*applied sciences*

MDPI

*Article*

# A Decision-Focused End-to-End Learning Framework for Scalable Vehicle Dispatching in Ride-Hailing Services

**Yuhan Guo *, Haoran Wang, Hongao Wang and Yiyang Qian**

School of Artificial Intelligence and Information Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China; haoranwang@zust.edu.cn (H.W.); 222409252007@zust.edu.cn (H.W.); 222309252032@zust.edu.cn (Y.Q.)

* Correspondence: yuhan.guo@zust.edu.cn

**Abstract**

Vehicle dispatching is essential for maintaining supply–demand balance in ride-hailing systems, but existing frameworks separate demand prediction from vehicle allocation, causing a mismatch between prediction accuracy and operational performance. This study proposes a decision-focused end-to-end learning framework that integrates spatiotemporal demand representation and vehicle-to-region dispatching within a unified trainable architecture. The model combines a multi-graph spatiotemporal prediction module with a recurrent Transformer-based dispatching module that iteratively refines vehicle assignments under capacity-aware constraints. A structured loss jointly considers demand prediction, dispatch objectives, and constraint violations, reducing dependence on expensive optimization labels. Experiments on real ride-hailing datasets from Chengdu, Xi'an, and Haikou show competitive dispatching quality and zero capacity-constraint violation rate (CCVR) in the reported city-level comparison. Relative to the Kuhn–Munkres benchmark, the proposed model reduces average reported online dispatching computation time from 242.47 s to 2.45 s, a 98.99% reduction (approximately 98.80-fold acceleration). The decision-oriented mechanism yields operationally useful demand representations, suggesting neural dispatchers offer a scalable approximation for high-frequency operations where responsiveness and feasibility are critical.

**Keywords:** decision-focused learning; ride-hailing; recurrent Transformer; vehicle dispatching; constraint-aware learning

## 1. Introduction

Ride-hailing platforms have become a core component of urban mobility by matching distributed travel demand with available vehicles [1–3]. Their dispatch systems must balance service coverage, vehicle utilization, passenger waiting time, and platform revenue, and these decisions become computationally demanding when updated at high frequency across many vehicles and regions [4,5].

Most practical systems follow a predict-then-dispatch workflow: future demand is first estimated from historical observations, and vehicles are then allocated by an optimization or heuristic procedure [6–8]. This modular design is interpretable, but its components are trained under different objectives—demand models minimize statistical error, whereas dispatching models optimize operational quantities such as coverage, revenue, or idle travel [9–11]. The prediction with the smallest standalone error is therefore not necessarily the representation that yields the most useful downstream allocation.

Check for updates

Academic Editor: Krzysztof Zboiński

Received: 18 September 2026
Revised: 16 October 2026
Accepted: 22 October 2026
Published: 27 October 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Decision-focused learning reduces this mismatch by allowing downstream decision losses to shape the predictive representation [12–14]. Reinforcement learning couples state estimation with sequential decisions but requires extensive interaction data and carefully designed rewards [15], while exact and heuristic optimization is too slow to re-solve at high frequency. The central challenge is thus to balance dispatch quality, constraint satisfaction, and online computation time rather than to dominate every baseline.

This study develops an End-to-End Dispatching Model (E2EDM), a decision-focused neural approximation that couples a multi-graph spatiotemporal demand module with a recurrent Transformer to iteratively refine vehicle-to-region assignments. Trained end-to-end with prediction, dispatch-objective, and capacity-constraint terms, it aligns its representations with downstream allocation. It targets dispatch-level decision support and does not model autonomous-vehicle constraints such as battery state of charge or charging availability.

The main contributions of this study are summarized as follows:

(1) We formulate a decision-oriented dispatching problem that unifies idle distance, revenue potential and a demand-weighted waiting-distance surrogate, and regional capacity feasibility in a single training objective.
(2) We develop an end-to-end prediction–dispatch framework that links spatiotemporal demand representation to downstream allocation, so predictions are optimized for dispatching utility rather than forecasting error alone.
(3) We design a multi-graph spatiotemporal prediction module that encodes spatial separation, functional similarity, and demand-pattern similarity among regions.
(4) We propose an encoder-only recurrent Transformer that non-autoregressively refines region-level assignment through full-attention global reasoning.
(5) We evaluate the framework on ride-hailing data from three cities, showing competitive dispatching quality, stable capacity feasibility, and substantially lower online computation time than the KM benchmark.

The remainder of this paper is organized as follows. Section 2 reviews related work, Section 3 formulates the problem, Section 4 presents the proposed framework, Section 5 reports the experiments, and Section 6 concludes.

## 2. Related Work

This section reviews solution algorithms for vehicle dispatching systems, emphasizing computational efficiency and deployment [16–18]. Existing methods fall into four categories—traditional optimization, heuristic, machine learning, and reinforcement learning—whose limitations motivate our decision-focused approach.

First, traditional optimization methods provide interpretable, reference-quality solutions when the problem structure is well defined, including mixed-integer programming for intertemporal resource allocation [19], routing and assignment [20], and adaptive KM matching [21]. However, repeatedly solving these models online becomes computationally burdensome as decision variables and dispatch windows grow.

Second, heuristic algorithms offer flexibility and rapid response at the cost of exact optimality. Representative work includes online insertion heuristics that anticipate future demand [22], hybrid assignment–optimization for dial-a-ride problems [23], non-myopic relocation incorporating long-term objectives [24], batch order dispatching with short-term demand [25], variable neighborhood search for demand-responsive transit and shared autonomous vehicles [26,27], hybrid multi-criteria frameworks for rural last-mile planning [28], and green or time-dependent routing solved by swarm-based metaheuristics [29–32]. Despite this flexibility, heuristics rely on hand-designed rules and domain knowledge [33,34], limiting

their adaptability and generalization in dynamic, heterogeneous urban environments [35,36]; these limitations motivate more data-driven alternatives [37,38].

Lee and Jeong studied accessible taxi routing by combining a Gaussian mixture model of passenger travel behavior with a vehicle routing problem with time windows, illustrating the importance of demand heterogeneity for routing decisions [39].

Third, data-driven machine learning methods learn dispatching policies directly from large-scale historical data, improving adaptability over rule-based heuristics [40–42]. Guo et al. [43,44] integrate data-driven demand estimation into optimization-based dispatching, classifying approaches into order matching, vehicle repositioning, and joint modeling. Such methods improve efficiency and accuracy but face scalability and generalization challenges in large-scale deployment.

In ride-hailing subsidy allocation, Yang et al. developed a decision-focused learning approach that links trip-completion prediction to downstream subsidy decisions, highlighting the limitations of optimizing predictive accuracy alone [45]. Unlike subsidy allocation, the present study addresses vehicle-to-region dispatching through a constraint-aware recurrent Transformer.

Fourth, reinforcement learning (RL) addresses sequential decision making through environment interaction [46]. Prior work formulates taxi dispatching as a Markov decision process with multi-agent assignment and repositioning [47–49], and hybrid methods combine deep RL with integer programming for assignment and rebalancing [50], while others incorporate passenger behavior and driver heterogeneity [51,52]. Shared-bicycle studies explore RL-based rebalancing and predictive scheduling [53,54], and large-scale implementations adopt multi-agent actor–critic frameworks [55]. These methods are expressive but their interaction requirements, training cost, and reward design complicate deployment.

Despite this progress, several challenges remain (Table 1). Specifically, most approaches exhibit the following limitations:

(1) Prediction and dispatching objectives are often optimized separately, though lower prediction error does not guarantee better allocation.
(2) Optimization and heuristic methods provide strong reference solutions, but their online cost grows with instance size and dispatch frequency.
(3) Learning-based dispatchers must balance operational objectives, capacity constraints, and scalable inference.

