment and construct the dataset based on the results of FleetPy's runs. Table 1 shows all the key configurations of FleetPy. It is important to note that t_{mw} is set to 600 seconds, meaning that all users must be picked up within 10 minutes after their t_{ed}. Additionally, f_{mdt} is set to 40%, indicating that the detour time incurred by the RP services must be at most 40% of the t_{od}; otherwise, the user's request will be rejected. All vehicles in the fleet have a maximum capacity n_{mpax} of 4 passengers.

The operating area for the MOD service is confined to Manhattan, NYC. The network is extracted from OpenStreetMap. To reflect real road traffic conditions, all link travel times are dynamically scaled based on the simulation time. The shortest paths between nodes are precomputed, and a lookup table is created to represent the travel times between all origin and destination pairs.

The MoD demand in this study is derived from the New York City taxi dataset (*4*), a widely used open-source resource. We utilize taxi trip data from November 11 to November 18, 2018, selecting trips that both start and end in Manhattan. To reduce computational complexity, 10% of these trips are randomly sampled and treated as RP demand, resulting in a total of 320,434 trip requests. To ensure robustness, we generate three sets of trip data using three different random seeds. All results presented in the following sections are the averaged outcomes from these three datasets.

Additional details on the processing of network and trip data can be found in the work of Engelhardt et al. (*35*).

TABLE 1: Configurations of FleetPy

Parameter	Symbol	Value
Simulation duration	t_{sim}	24 h
Simulation step	t_{step}	1 s
Fleet size	f_s	200
Maximum waiting time	t_{mw}	600 s
Maximum detour time factor	f_{dt}	40%
Demand	D	Manhattan taxi data, from Nov. 11 to Nov. 18 2018
Network	G	Manhattan network
Vehicle capacity	n_{mpax}	4

Dataset

During the FleetPy simulation, the fleet state at t_{ed} of each request is recorded and combined with the request information to form the feature dataset. At the end of the simulation, acceptance or rejection of requests, realized waiting times, and realized travel times are used to construct the corresponding target dataset. Data from the entire week of November 12 to November 18 is utilized to build the training and testing sets, with a split ratio of 70% for training and 30% for testing. Additionally, data from the entire day of November 11 is used to construct the validation set. Figure 4 shows the distribution of features and targets for each dataset.

It can be observed that the rejection rate of user requests in all the datasets is approximately 14%, indicating that the FleetPy configurations are sufficient to meet the travel demand used in this study. The distributions of travel time and waiting time are consistent across all datasets, with the mean travel time being around 644 seconds and the mean waiting time approximately 371 seconds.

Scenario Specification

The role of the selection mechanism is primarily defined by the parameters n_{mv} and n_{mps}. In the spatial domain, n_{mv} controls the maximum number of vehicles the model considers. In the temporal domain, n_{mps} limits the scope of the model's perspective on the vehicle routing plan.

By analyzing the FleetPy simulation results, Figure 5 illustrates the distribution of the number of available vehicles corresponding to requests and the order of the planned stop at which newly incoming requests are inserted into existing vehicle routes. It can be observed that while the fleet

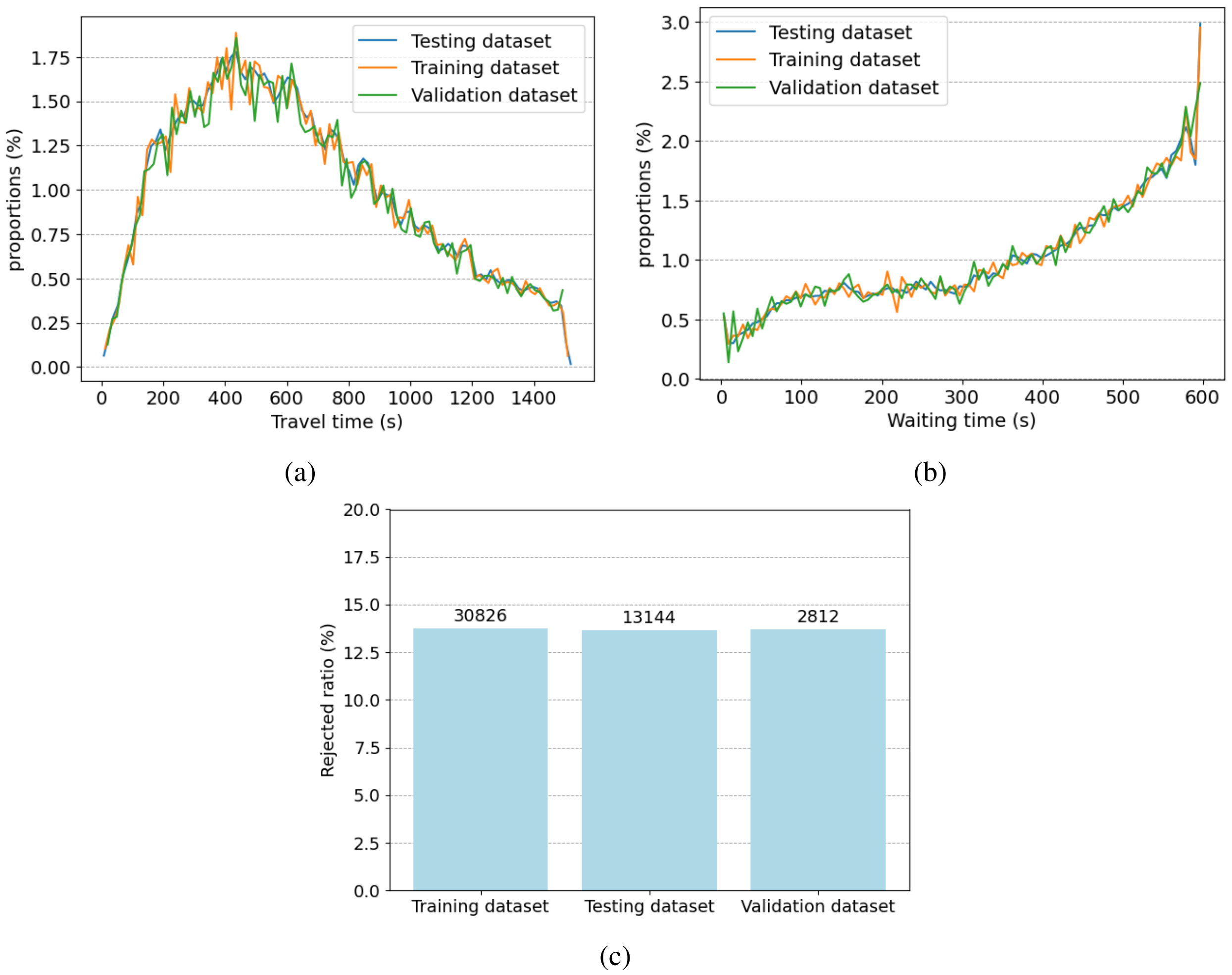


FIGURE 4: The distribution of features and targets for each dataset. (a) shows the distribution of travel time, (b) shows the distribution of waiting time, and (c) shows the ratio of requests rejected by FleetPy in each dataset, with the numbers above indicating the quantities.

size is 200 vehicles, over 95% of requests only have at most 60 vehicles potentially offering service. Moreover, most requests are picked up before the vehicle's fifth planned stop.

This study designs various scenarios from these two aspects to explore the performance of the proposed model and selection mechanism, respectively. Table 2 summarizes the model configurations for these scenarios. In the base scenario (BS), n_{mv} is set to 60 vehicles, and n_{mps} is set to 5 planned stops. NV-20, NV-40, NV-100, and NV-200 represent different scales of n_{mv}, ranging from considering only the 20 vehicles in the user's neighborhood to considering the entire fleet to examine the role of the selection mechanism in the spatial domain. NPS-2, NPS-3, NPS-4, NPS-6, and NPS-7 explore the role of selection mechanisms in the temporal domain, ranging from considering only the 2 planned stops of each vehicle to considering 5 planned stops.

Each of the aforementioned scenarios includes two ML modules: one decoder is designed as an FCN dealing with the classification problem (i.e. user acceptance or rejection), and the other is an FCN capable of outputting two predicted offer attribute values (i.e. waiting and travel time). The encoder of both modules is a CNN. For the classification problem, the ML module employs

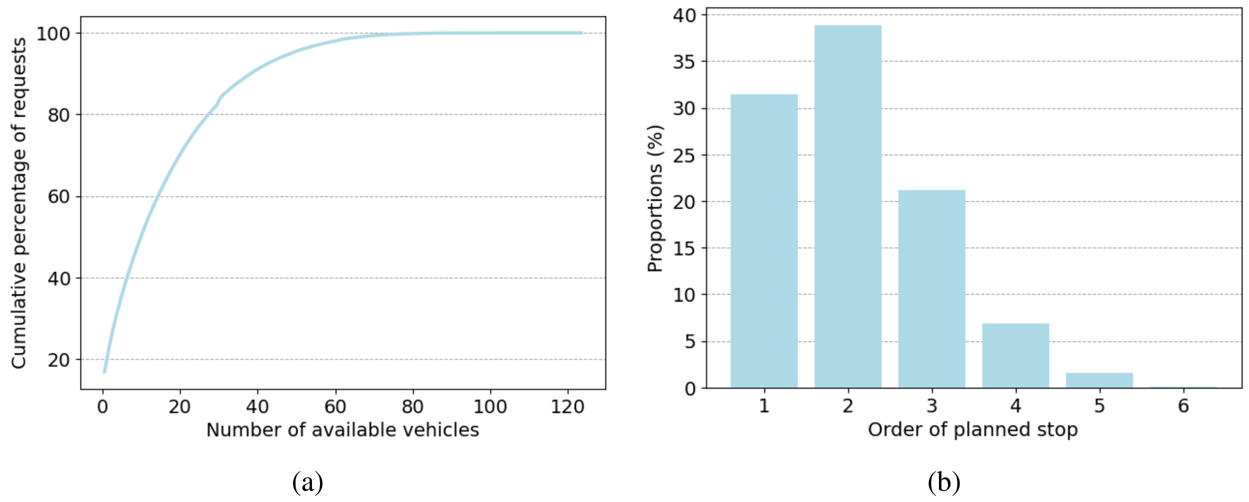


FIGURE 5: The distribution of available vehicle numbers and valid planned stop order. (a) shows the number of available vehicles for each request, and (b) illustrates the order of the planned stop at which the newly incoming request is served.

TABLE 2: Scenario Configurations

Scenario name	n_{mv}	n_{mps}
Base scenario (BS)	60	5
NV-200	200	5
NV-100	100	5
NV-40	40	5
NV-20	20	5
NPS-7	60	7
NPS-6	60	6
NPS-4	60	4
NPS-3	60	3
NPS-2	60	2

Binary Cross Entropy (BCE) as the loss function, while for the offer attribute values prediction, it uses Mean Squared Error (MSE). The model is trained using Adaptive Moment Estimation (Adam) (*36*) as the optimizer, with a learning rate of 0.001.

Key Performance Indicators

To compare the model's prediction performance across scenarios, Key Performance Indicators (KPIs) need to be defined. The results are compared with both $\mathbf{O_P^{IR}}$ and **RS** to evaluate the model's performance. When predicting request acceptance, it is crucial to avoid misclassifying unfeasible requests as feasible, as this significantly impacts users' ride experiences. In the dataset, served requests are labeled as 1 (positive), while rejected requests are labeled as 0 (negative). Therefore, the model should prioritize accurately identifying rejected requests by increasing true negative (TN) counts and decreasing false positive (FP) counts. Additionally, true positive (TP) and false negative (FN) requests should also be considered to enhance the overall benefit for TNCs.

The True Negative Rate (TNR) (7) assesses the model's ability to recognize rejected requests. Precision (8) evaluates the model's prediction accuracy for served requests, while Recall (9) indicates the model's tendency to underreport. However, since approximately 86% of requests are served, relying solely on a single metric does not fully capture the model's performance on positive cases. Therefore, the F1-Score (10), which combines Precision and Recall, is more suitable for assessing the model's overall performance.

For predicting the attribute values of offers, two key features are considered: waiting time and travel time. The Mean Absolute Error (MAE) of these two attributes is used to evaluate the model's overall prediction performance. Additionally, heat maps of the joint distribution of absolute errors for these two attributes are used to visualize the model's prediction accuracy at the individual offer level.