These gaps motivate a decision-focused neural approximation rather than a claim that learning should replace optimization. The proposed framework couples prediction and dispatching during training, incorporates capacity terms, and moves most computation offline, providing fast, constraint-aware assignments while retaining optimization and heuristic methods as high-quality references [19,56].

**Table 1.** Existing research analysis.

| Research | Method | Scalability | Multi-Objective | Structure | Heterogeneous |
|---|---|---|---|---|---|
| [22] | Heuristic | Medium | Yes | Pipeline | No |
| [23] | Heuristic + Optimization | Large | Yes | Pipeline | No |
| [24] | Heuristic | Medium | Yes | Pipeline | No |
| [26] | Heuristic (VNS) | Medium | Yes | Pipeline | No |
| [27] | Heuristic (VNS) | Medium | Yes | Pipeline | No |
| [28] | Heuristic + MCDM | Medium | Yes | Pipeline | Yes |
| [47] | Reinforcement Learning | Medium | Yes | Pipeline | No |
| [48] | Multi-Agent Reinforcement Learning | Medium | No | Pipeline | No |

**Table 1.** *Cont.*

| Research | Method | Scalability | Multi-Objective | Structure | Heterogeneous |
|---|---|---|---|---|---|
| [25] | Learning + Planning | Large | Yes | Pipeline | Yes |
| [42] | Deep Reinforcement Learning | Medium | Yes | Pipeline | No |
| [43] | Data-Driven Optimization | Large | Yes | Pipeline | Yes |
| [44] | Robust Optimization | Large | Yes | Pipeline | Yes |
| [50] | Reinforcement Learning + Integer Programming | Large | Yes | Pipeline | Yes |
| [40] | Offline Reinforcement Learning | Large | Yes | End-to-End | Yes |
| [8] | Reinforcement Learning | Large | Yes | End-to-End | No |
| [53] | Reinforcement Learning | Medium | Yes | Pipeline | No |
| [54] | Regression + Heuristic | Medium | No | Pipeline | No |
| **Ours** | **Deep Learning** | **Large** | **Yes** | **End-to-End** | **Yes** |

## 3. Problem Formulation

This section begins by introducing the fundamental definitions and problem formulation that underpin our approach. This is followed by a comprehensive description of the proposed vehicle dispatching model, which jointly considers multiple optimization objectives from both the platform and user perspectives, ensuring a balanced and holistic dispatching strategy.

**DEF 1. SERVICE BLOCK.** Let $S$ denote the given geographic region, with longitude boundaries $lng_{max}$ and $lng_{min}$, and latitude boundaries $lat_{max}$ and $lat_{min}$. The region is partitioned into $p \times q$ service blocks $s_{ij}$, as defined in Equations (1)–(3). In this partition, the length and width of each service block are given by

$$l_{lng}^{ij} = \frac{lng_{max} - lng_{min}}{p} \tag{1}$$

$$l_{lat}^{ij} = \frac{lat_{max} - lat_{min}}{q} \tag{2}$$

$$S = \left\{ s_{ij} \,\middle|\, i \in [1,p] \cap \mathbb{N},\ j \in [1,q] \cap \mathbb{N} \right\} \tag{3}$$

**DEF 2. TIME WINDOWS.** The 24 h day is evenly divided into $\delta$ intervals of equal duration, represented by the set $T = \{t_k \mid k \in [1,\delta] \cap \mathbb{N}\}$.

**DEF 3. SERVICEABLE VEHICLE.** All available service vehicles are represented by the set $V = \{v_n \mid n = 1,2,\ldots,C_N\}$, where $C_N$ denotes the total number of available vehicles.

**DEF 4. PASSENGER DEMAND.** Passenger demand is denoted by $Or_{t_k}^{s_{ij}}$, defined as the total number of user orders submitted within service block $s_{ij}$ during time window $t_k$.

**DEF 5. DECISION VARIABLE.** Let the time-indexed decision variable $M_{n,ij}^{t_k}$ indicate whether vehicle $v_n$ is assigned to service block $s_{ij}$ during time window $t_k$:

$$M_{n,ij}^{t_k} = \begin{cases} 1, & \text{if } v_n \text{ is assigned to } s_{ij} \text{ at } t_k, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We consider three dispatching objective components: vehicle idle distance, a demand-weighted revenue-potential surrogate, and a demand-weighted distance surrogate for passenger waiting. Their weighted combination is used as a differentiable decision-oriented training signal; it is not presented as an exact mathematical program solved online at every window:

$$\mathcal{J}^{t_k} = \sum_{n=1}^{C_N} \left( \lambda_1 \widetilde{C}_n^{t_k} - \lambda_2 \widetilde{I}_n^{t_k} + \lambda_3 \widetilde{T}_n^{t_k} \right) \tag{5}$$

Here, $C_n^{t_k}$, $I_n^{t_k}$, and $T_n^{t_k}$ denote the corresponding quantities for vehicle $n$ at time $t_k$. The coefficients $\lambda_1$, $\lambda_2$, and $\lambda_3$ control their relative importance. The symbols $\widetilde{C}$, $\widetilde{I}$, and $\widetilde{T}$ denote the individually Z-score-normalized objective components before weighted aggregation. For each objective component $X \in \{C, I, T\}$, the normalization is defined as $\widetilde{X} = (X - \mu_X)/\sigma_X$, where $\mu_X$ and $\sigma_X$ are the mean and standard deviation of that component, respectively, calculated from the training data and then held fixed for validation and testing. The approximately equal weights of 0.33 provide a reference configuration without privileging one objective; a systematic sensitivity analysis has not yet been performed.

Specifically,

$$C_n^{t_k} = \sum_{s_{ij} \in S} M_{n,ij}^{t_k} \cdot c_{n,ij}^{t_k} \tag{6}$$

$$I_n^{t_k} = \sum_{s_{ij} \in S} M_{n,ij}^{t_k} \cdot \frac{I_{aver} \cdot Or_{t_k}^{s_{ij}}}{\sum_{s_{ab} \in S} Or_{t_k}^{s_{ab}}} \tag{7}$$

$$T_n^{t_k} = \sum_{s_{ij} \in S} M_{n,ij}^{t_k} \cdot \frac{c_{n,ij}^{t_k} \cdot Or_{t_k}^{s_{ij}}}{\sum_{s_{ab} \in S} Or_{t_k}^{s_{ab}}} \tag{8}$$

where $c_{n,ij}^{t_k}$ denotes the distance traveled by vehicle $n$ to service block $s_{ij}$, and $I_{\text{aver}}$ represents the average revenue from historical orders. Equation (7) measures demand-weighted revenue potential, not realized platform revenue: successful order fulfillment, dynamic pricing, and direct competition among vehicles in the same block are not explicitly modeled. Equation (8) measures a demand-weighted distance surrogate, not actual passenger waiting duration; it does not incorporate speeds or passenger pickup delays. The three objectives are normalized separately before applying their coefficients.

The standard one-to-one assignment problem can be solved in polynomial time by Hungarian-type algorithms. However, when regional capacity constraints, multiple operational objectives, rolling time windows, and large online instances are considered together, repeatedly solving optimization or heuristic models can become computationally burdensome. This motivates learning a fast approximation that is evaluated against optimization-based references.

## 4. Methodology

We propose an End-to-End Prediction–Dispatch Model (E2EDM) composed of two modules: prediction module and dispatching module. Section 4.1 outlines the overall architecture, Section 4.2 details the prediction module, Section 4.3 presents the dispatching module, and Section 4.4 introduces the training objective and optimization procedure.

### *4.1. Overview of the End-to-End Dispatching Model*

The proposed model encompasses the full dispatching workflow, spanning from historical order data ingestion, through pattern learning, to final vehicle dispatching decisions and assignment outputs. The overall architecture is structured into two main stages, as illustrated in Figure 1.

#### 4.1.1. Offline Learning

During the offline learning phase, the prediction and dispatching modules are jointly optimized based on a grid-based spatial partitioning of the urban area and historical order data. Specifically, we construct multiple heterogeneous graph structures that capture service block adjacency and historical mobility patterns. A Multi-Graph Convolutional Network (MGCN) is employed to extract spatial dependency features across service blocks.

These spatial embeddings are then fed into an LSTM-based encoder–decoder architecture to model the temporal dynamics of demand evolution, enabling accurate prediction of order distributions across future time windows.

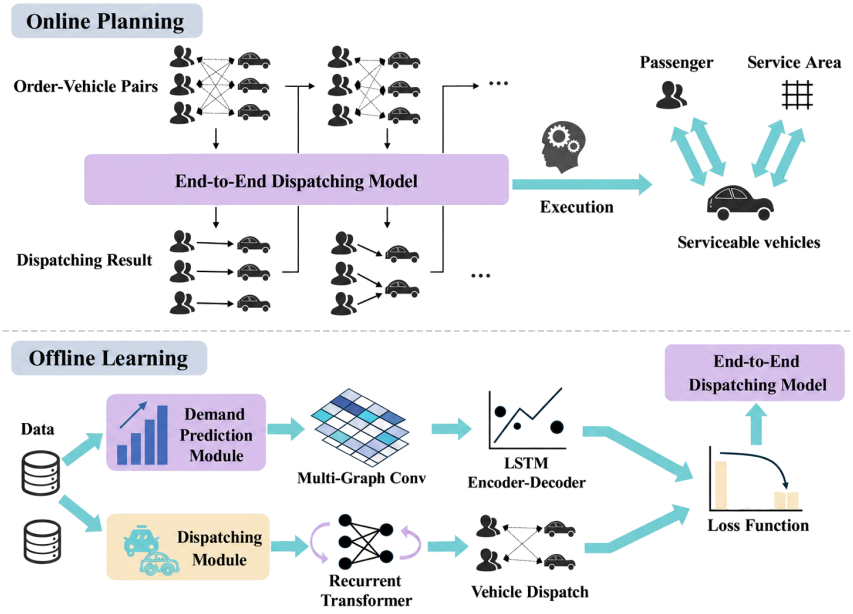



**Figure 1.** Overall model architecture.

The prediction outputs are embedded as structured inputs into the dispatching module, enabling the latter to generate vehicle-to-block assignment strategies that optimize platform-level objectives such as idle distance minimization, revenue maximization, and supply–demand balance. Unlike traditional pipeline designs, where prediction and dispatching are trained independently, our model implements a fully differentiable architecture wherein both modules are jointly optimized in an end-to-end manner.

Specifically, gradients from the dispatching loss can propagate through the Transformer layers into the upstream prediction module. The architecture is therefore end-to-end trainable and structurally coupled, allowing demand representations to be shaped by both prediction error and downstream dispatch objectives. The experiments evaluate the integrated framework, but do not claim that this coupling universally dominates independently trained prediction models.

Service-block capacity limits are incorporated through differentiable penalty terms. These terms encourage feasible assignments rather than mathematically guaranteeing feasibility for every possible input.

#### 4.1.2. Online Planning

In the online planning phase, the trained model predicts upcoming demand and generates vehicle-to-block assignments without repeatedly solving the KM reference problem. Most computation is shifted to offline training, while online inference remains at the seconds level on the reported hardware.

### *4.2. Prediction Module*

To capture prior knowledge of passenger demand pattern, we propose a Spatiotemporal Multi-Graph Convolution (STMGC) module that effectively models the complex spatial and temporal dependencies among service blocks. The structure of the module is illustrated in Figure 2. The module jointly captures spatial correlations between different regions and temporal dynamics in historical demand patterns, enabling more accurate demand

prediction. We propose a Spatiotemporal Multi-Graph Convolution (STMGC) module that jointly models spatial and temporal dependencies among service blocks (Figure 2), providing reliable demand representations for downstream dispatching.

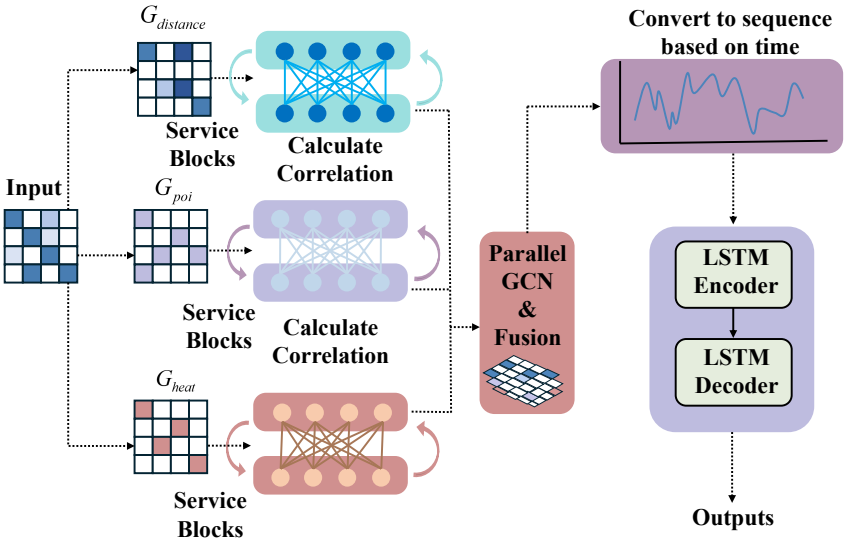



**Figure 2.** Structure of the prediction module.

Since a single graph cannot capture the multidimensional correlations among service blocks, we construct three graphs from distinct perspectives: geographical proximity, functional similarity, and demand heat similarity. Each graph encodes spatial relationships from a distinct perspective: geographical proximity, functional similarity, and historical demand heat similarity. This multi-view graph construction enables a more comprehensive and nuanced representation of the associative features among service blocks, thereby enhancing the model's capacity to capture realistic spatial interactions.

**GRAPH 1. DISTANCE–COST GRAPH.** The distance–cost graph records the spatial separation between service blocks. Each node corresponds to a grid cell, and a larger edge value indicates greater separation rather than stronger similarity. The subsequent normalization and learned transformations determine how this cost information contributes to the representation. Specifically, for any two blocks $s_i$ and $s_j$, their Euclidean distance is defined as follows:

$$G_{i,j}^{distance} = \sqrt{\left(lat_i - lat_j\right)^2 + \left(lng_i - lng_j\right)^2} \tag{9}$$

Here, $\text{lat}_i$ and $\text{lng}_i$ denote the centroid coordinates of block $i$. This graph provides explicit geographic cost information that complements the functional and historical-demand similarity graphs.