Beyond accuracy, the efficiency of the prediction process is also crucial. The offer prediction time of the proposed model, t_{op}^{ml}, is the combined duration of data preprocessing, data selection, and ML processing. This approach provides a realistic and comprehensive view of the model's efficiency, making the evaluation practical and applicable to real-world applications. Based on the processing time of the IR module, t_{op}^{ir}, the Efficiency Improvement Rate (EIR) (11) is used as a KPI to evaluate the improvement in predictive efficiency of the proposed model.

$$\text{TNR} = \frac{TN}{TN+FP} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{8}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{9}$$

$$\text{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{10}$$

$$\text{EIR} = \frac{t_{op}^{ir} - t_{op}^{ml}}{t_{op}^{ir}} \times 100 \tag{11}$$

Results

This section discusses the model's results across various scenarios, focusing on three key aspects of offer prediction. The first part analyzes the model's ability to accurately recognize rejected requests. The second part evaluates the model's accuracy in predicting offer attribute values. The third part examines the model's operational efficiency.

Recognition

According to the FleetPy configuration used in this study, requests are determined to be serviceable only by the IR module. Therefore, this study assumes the recognition results of the IR module as ground truth and trains and validates the model based on these results. Figure 6 shows the recognition performance of the model in all scenarios.

In BS, the model achieves a TNR of approximately 90%, indicating its effectiveness in

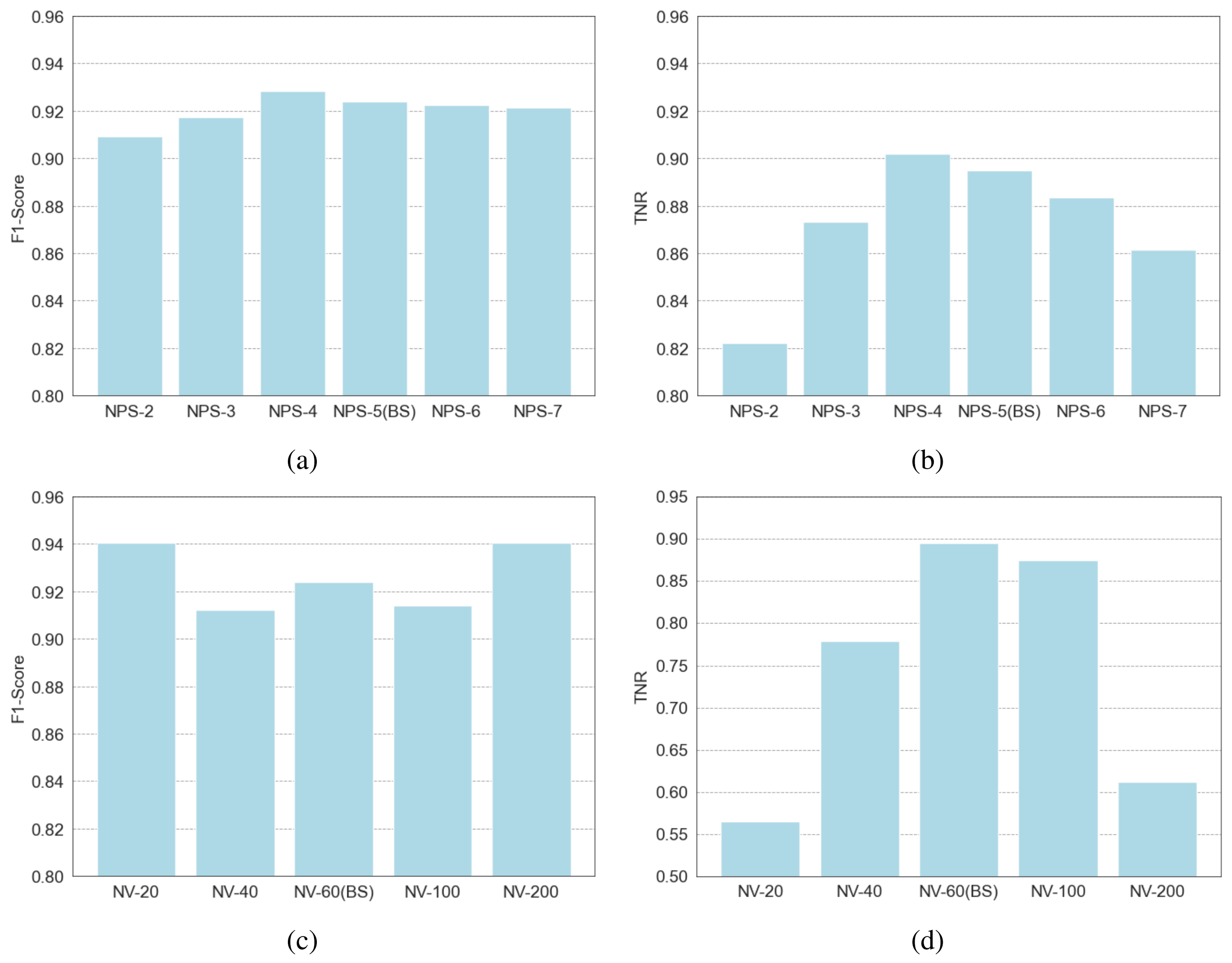


FIGURE 6: F1-Score and TNR for all scenarios. (a) and (b) illustrate the model's recognition performance for different n_{mps}, and (c) and (d) show the effect of n_{mv} on the model's performance.

recognizing the majority of requests that need to be rejected, despite the rejection rate being only around 14% in the dataset. NPS-4 slightly outperforms BS in temporal domain selection. As shown in Figure 5, only about 2% of the requests are served at the fifth planned stop of the vehicle. This suggests that ignoring this stop allows the model to focus on the first four stops, which contain more relevant information. As the number of planned stops considered by the selection mechanism decreases, the model's recognition rate for rejected requests declines. This reduction is attributed to the loss of information in the shortened field of view, leaving the ML model with insufficient features to make accurate judgments. Specifically, nearly 22% of the requests are served at the vehicle's third planned stop, and ignoring this stop's feature information in NPS-2 results in a TNR of about 82%. In scenarios where the selection mechanism considers additional planned stops, such as NPS-6 and NPS-7, the model's TNR decreases. This decline occurs because all requests in the dataset are serviced by or before the vehicle's fifth planned stop, making information about subsequent stops a distraction that interferes with the model's judgment.