**GRAPH 2. POI SIMILARITY GRAPH.** Service blocks with similar functional attributes—such as commercial centers, residential neighborhoods, or entertainment districts, often exhibit analogous patterns of demand fluctuation. To quantify these functional similarities, we construct a Point-of-Interest (POI) similarity graph, in which edge weights reflect the degree of overlap in POI category distributions between blocks. This graph provides valuable contextual information that complements purely spatial and temporal signals, enhancing the model's ability to generalize across diverse urban environments. Specifically, for each block $s_i$, we generate a functional feature vector $f_i$ by aggregating the distribution of different categories of POIs within that block. The POI similarity between any two blocks is computed as shown in Equation (10):

$$G_{i,j}^{poi} = \frac{f_i \cdot f_j}{\|f_i\| \, \|f_j\|} \tag{10}$$

Here, $G_{i,j}^{poi}$ represents the POI similarity between service blocks $i$ and $j$, $f_i = [poi_{i1}, poi_{i2}, \ldots, poi_{ik}]$ denotes the POI feature vector of service block $i$, where $poi_{ik}$ is the count of POIs of category $k$ within block $i$.

All data-dependent preprocessing statistics and historical-demand correlations used to construct the heat graph are estimated only from the training period; the resulting graph structures remain fixed during validation and testing to prevent temporal information leakage.

**GRAPH 3. HEAT GRAPH.** The heat graph captures dependencies between service blocks that, although geographically distant and functionally dissimilar, exhibit highly similar patterns of demand evolution over time. For instance, a commercial district and a medical center—despite their spatial separation and distinct land-use functions—may display comparable temporal trends in order demand. To model these latent demand correlations, we construct a demand heat similarity graph, in which each node represents a service block and the edge weights are determined by the pairwise correlation between their historical passenger demand time series. This approach enables the model to identify and leverage functional analogies that are not evident from geographic or categorical features alone. By connecting service blocks with similar demand dynamics, the heat graph enhances the model's capacity to generalize and adapt to complex, real-world dispatching scenarios. Specifically, the heat graph is defined as shown in Equation (11):

$$G_{i,j}^{heat} = \frac{\mathrm{Cov}(Or^{s_i}, Or^{s_j})}{\sigma(Or^{s_i})\sigma(Or^{s_j}) + \epsilon} \tag{11}$$

Here, $Or^{s_i}$ and $Or^{s_j}$ denote the historical demand sequences of blocks $i$ and $j$, respectively. Equation (11) is the Pearson correlation coefficient, with a small $\epsilon$ added for numerical stability.

After constructing the three heterogeneous graphs: representing geographical proximity, functional similarity, and demand heat similarity—we perform graph convolution operations on each graph in parallel. This allows us to extract service block embeddings from multiple distinct perspectives simultaneously. The resulting multi-view embeddings are then aggregated through a feature fusion mechanism to generate a unified representation for each service block. This fused representation serves as the input to the downstream demand prediction module. By jointly modeling the heterogeneous graph structures, the proposed approach effectively enhances the model's ability to capture complex inter-block dependencies across spatial, functional, and temporal dimensions. This leads to more informative and robust feature representations, ultimately contributing to improved accuracy in predicting future demand patterns.

$$X^l = \sigma\left(AX^{(l-1)}W^{(l-1)}\right) \tag{12}$$

Here, $\sigma$ denotes the activation function, $A$ represents the adjacency matrix corresponding to the input graph, $X^{(l-1)}$ is the output of the previous layer, and $W^{(l-1)}$ denotes the weight matrix.

To extend the capability of the GCN for handling multi-view graph structures, we adopt an enhanced multi-graph convolution approach [57], which can be formulated as follows:

$$G(X^P, A^u) = \sum_u \hat{A}^u X^P W^P \tag{13}$$

where $\hat{A}^u = (D^u)^{-1/2}(I + A^u)(D^u)^{-1/2}$ denotes the symmetrically normalized adjacency matrix, and $A^u$ represents the adjacency matrix of each graph type.

To further integrate different types of service block relationship graphs, we employ a Multi-Graph Convolution (MGC) mechanism [58]. Specifically, for the constructed heterogeneous graphs, the MGC operation is defined as follows:

$$\mathrm{MGC}\left(X^P, G^{distance}, G^{poi}, G^{heat}\right) = \sum_{u \in \{distance,\, poi,\, heat\}} \left(G^u X^P W^P\right) \tag{14}$$

To improve computational efficiency, we adopt a Chebyshev polynomial approximation to replace the standard graph convolution operation, thereby accelerating the convolution process for each graph. This approximation is formulated as

$$\mathrm{GCN}(G^u, X^l) = X^{(l+1)} = \sigma\left(g(\theta_i, G^u) X^{(l)} W^l\right) \tag{15}$$

where $g(\theta_i, G^u) = \sum_{k=0}^{K} \theta_i T_i(L)$ represents the $K$-order Chebyshev polynomial function of the graph Laplacian, and $W^l$ denotes the feature transformation matrix. This mechanism significantly reduces computational complexity while preserving accurate feature propagation across heterogeneous graph structures, thereby enhancing the model's efficiency in encoding spatial dependencies.

To capture the temporal evolution patterns of block-level demand, we further incorporate an LSTM encoder–decoder architecture, which effectively models sequential dependencies over time. The architecture follows a sequence-to-sequence (Seq2Seq) framework, consisting of an encoder that captures historical temporal patterns and a decoder that generates future demand forecasts.

Specifically, the model first applies multi-graph convolution to historical order data across multiple time windows. Within each time window, graph convolutions are performed in parallel across the distance graph, POI similarity graph, and heat graph, extracting spatial features from distinct perspectives. These multi-view features are then aggregated through a weighted fusion operation, yielding a comprehensive spatial representation for each time window, denoted as $H^S_{t_k}$. For a sequence of $k$ time windows, this yields a spatial feature sequence:

$$H^S_t = \left\{H^S_{t_1}, H^S_{t_2}, \ldots, H^S_{t_k}\right\} \tag{16}$$

This sequence serves as the encoder input to capture long-term temporal dependencies in block features and is compressed into a fixed-length internal representation, which is passed as the hidden state to the decoder. The decoder then takes the encoder's final state as its initial state and sequentially predicts the output demand estimates:

$$\hat{Y} = \left\{\hat{y}_{t_{k+1}}, \ldots, \hat{y}_{t_{k+r}}\right\} \tag{17}$$

Algorithm 1 below outlines the procedure of the STMGC prediction module.

**Algorithm 1:** STMGC prediction module

**Input:** historical demand sequence $\{X^t\}$, multi-graph adjacency matrices $\{G^{distance}, G^{poi}, G^{heat}\}$
Initialize parameters of GCN layers and LSTM encoder
**for** $t = 1$ **to** $T$ **do**
  **for each** graph $G^k \in \{G^{distance}, G^{poi}, G^{heat}\}$ **do**
    Compute normalized adjacency matrix: $\hat{A}^u = (D^u)^{-1/2}(I + A^u)(D^u)^{-1/2}$
    Perform Chebyshev GCN convolution: $X^{l+1} = \sigma\left(g(\theta_i, G^u) X^{(l)} W^l\right)$
  **end for**
  Fuse features from all graphs: $H_t :=$ Combine$(G^{distance}, G^{poi}, G^{heat})$
**end for**
Get spatial-temporal sequence: $H_{seq} = \{H_{t_1}, H_{t_2}, \ldots, H_{t_k}\}$
Feed sequence into LSTM encoder to capture temporal patterns
Use decoder to predict future demand: $\hat{D}_t :=$ LSTM_Decoder$(H_{seq})$
**Output:** predicted demand sequence $\{\hat{D}_t\}$

### *4.3. Dispatching Module*

The dispatching module uses an encoder-only recurrent Transformer to represent coupled vehicle-to-region decisions. Unlike autoregressive sequence generation, all vehicle representations are updated in parallel with non-causal full attention. A recurrent inference mechanism feeds the hidden representation from one reasoning round into the next, enabling non-autoregressive iterative refinement of the assignment distribution.