The model exhibits similar performance trends in selecting information in the spatial domain. Overly reducing or increasing the number of considered vehicles diminishes the model's

TNR. Notably, the model's performance exhibits different sensitivities to information selection across the spatial and temporal domains, with spatial information selection having a greater impact. When the amount of temporal information is scaled down to 70%, as in NPS-2, the model still achieves a TNR higher than 82%. Conversely, when spatial information is scaled down to 70%, as in NV-20, the model's accuracy in correctly identifying rejected requests drops to approximately 57%, which is only slightly better than random guessing.

This pattern is also observed in the model's ability to determine which requests can be served. Compared to BS, NPS-4, which considers one less planned stop, achieves a higher F1-Score of over 0.92. As n_{mps} increases or decreases, the model's F1-Score decreases. The selection mechanism in the spatial domain causes more significant changes in the F1-Score. Particularly in NV-20 and NV-200, although the F1-Score exceeds 0.94 in both scenarios, higher than in all other scenarios, this occurs because the model either receives too little valid information or too much irrelevant information, impairing its ability to make accurate judgments. Consequently, the model tends to classify more requests as serviceable, which is a more prevalent category in the dataset.

Offer Attributes

In FleetPy, $\mathbf{O_P^{IR}}$ is the prediction of **RS**. Therefore, in this study, **RS** is considered the ground truth on which the model's training and validation are based. By comparing the mean error of the model's predicted values with **RS**, we can evaluate the model's predictive performance. Additionally, by comparing the predicted values of the model with $\mathbf{O_P^{IR}}$, we can determine how optimized the proposed model is compared to the IR module. Figure 7 illustrates these two comparisons of the model for all scenarios.

In the FleetPy configuration, t_{mw} is set to 600 seconds. In BS, the model's mean prediction error for this attribute is approximately 130 seconds. The best prediction result is observed in NPS-4, where one less planned stop is considered in the temporal domain, reducing the error to around 112 seconds. The model's prediction accuracy declines as the selection mechanism further scales down the data in the temporal domain. It is notable that reducing the number of planned stops considered has a more detrimental effect on the model's prediction accuracy than increasing the temporal domain information.

In the spatial domain, both reducing or increasing the number of vehicles considered by the model diminishes its predictive performance. Particularly in NV-20 and NV-200, insufficient fleet information fails to provide adequate support for the model's predictions, while excessive information dilutes the model's focus. This demonstrates that the selection mechanism can effectively filter out key information to aid the model in predicting accurate offer attribute values. Additionally, spatial information selection has a more significant impact on the model's prediction accuracy compared to temporal information selection.

In the dataset, the average travel time for all requests is about 644 seconds. In BS, the model's mean prediction error for this attribute is around 60 seconds. The best prediction result is again observed in NPS-4, with an error of approximately 52 seconds. The impact of the selection mechanism on the model's prediction accuracy for travel time, whether in the temporal or spatial domain, follows the same trend as its impact on the prediction accuracy for waiting time. The worst result is observed in NV-20, where the mean error reaches about 135 seconds.

The model's prediction results are compared with $\mathbf{O_P^{IR}}$. For travel time prediction, the IR's mean prediction error is approximately 85 seconds, while the proposed model's prediction error is consistently lower across most scenarios, with the smallest error of about 52 seconds in NPS-4.

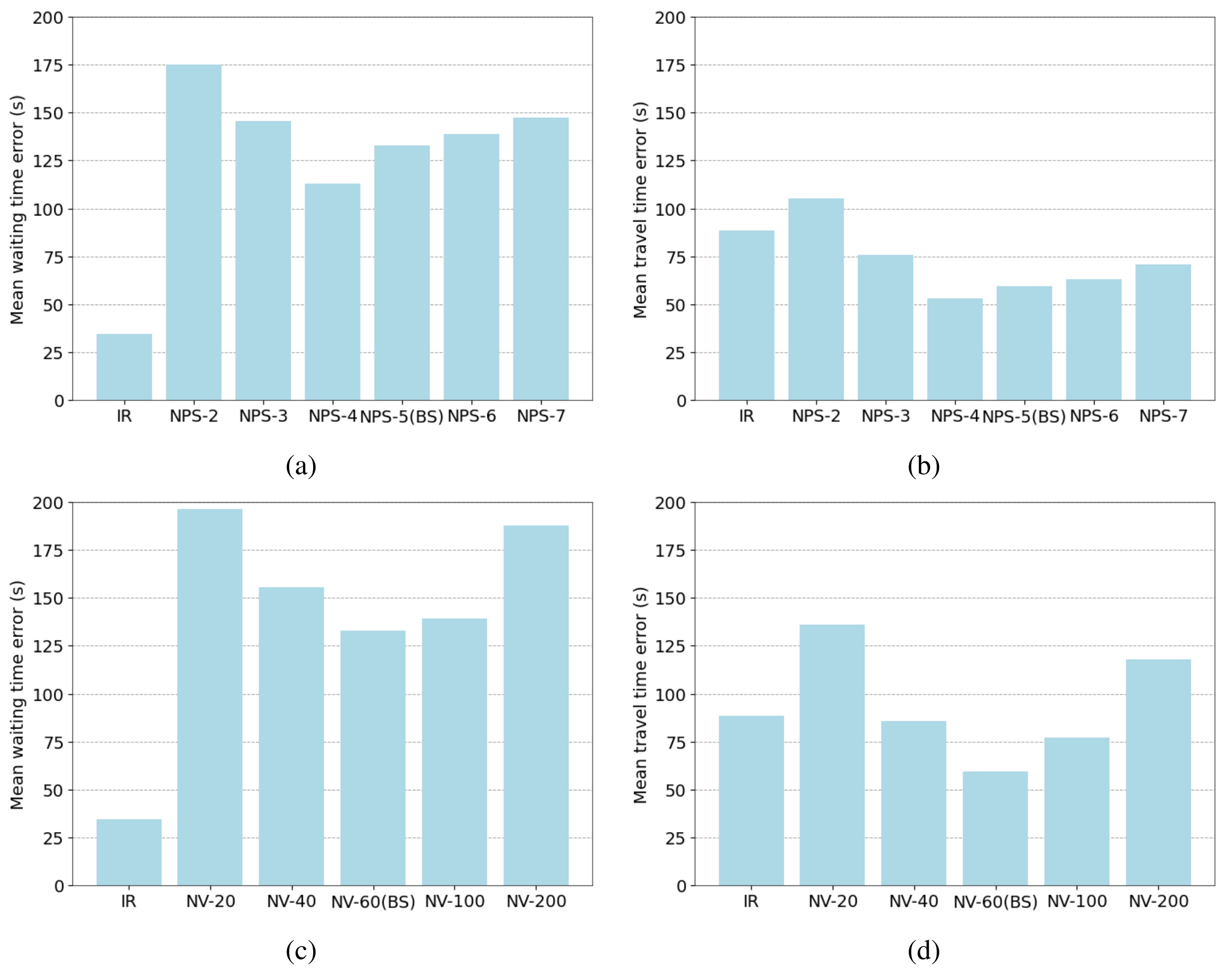
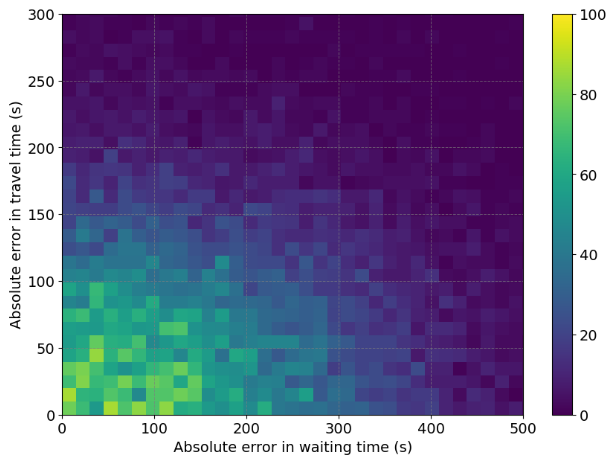


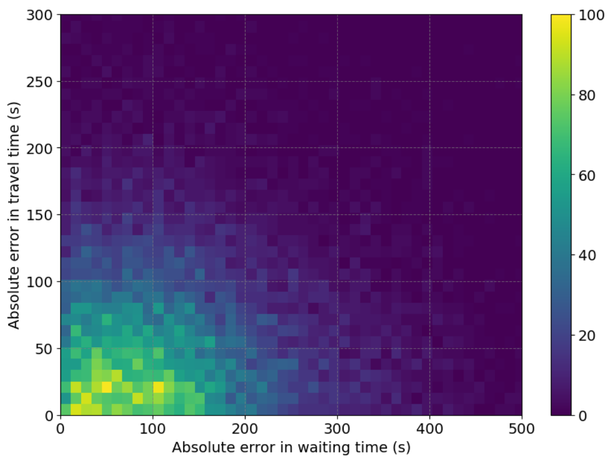
FIGURE 7: Comparison of mean errors in waiting time and travel time. Figures (a) and (b) show the mean errors of the model with different n_{mps} values; Figures (c) and (d) show the mean errors of the model with different n_{mv} values.

This indicates that the model outperforms the IR module in predicting travel time. For waiting time prediction, the IR's mean prediction error is around 37 seconds, whereas the proposed model's minimum prediction error is 112 seconds. This shows that the model's prediction accuracy for waiting time is not as good as that of the IR. Theoretically, waiting time is primarily influenced by $\mathbf{S_{pu}}$ in the fleet encoding information. However, in this study, the proposed model employs a relatively simple linear equation to calculate s_{pu}. This linear mapping may not fully reflect the complexities of real-world vehicle pick-up capabilities.

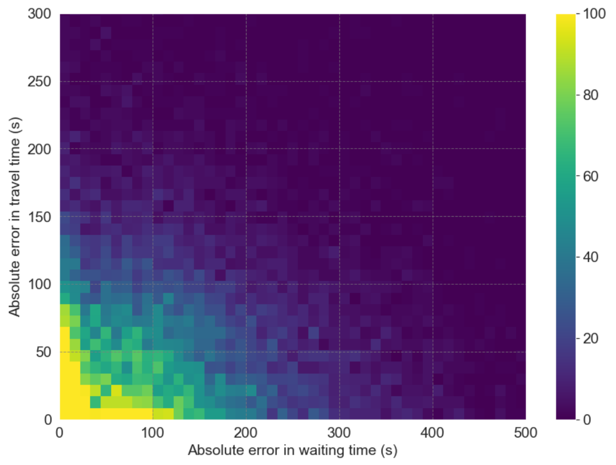
Figures 8 and 9 illustrate the joint distribution of absolute errors for waiting time and travel time at the individual offer level. The horizontal axis represents the absolute prediction error for waiting time, while the vertical axis represents the absolute prediction error for travel time for each offer. From both figures, it can be observed that scenarios with lower mean errors also have more concentrated joint absolute error distributions at the individual offer level. The heat map for NPS-4, in particular, shows the most concentrated highlighted area in the lower left corner, indicating that most offers have relatively low absolute errors for both attributes.