The module has three features: (1) encoder-only parallel updates, where all vehicle states are updated simultaneously rather than autoregressively [59]; (2) recurrent refinement, where multiple self-attention rounds reuse the previous hidden state [60]; and (3) non-causal full attention over the global vehicle and block state, supporting coordinated reasoning (Figure 3).

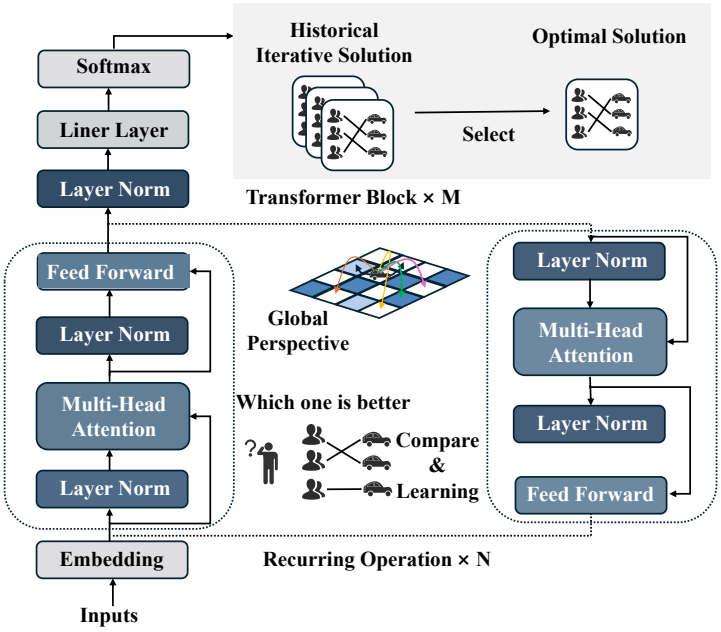



**Figure 3.** Structure of the dispatching module.

In the modeling process, the dispatching module receives two key input matrices: the demand distribution matrix $P_d \in \mathbb{R}^{m \times m}$, which records the demand information for each service block within a specified time window. This matrix is used to enforce column constraints, ensuring that the number of vehicles assigned to each block does not exceed its corresponding demand capacity. Additionally, the vehicle distribution matrix $P_v \in \mathbb{R}^{n \times m^2}$ captures the current vehicle distribution status.

We first embed the vehicle distribution into token and positional representations Emtoken and Empos, where the former projects vehicle features into a shared latent space and the latter preserves each vehicle's identity and relative position. Meanwhile, $\mathrm{Em}_{\mathrm{pos}}$ is a randomly initialized positional encoding vector of dimension $l_{pq}$, used to preserve the identity and relative position of each vehicle within the fleet, ensuring that the model can distinguish between different vehicles.

The proposed Recurrent-Transformer model adopts a stacked architecture integrated with a recursive looping mechanism, consisting of $M$ self-attention blocks and $N$ recursive iterations. Through this multi-step reasoning framework, the model iteratively refines dispatching decisions, progressively improving the alignment between vehicle assignments and dynamic demand patterns:

$$H^{(n,0)} = H^{(n-1,M)} \tag{18}$$

$$H^{(n,m)} = \text{Block}_m\left(H^{(n,m-1)}\right), \quad \forall m \in \{1,\ldots,M\} \tag{19}$$

$$X^{(n,m)} = \text{softmax}\left(\ln\left(H^{(n,m)}\right) \cdot \lambda_{out}\right), \quad \forall m \in \{1,\ldots,M\} \tag{20}$$

Here, $H^{(n,0)}$ denotes the initial hidden state of the $n$-th recursive iteration. In particular, the initial hidden embedding input is defined as $H^{(1,0)} = \text{Em}_{\text{token}} + \text{Em}_{\text{pos}}$. $\text{Block}_m$ represents the $m$-th Transformer block, which performs self-attention mechanisms and feedforward network operations at each layer. The operator $\ln(\cdot) = \text{layer_norm}(\cdot)$ denotes layer normalization, which stabilizes the training process. $\lambda_{\text{out}} \in \mathbb{R}^{l_{pq} \times c}$ denotes the output layer weights, and $X^{(n,m)} \in \mathbb{R}^{t \times c}$ denotes the neural network outputs.

For each dispatching instance, the set of idle vehicles eligible for repositioning is specified by the input scenario. The dispatching module takes this predefined set as input and assigns target service blocks to the participating vehicles; it does not perform a separate vehicle-selection procedure. Other idle vehicles remain at their current locations. For each participating vehicle, the final probabilistic output is converted to a discrete target block by a vehicle-wise argmax operation. The decoder does not itself impose a strict block-capacity guarantee; constraint penalties during training discourage capacity violations, and no separate capacity-repair algorithm is applied in the reported decoding.

To enhance the model's ability to capture diverse subspace features within the global system state, we incorporate the multi-head attention mechanism, which enables the model to jointly attend to information from different representation subspaces.

Assume that the hidden state is linearly projected into h attention heads, each with a reduced feature dimension of

$$l_h = \frac{l_{pq}}{h} \tag{21}$$

For the $k$-th attention head, in the $n$-th recursive iteration and the $m$-th Transformer block, the $Q_k^{(n,m)}$, $K_k^{(n,m)}$, and $V_k^{(n,m)}$ representations are defined as follows:

$$Q_k^{(n,m)} = \ln\left(H^{(n,m)}\right) \cdot W_Q^{(m,k)} \tag{22}$$

$$K_k^{(n,m)} = \ln\left(H^{(n,m)}\right) \cdot W_K^{(m,k)} \tag{23}$$

$$V_k^{(n,m)} = \ln\left(H^{(n,m)}\right) \cdot W_V^{(m,k)} \tag{24}$$

The scaled dot-product attention weights are defined as follows:

$$A_k^{(n,m)} = \text{softmax}\left(\frac{Q_k^{(n,m)}\left(K_k^{(n,m)}\right)^T}{\sqrt{l_h}}\right), \quad k \in \{1,\ldots,h\} \tag{25}$$

The attention output and its projected head representation are then computed as

$$\begin{aligned} Z_k^{(n,m)} &= A_k^{(n,m)} V_k^{(n,m)}, \\ V_k^* &= Z_k^{(n,m)} W_P^{(m)}, \quad k \in \{1,\ldots,h\} \end{aligned} \tag{26}$$

By concatenating the outputs of all attention heads and applying a linear transformation, we derive the aggregated representation:

$$V^* = \text{Concat}(V_1^*,\ldots,V_k^*) \cdot W_O, \quad k \in \{1,\ldots,h\} \tag{27}$$

Finally, the overall update operation of each Transformer block is expressed as follows:

$$\text{Block}_m\left(H^{(n,m)}\right) = \text{MLP}_m(\ln(V^*)) + V^* \tag{28}$$

Here, the subscript $k$ indicates the query $Q_k^{(n,m)}$, key $K_k^{(n,m)}$, and value $V_k^{(n,m)}$ matrices corresponding to the $k$-th attention head. $A_k^{(n,m)}$ denotes the attention-weight matrix, and $Z_k^{(n,m)}$ is the corresponding value-weighted attention output. $\sqrt{l_h}$ denotes the dimensionality of the key vectors and scales the dot product to improve numerical stability. $W_P^{(m)}$ projects each head output, while $W_O$ projects the concatenated multi-head representation into the desired output dimension.