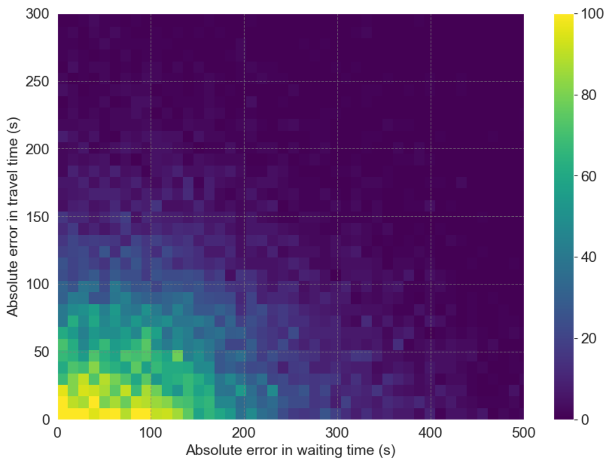
(a) NPS-2



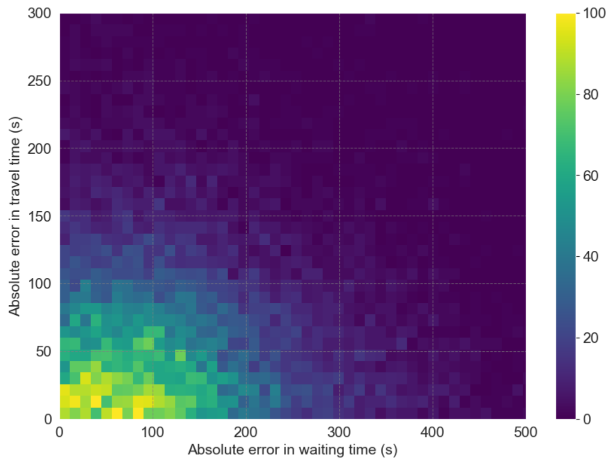
(b) NPS-3



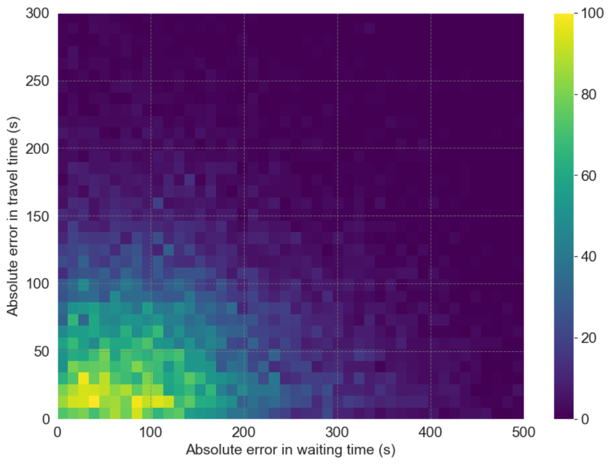
(c) NPS-4



(d) NPS-5 (BS)

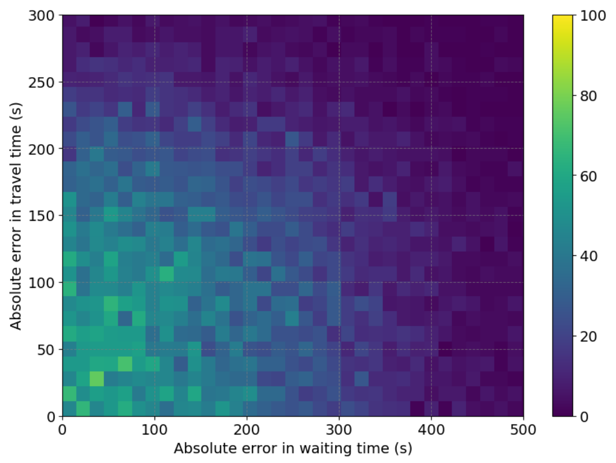


(e) NPS-6

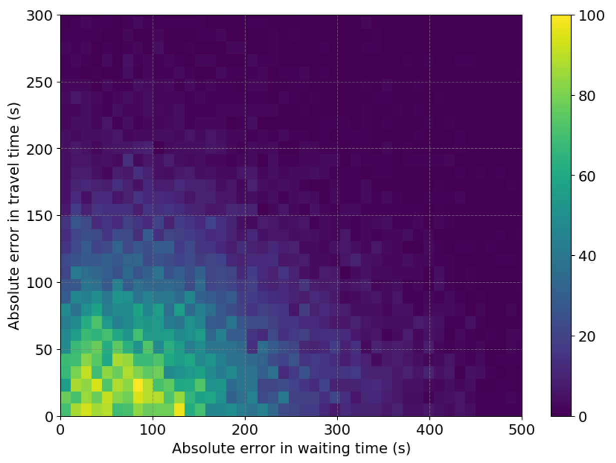


(f) NPS-7

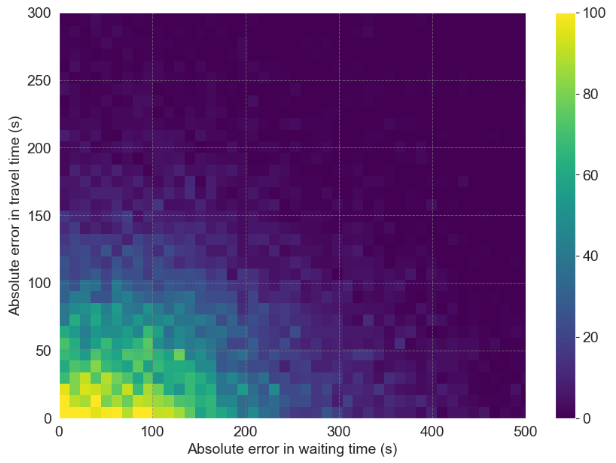
FIGURE 8: Impact of n_{mps} on model performance. Figures (a) to (f) present the joint distribution of absolute errors in the model's predictions across various scenarios characterized by different n_{mps} values.



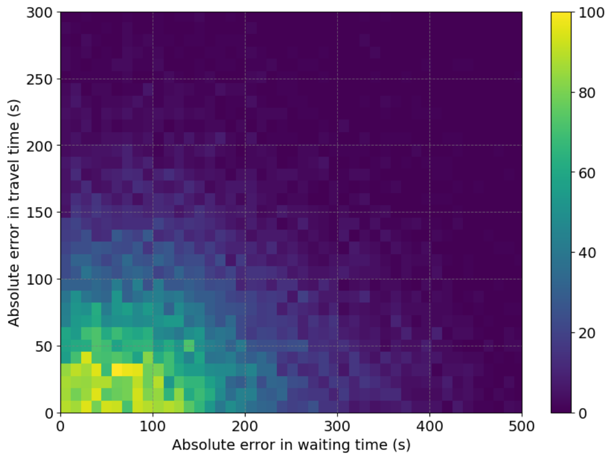
(a) NV-20



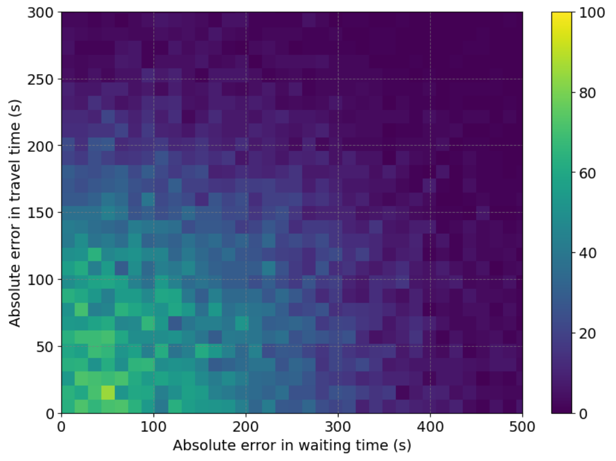
(b) NV-40



(c) NV-60 (BS)



(d) NV-100



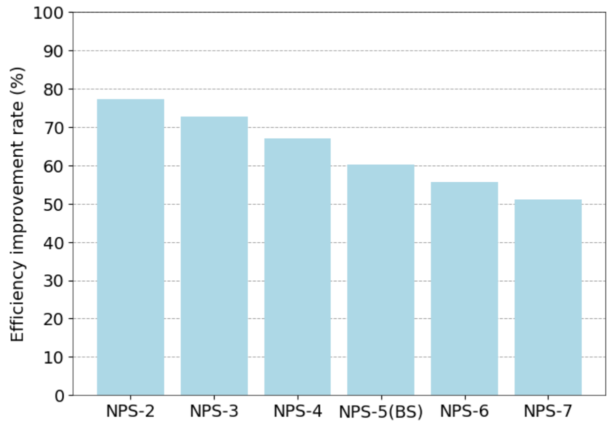
(e) NV-200

FIGURE 9: Impact of n_{mv} on model performance. Figures (a) to (f) present the joint distribution of absolute errors in the model's predictions across various scenarios characterized by different n_{mv} values.

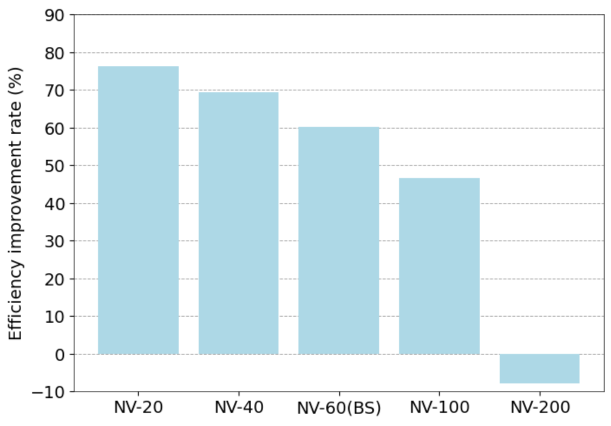
Efficiency
Another important indicator of the model's performance is efficiency. Using EIR, the predictive efficiency of the proposed model is compared to that of the IR module. Figure 10 presents the detailed comparison results.

Overall, by leveraging the selection mechanism to reduce the amount of information, the proposed model improves prediction efficiency by approximately 47% to 78% in most scenarios compared to the IR module. However, in NV-200, where the model needs to encode data from all vehicles, prediction efficiency is significantly impacted, resulting in an average prediction time of about 8% longer than that of the IR module. Notably, the model also does not achieve optimal prediction performance under NV-200.

The trend in efficiency changes indicates that reducing the amount of information in both the temporal and spatial domains generally enhances the model's prediction efficiency. However, considering prediction accuracy, indiscriminately reducing information is not recommended.



(a)



(b)

FIGURE 10: Comparison of EIR. (a) shows the EIR for the model across different values of n_{mps}; (b) shows the EIR for the model across different values of n_{mv}.

CONCLUSION AND FUTURE WORK

Ensuring the quality of services is crucial for TNCs like Uber, Lyft, and Didi, as it directly impacts user experience and revenue. A key challenge lies in providing fast and precise offer predictions based on user requests and fleet conditions. Response strategies solving vehicle routing problems for each incoming request struggle to deliver timely predictions in high-demand scenarios. The exponential growth in MoD demand and fleet sizes further complicates this task, underscoring the need for more efficient and scalable predictive methods.

This study proposes a machine learning-based model to effectively address these issues. The model encodes fleet state combined with user request information from three dimensions: pick-up likelihood, vehicle capacity, and detour degree, resulting in pick-up score, capacity score, and detour snapshot. By designing a selection mechanism, the model filters the encoded information in both temporal and spatial domains. This refined encoded information reduces the computational burden on the model, enhancing prediction efficiency and preventing the model's attention from being dispersed, thereby improving prediction accuracy. Given that offer prediction involves two aspects: identifying unserviceable requests and predicting specific offer attribute values, the model employs an encoder-decoder architecture. Fleet status is treated as an image, and a CNN is used in the encoder to embed image information. Different FCNs are then used to decode the embedded information, making various predictions.

Using FleetPy to simulate MoD services, a case study is conducted based on the Manhattan taxi dataset. Different scenarios are tested by controlling the selection mechanism's intensity in the temporal and spatial domains. Analysis and comparison of model results show that the selection mechanism critically impacts the model's predictive performance. Moreover, spatial domain information has a greater impact on the model's predictive performance than temporal domain information. When considering only 60 vehicles near the user and only the status of the first four stops for each vehicle, the model achieves a True Negative Rate of approximately 90% in identifying rejected requests, effectively recognizing requests the fleet cannot serve. Additionally, it outperforms the heuristic approach in both travel time prediction accuracy and prediction efficiency. However, the model's prediction error for waiting time is worse than the heuristic approach. This discrepancy is attributed to the model's use of a simple linear equation for encoding vehicle pick-up capability, which has limited expressive power.

Future research can improve the model's prediction performance for waiting time by using nonlinear encoding for vehicle pick-up capability. Additionally, different encoders and decoders can be tested to study the impact of various neural networks on the model's predictive ability. Furthermore, the model can be applied to different FleetPy configurations or tested under different networks and demands to examine its transferability. Finally, fleet control algorithms need to be adapted to work with a machine learning-based offer module; unlike the insertion heuristic-based offer module currently employed in FleetPy, machine learning-based approaches will accept requests, for which — according to the original constraints — no feasible vehicle-routing solution can be found.

ACKNOWLEDGMENTS
The authors acknowledge funding from the Transmove project, supported by the Federal Ministry for Digital and Transport (BMDV) of Germany and the Hamburg Authority for Traffic and Mobility Transition (BVM).