This multi-head attention design allows the model to simultaneously model vehicle–vehicle interactions, block–block correlations, and cross-entity dependencies, significantly enhancing its capacity to perform global collaborative reasoning in complex dispatching scenarios. Algorithm 2 below outlines the procedure of the Recurrent-Transformer dispatching module.

**Algorithm 2:** Recurrent-Transformer Dispatching Module

**Input:** Vehicle feature set $V$ with features (location, idle time, capacity), region set with predicted demand $\hat{D}_t$, recursion step $N$, transformer block number $M$.
Token embedding matrices $Em_{\text{token}}$ and $Em_{\text{pos}}$.
Set $H^{(1,0)} = Em_{\text{token}} + Em_{\text{pos}}$.
**For** $n = 1$ **to** $N$ **do**
  **If** $n > 1$, set $H^{(n,0)} = H^{(n-1,M)}$.
  **For** $m = 1$ **to** $M$ **do**
    **For each** head $k = 1$ **to** $h$ **do**
      $Q_k^{(n,m)} = \ln\left(H^{(n,m)}\right) W_Q^{(m,k)}$.
      $K_k^{(n,m)} = \ln\left(H^{(n,m)}\right) W_K^{(m,k)}$.
      $V_k^{(n,m)} = \ln\left(H^{(n,m)}\right) W_V^{(m,k)}$.
      $A_k^{(n,m)} = \text{softmax}\left(Q_k^{(n,m)} (K_k^{(n,m)})^\top / \sqrt{l_h}\right)$.
      $Z_k^{(n,m)} = A_k^{(n,m)} V_k^{(n,m)}$; $V_k^* = Z_k^{(n,m)} W_P^{(m)}$.
    **end for**
    $V^* = \text{Concat}\left(V_1^*, \ldots, V_h^*\right)$.
    $X^{(n,m)} = \text{softmax}\left(\text{LayerNorm}\left(H^{(n,m)}\right) \lambda_{\text{out}}\right)$.
  **end for**
**end for**
**Output:** Final dispatch decision matrix $X^{(n,m)}$.

### *4.4. Loss Function*

To jointly optimize demand prediction, dispatch objectives, and constraint penalties, we design a structured loss that supports two training strategies: KM-guided optimization (KMGO) and constraint-guided optimization (CGO). KMGO uses KM-generated reference assignments as supervised guidance, whereas CGO relies on the differentiable dispatch objective and structural constraint penalties without requiring KM labels.

The first component of the loss function is the Prediction Loss $L_{\text{demand}}$, which quantifies the model's performance in predicting future demand trends. This is implemented using the Mean Squared Error (MSE) between the predicted and actual demand distributions across service blocks:

$$L_{demand} = \text{MSE}_{\text{Loss}} = \frac{1}{k} \sum_{t=1}^{k} (y_t - \hat{y}_t)^2 \tag{29}$$

where $\hat{y}_t$ and $y_t$ denote the predicted and ground-truth demand values, and $k$ is the total number of service blocks.

To optimize the vehicle dispatching module, we define two complementary types of dispatching loss that accommodate different training scenarios and data availability:

**(1) KM-guided optimization loss.** When KM reference assignments are available, the model minimizes the discrepancy between the objective values of its predicted assignment and the reference assignment.

**(2) Constraint-guided optimization loss.** In CGO, reference assignments are not used. The model is guided by the differentiable dispatch objective and structural constraint penalties. This reduces dependence on optimization labels, although it does not constitute a proof of optimality.

Let $la$ represent the reference assignment indicating the target service block for each vehicle. Given the model's predicted dispatching plan $P$, we evaluate the quality of the dispatching solution based on idle distance $C$, a revenue-potential surrogate $I$, and a waiting-distance surrogate $T$, rather than directly observed revenue or waiting duration.

$$O(P) = \big(C(P), -I(P), T(P)\big) \tag{30}$$

Accordingly, in the supervised setting, the single-step dispatch loss is defined as the difference between the expected objective value derived from the model's output and the corresponding value from the reference assignment:

$$L_{opt_{single_{km}}}(P, la) = \|O(P) - O(la)\|_2 \tag{31}$$

In the absence of reference assignments, we define the single-step unsupervised dispatch loss as

$$L_{opt_{single_{none}}}(P) = \lambda_1 \widetilde{C}(P) - \lambda_2 \widetilde{I}(P) + \lambda_3 \widetilde{T}(P) \tag{32}$$

To account for the recursive nature of the Recurrent Transformer architecture, we accumulate the single-step losses across all recursive iterations and Transformer blocks to compute the overall dispatch loss:

$$L_{dispatch_{km}} = L\Big(X^{(n,m)}, la\Big) = \sum_{n=1}^{N}\sum_{m=1}^{M} L_{opt_{single_{km}}}\Big(X^{(n,m)}, la\Big) \tag{33}$$

$$L_{dispatch_{none}} = L\Big(X^{(n,m)}\Big) = \sum_{n=1}^{N}\sum_{m=1}^{M} L_{opt_{single_{none}}}\Big(X^{(n,m)}\Big) \tag{34}$$

When applying unsupervised loss, to ensure that the dispatching solutions adhere to real-world operational constraints, we incorporate a Constraint Embedding Mechanism, which introduces explicit structural loss terms to encode key scheduling rules:

**(1) Row Constraint (Vehicle Single Assignment):** Each dispatched vehicle is assigned to one service block.

$$L_{row} = \sum_{i=1}^{n}\left(\sum_{j=1}^{m} P(X_i = j) - 1\right)^2, \quad \forall i \in \{1, 2, \ldots, n\} \tag{35}$$

**(2) Column Constraint (Block Capacity Limitation):** The number of vehicles assigned to block $j$ must not exceed its demand capacity $Or_j$.

$$L_{col} = \sum_{j=1}^{m}\left[\mathrm{ReLU}\left(\sum_{i=1}^{n} P(X_i = j) - Or_j\right)\right]^2 \tag{36}$$

As softmax is applied across candidate blocks for each vehicle, row probabilities already sum to one. Thus, the row loss in Equation (35) is theoretically redundant under this normalization and explicitly records the single-assignment condition; the column term supplies the principal capacity-related training penalty.

The overall constraint loss $L_{constraint}$ is then defined as a weighted combination:

$$L_{constraint} = \lambda_{row} L_{row} + \lambda_{col} L_{col} \tag{37}$$

where $\lambda_{row}$ and $\lambda_{col}$ are hyperparameters balancing the influence of each constraint.

The total loss is a weighted combination of all components:

$$L_{kmgo} = \lambda_d L_{demand} + \lambda_c L_{dispatch_{km}} \tag{38}$$

$$L_{cgo} = \lambda_d L_{demand} + \lambda_c L_{dispatch_{none}} + \lambda_{row} L_{row} + \lambda_{col} L_{col} \tag{39}$$

where $L_{kmgo}$ or $L_{cgo}$ is applied depending on the training strategy, and $\lambda_d$ and $\lambda_c$ are coefficients controlling the relative importance of each component.

The constraint penalties encourage single assignments and capacity-aware allocations. Their empirical contribution is assessed by the loss ablation in Section 5.2; no hard feasibility guarantee is assumed.

## 5. Experiments

This section evaluates prediction accuracy, dispatch quality, constraint behavior, and online computation time on three real-world ride-hailing datasets.