AUTHOR CONTRIBUTIONS
The authors confirm their contribution to the paper as follows: study conception and design: Chenhao Ding, Florian Dandl, Klaus Bogenberger; methodology and software: Chenhao Ding, Florian Dandl; data collection: Chenhao Ding; analysis and interpretation of results: Chenhao Ding, Florian Dandl; draft manuscript preparation: Chenhao Ding, Florian Dandl. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

1. Uber Technologies, Inc., *Uber Announces Results for Fourth Quarter and Full Year 2023*. Uber Technologies, Inc., San Francisco, 2024.
2. DiDi Global Inc., *DiDi Announces Results for Fourth Quarter and Full Year 2023*. DiDi Global Inc., Beijing, 2024.
3. Kumar, A., A. Gupta, M. Parida, and V. Chauhan, Service quality assessment of ride-sourcing services: A distinction between ride-hailing and ride-sharing services. *Transport Policy*, Vol. 127, 2022, pp. 61–79.
4. NYC Taxi & Limousine Commission, *TLC Trip Record Data*. NYC Taxi & Limousine Commission, New York, 2018.
5. Guo, G. and X. Li, A scientometric review of mobility-on-demand car-sharing systems. *IEEE Intelligent Transportation Systems Magazine*, Vol. 15, No. 1, 2022, pp. 212–229.
6. Shaheen, S. and A. Cohen, Mobility on demand (MOD) and mobility as a service (MaaS): Early understanding of shared mobility impacts and public transit partnerships. In *Demand for emerging transportation systems*, Elsevier, 2020, pp. 37–59.
7. Paparella, F., L. Pedroso, T. Hofman, and M. Salazar, A time-invariant network flow model for two-person ride-pooling mobility-on-demand. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, IEEE, 2023, pp. 4118–4123.
8. Pavone, M., S. L. Smith, E. Frazzoli, and D. Rus, Robotic load balancing for mobility-on-demand systems. *The International Journal of Robotics Research*, Vol. 31, No. 7, 2012, pp. 839–854.
9. Aimsun, S., *Aimsun Ride*, 2022, aimsun Ride Research Program.
10. Group, P. et al., *PTV MaaS Modeller*, 2022.
11. Ruch, C., S. Hörl, and E. Frazzoli, Amodeus, a simulation-based testbed for autonomous mobility-on-demand systems. In *2018 21st international conference on intelligent transportation systems (ITSC)*, IEEE, 2018, pp. 3639–3644.
12. Horni, A., K. Nagel, and K. W. Axhausen (eds.) *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, London, 2016.
13. Auld, J., M. Hope, H. Ley, V. Sokolov, B. Xu, and K. Zhang, POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. *Transportation Research Part C: Emerging Technologies*, Vol. 64, 2016, pp. 101–116.
14. Kucharski, R. and O. Cats, MaaSSim–agent-based two-sided mobility platform simulator. *arXiv preprint arXiv:2011.12827*, 2020.
15. Engelhardt, R., F. Dandl, A.-A. Syed, Y. Zhang, F. Fehn, F. Wolf, and K. Bogenberger, *FleetPy: A Modular Open-Source Simulation Tool for Mobility On-Demand Services*, 2022.
16. Fagnant, D. J. and K. M. Kockelman, The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies*, Vol. 40, 2014, pp. 1–13.
17. Zhang, W. and S. Guhathakurta, Parking spaces in the age of shared autonomous vehicles: How much parking will we need and where? *Transportation Research Record*, Vol. 2651, No. 1, 2017, pp. 80–91.

18. Alonso-Mora, J., S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, Vol. 114, No. 3, 2017, pp. 462–467.
19. Kucharski, R. and O. Cats, Exact matching of attractive shared rides (ExMAS) for system-wide strategic evaluations. *Transportation Research Part B: Methodological*, Vol. 139, 2020, pp. 285–310.
20. Engelhardt, R., F. Dandl, and K. Bogenberger, Speed-up heuristic for an on-demand ride-pooling algorithm. *arXiv preprint arXiv:2007.14877*, 2020.
21. Hyland, M. and H. S. Mahmassani, Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests. *Transportation Research Part C: Emerging Technologies*, Vol. 92, 2018, pp. 278–297.
22. Fleckenstein, D., R. Klein, and C. Steinhardt, Recent advances in integrating demand management and vehicle routing: A methodological review. *European Journal of Operational Research*, Vol. 306, No. 2, 2023, pp. 499–518.
23. Adnan, M., F. C. Pereira, C. M. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva, Simmobility: A multi-scale integrated agent-based simulation platform. In *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record*, The National Academies of Sciences, Engineering, and Medicine Washington, DC, 2016, Vol. 2.
24. Ben-Akiva, M. E. and S. R. Lerman, *Discrete choice analysis: theory and application to travel demand*, Vol. 9. MIT press, 1985.
25. Bengio, Y., A. Lodi, and A. Prouvost, Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, Vol. 290, No. 2, 2021, pp. 405–421.
26. Chen, L., P. Thakuriah, and K. Ampountolas, Short-term prediction of demand for ride-hailing services: A deep learning approach. *Journal of Big Data Analytics in Transportation*, Vol. 3, 2021, pp. 175–195.
27. Chu, K.-F., A. Y. Lam, and V. O. Li, Deep multi-scale convolutional LSTM network for travel demand and origin-destination predictions. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 21, No. 8, 2019, pp. 3219–3232.
28. Ke, J., H. Zheng, H. Yang, and X. M. Chen, Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation research part C: Emerging technologies*, Vol. 85, 2017, pp. 591–608.
29. Wang, D., Y. Yang, and S. Ning, DeepSTCL: A deep spatio-temporal ConvLSTM for travel demand prediction. In *2018 international joint conference on neural networks (IJCNN)*, IEEE, 2018, pp. 1–8.
30. Lei, Z., X. Qian, and S. V. Ukkusuri, Efficient proactive vehicle relocation for on-demand mobility service with recurrent neural networks. *Transportation Research Part C: Emerging Technologies*, Vol. 117, 2020, p. 102678.
31. Guo, G. and Y. Xu, A deep reinforcement learning approach to ride-sharing vehicle dispatching in autonomous mobility-on-demand systems. *IEEE Intelligent Transportation Systems Magazine*, Vol. 14, No. 1, 2020, pp. 128–140.
32. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need. *Advances in neural information processing systems*, Vol. 30, 2017.

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

34. Bengio, Y. et al., Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, Vol. 2, No. 1, 2009, pp. 1–127.

35. Engelhardt, R., P. Malcolm, F. Dandl, and K. Bogenberger, Competition and Cooperation of Autonomous Ridepooling Services: Game-Based Simulation of a Broker Concept. *Frontiers in Future Transportation*, Vol. 3, 2022, p. 915219.

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