The experimental data are anonymized ride-hailing order records from Didi Chuxing, collected in three cities—Chengdu, Xi'an, and Haikou—including order timestamps, pickup and drop-off locations, trip durations and distances, and vehicle status. Supplementary geographical information was obtained using the Baidu Maps Open Platform API (https://lbsyun.baidu.com/), which is provided as a public web service without a separately specified software version. Table 2 summarizes the study periods, order and vehicle counts, and spatial partitions. The datasets were split chronologically by collection date into training (70%), validation (10%), and test (20%) subsets. Each demand prediction uses 12 historical 30 min windows (6 h) to predict the next 30 min window. Experiments used Windows 10, Python 3.10, PyTorch 2.1.0, an Intel Core i7-14600KF CPU, an NVIDIA RTX 4060 Ti GPU, and 32 GB RAM. Table 3 presents the hyperparameter configuration.

**Table 2.** Dataset statistics and experimental spatial–temporal settings.

| Characteristic | Chengdu | Xi'an | Haikou |
|---|---|---|---|
| Study period | 1–30 October 2021 | 1–30 October 2021 | 1 October–1 November 2021 |
| Order records | 9,542,273 | 5,509,214 | 2,477,150 |
| Unique vehicles | 42,664 | 14,973 | 8947 |
| Spatial grids | $16 \times 16$ | $16 \times 16$ | $10 \times 10$ |
| Service blocks | 256 | 256 | 100 |
| Time-window duration | 30 min | 30 min | 30 min |
| Train/validation/test | 70/10/20% | 70/10/20% | 70/10/20% |

**Table 3.** Hyperparameter configuration.

| Parameter | Value |
|---|---|
| Optimizer | Adam |
| Learning rate | $1 \times 10^{-5}$ |
| Batch size | 32 |
| Total epochs | 500 |
| Transformer blocks ($M$) | 4 |
| Recursive steps ($N$) | 16 |
| $\lambda_1, \lambda_2, \lambda_3$ | 0.33 |
| $\lambda_d$ | 1 |
| $\lambda_c$ | 0.5 |
| $\lambda_{row}$ | 1 |
| $\lambda_{col}$ | 1 |

### *5.1. Evaluation of Prediction Module*

We evaluate the demand prediction module using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), comparing against statistical, machine learning, and deep learning baselines.

Table 4 and Figure 4 summarize the prediction results. The proposed model achieves competitive accuracy across the three cities and improves substantially over the statistical and several machine-learning baselines. ASTGCN, however, obtains lower MAE and RMSE in all three datasets. This result is expected because ASTGCN is optimized specifically for standalone forecasting, whereas the proposed prediction module is trained within a decision-oriented objective that also reflects downstream dispatching. Accordingly, Table 4 should be interpreted as evidence that the integrated model retains a reasonable forecasting capability, not as evidence of best-in-class prediction accuracy.

**Table 4.** Prediction performance analysis across different models.

| Method | Chengdu | | Xi'an | | Haikou | |
|---|---|---|---|---|---|---|
| | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** |
| Linear regression | 9.48 | 9.67 | 8.64 | 9.11 | 7.56 | 10.33 |
| XGBoost | 4.84 | 6.94 | 3.40 | 5.16 | 4.15 | 8.63 |
| CatBoost | 5.13 | 7.16 | 4.38 | 5.62 | 5.24 | 7.93 |
| LSTM | 4.61 | 6.81 | 3.27 | 5.28 | 4.16 | 7.23 |
| LSTM-KNN | 4.52 | 6.59 | 2.99 | 5.11 | 3.27 | 6.91 |
| ASTGCN | 4.32 | 6.33 | 2.41 | 4.36 | 2.12 | 6.51 |
| **Ours** | 4.48 | 6.45 | 2.55 | 4.82 | 2.46 | 6.57 |

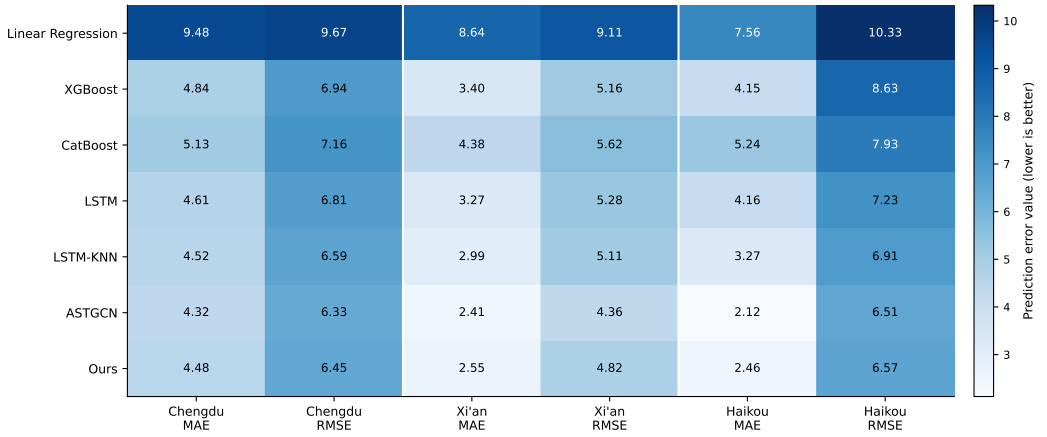



**Figure 4.** Prediction error comparison across different models; lower values are better.

Table 5 reports the graph ablation. Removing the distance–cost graph degrades both MAE and RMSE, indicating that geographic separation is important. Removing the POI or heat graph increases MAE but decreases RMSE. Specifically, the full model achieves the lowest MAE (2.46), whereas excluding the POI and heat graphs reduces RMSE from 6.57 to 4.78 and 5.36, respectively. This indicates a trade-off between average absolute error and sensitivity to larger prediction errors; incorporating additional spatial relationships does not necessarily improve every forecasting metric. Multi-graph fusion is interpreted here as a decision-oriented representation design rather than a mechanism that minimizes every standalone forecasting metric. Because RMSE places greater weight on large errors, this pattern raises a potential concern for demand spikes or individual service blocks, where localized forecasting errors may affect capacity-aware dispatch. The reported ablation does not identify the locations of these errors or establish which graph configuration yields the best downstream dispatch performance.

**Table 5.** Graph ablation experimental results.

| Ablation Setting | MAE | RMSE |
|---|---|---|
| Full model (all graphs) | 2.46 | 6.57 |
| w/o distance–cost graph | 4.39 | 7.02 |
| w/o POI graph | 2.89 | 4.78 |
| w/o heat graph | 3.31 | 5.36 |

### *5.2. Evaluation of Dispatching Module*

This subsection evaluates the learned dispatcher on representative instances from Chengdu, Xi'an, and Haikou. The comparison baselines include the Kuhn–Munkres (KM) algorithm, the prediction-based Iterative Kuhn–Munkres (IKM) approach [61], Fast Solution Reconstruction (FSR) [3], and Adaptive Variable Neighborhood Search (AVNS) [62]. KM provides a reference solution for the underlying bipartite matching component. Because the complete evaluation also includes regional capacity and supply–demand metrics, KM is treated as a benchmark rather than a globally optimal solution to every operational objective considered in this study.

Figure 5 illustrates one dispatching instance on a $10 \times 10$ grid. The learned allocation broadly follows the spatial pattern of the KM reference while not reproducing it exactly. The example is qualitative; the aggregate comparisons in Tables 6 and 7 provide the quantitative evidence.

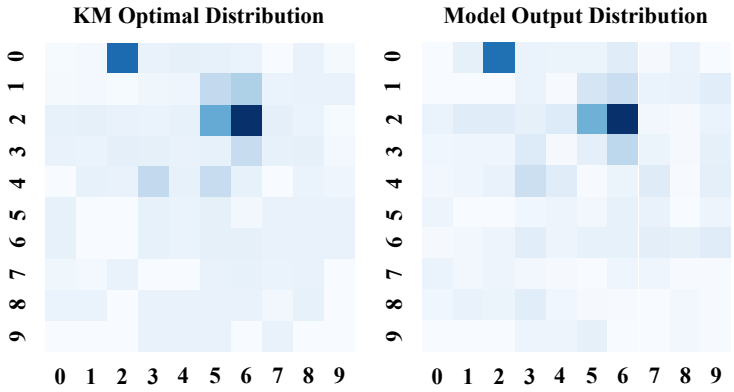



**Figure 5.** Specific dispatching scenario demonstration.

To further quantify the model's performance, we define the supply–demand equilibrium ratio (SDER) and the capacity constraint violation rate (CCVR) as assessment metrics. Let $D_j$ be the final discretely assigned vehicle count, $O_j$ the order demand in service block $j$, and $S^+ = \{j : O_j > 0\}$. The SDER is evaluated as

$$\mathrm{SDER} = \frac{1}{|S^+|} \sum_{j \in S^+} \min\left(1, \frac{D_j}{O_j}\right). \tag{40}$$

Blocks with zero demand are excluded, and supply in excess of demand is capped at full coverage. Thus, SDER indicates aggregate regional demand coverage; it is not, by itself, a measure of fairness between passenger groups or robustness to demand shocks. The CCVR is defined as $\mathrm{CCVR} = \frac{\sum_{i=1}^{N} \max(0, s_i - C_i)}{\sum_{i=1}^{N} C_i}$, where $s_i$ counts vehicles in the final discrete assignment after argmax decoding, rather than summing assignment probabilities, and $C_i$ is the block capacity.

As shown in Table 6, the proposed model achieves an average SDER of 92.95% across the three cities and maintains zero CCVR in all cases. It improves SDER over KM, IKM, and AVNS, while FSR obtains higher SDER in Chengdu and Xi'an and the proposed model

obtains the highest value in Haikou. The results therefore support competitive dispatch quality and stable capacity behavior, rather than strict dominance over every baseline. The zero CCVR is consistent with the use of capacity-aware loss terms.

**Table 6.** Dispatch performance comparison across methods and cities. Bold values indicate the best SDER results for each city.

| Method | Chengdu | | Xi'an | | Haikou | |
|---|---|---|---|---|---|---|
| | SDER | CCVR | SDER | CCVR | SDER | CCVR |
| KM | 78.48% | 1% | 79.53% | 0% | 81.42% | 4% |
| IKM | 82.35% | 0% | 84.65% | 0% | 85.36% | 1% |
| FSR | **93.12%** | 0% | **94.71%** | 0% | 91.38% | 0% |
| AVNS | 90.69% | 0% | 92.25% | 0% | 90.16% | 0% |
| ASTGCN + Dispatcher | 91.48% | 0% | 92.25% | 0% | 90.74% | 0% |
| **Ours** | 92.61% | 0% | 93.92% | 0% | **92.33%** | 0% |

**Table 7.** Objective function value analysis.

| Instance | Method | Idle Distance | Revenue Surrogate | Waiting-Distance Surrogate |
|---|---|---|---|---|
| 1 | KM | 4520 | 1283 | 903 |
| | KMGO | 4671 | 1371 | 934 |
| | CGO | 4710 | 1278 | 958 |
| 2 | KM | 5250 | 1431 | 955 |
| | KMGO | 5349 | 1402 | 973 |
| | CGO | 5315 | 1387 | 959 |
| 3 | KM | 6120 | 1652 | 1020 |
| | KMGO | 6230 | 1569 | 1038 |
| | CGO | 6285 | 1569 | 1047 |
| 4 | KM | 6221 | 1665 | 1037 |
| | KMGO | 6214 | 1643 | 1036 |
| | CGO | 6236 | 1650 | 1042 |
| 5 | KM | 7723 | 2069 | 1331 |
| | KMGO | 7741 | 2023 | 1336 |
| | CGO | 7763 | 2088 | 1341 |
| 6 | KM | 8109 | 2161 | 1329 |
| | KMGO | 8238 | 2138 | 1350 |
| | CGO | 8182 | 2158 | 1335 |
| 7 | KM | 9341 | 2413 | 1531 |
| | KMGO | 9527 | 2377 | 1562 |
| | CGO | 9386 | 2295 | 1560 |
| 8 | KM | 11,233 | 3009 | 2340 |
| | KMGO | 11,187 | 2947 | 2326 |
| | CGO | 11,225 | 2981 | 2335 |

For a direct prediction–dispatch comparison, independently trained ASTGCN forecasts are supplied to the same recurrent Transformer dispatcher. Despite ASTGCN's lower standalone forecasting MAE (Table 4), the jointly trained E2EDM attains SDER gains of 1.13, 1.67, and 1.59 percentage points in Chengdu, Xi'an, and Haikou, respectively, over ASTGCN + Dispatcher; both achieve zero CCVR. The difference may also reflect prediction-architecture choices, so this comparison does not by itself isolate the effect of joint optimization.

The dispatching instances in Tables 7–10 were randomly sampled from the test datasets. Instance numbers identify evaluation scenarios rather than predefined fleet sizes.

To further investigate the impact of architectural design choices on model performance, we conducted a comprehensive ablation study by varying key hyperparameters of the Recurrent Transformer architecture. The results of these experiments are summarized in Table 8.

**Table 8.** Performance analysis with different $M/N$ combinations. Bold values highlight the best SDER results for each instance.

| Configuration | Instance 1 | | Instance 2 | | Instance 3 | | Instance 4 | |
|---|---|---|---|---|---|---|---|---|
| | SDER | CCVR | SDER | CCVR | SDER | CCVR | SDER | CCVR |
| M1N16 | 81.33% | 0% | 81.51% | 3% | 83.13% | 0% | 80.62% | 5% |
| M1N32 | 84.59% | 0% | 83.77% | 0% | 86.41% | 0% | 85.42% | 1% |
| M2N16 | 89.19% | 0% | 90.21% | 1% | 88.79% | 0% | 86.94% | 1% |
| M2N32 | 91.83% | 0% | 92.44% | 0% | **93.13%** | 0% | 91.39% | 0% |
| **M4N16** | **92.61%** | 0% | **93.92%** | 0% | 91.85% | 0% | **92.83%** | 0% |
| M4N32 | 89.37% | 0% | 89.18% | 1% | 89.71% | 0% | 90.38% | 0% |
| M32N1 | 72.21% | 3% | 73.36% | 4% | 74.31% | 2% | 71.76% | 6% |

Increasing recurrent depth generally improves SDER up to a moderate level, but additional depth does not necessarily yield further gains. In particular, M4N32 performs below M4N16 on the four reported instances. Recurrent refinement is useful, whereas excessive depth may introduce redundant reasoning or optimization difficulty.

Figure 6 shows that SDER increases and stabilizes during training, while CCVR decreases toward zero after an intermediate fluctuation. This indicates convergence toward higher supply–demand balance and lower capacity violations under the reported configuration.

To examine positional encoding, we remove it from the M1N16 configuration. For the configuration with positional encoding, Table 8 reports CCVR values of 0%, 3%, 0%, and 5% across the four instances (an arithmetic mean of 2%). Relative percentage changes are not reported because zero-valued baselines make such comparisons undefined. Figure 7 illustrates the effect of removing positional encoding on the allocation pattern.

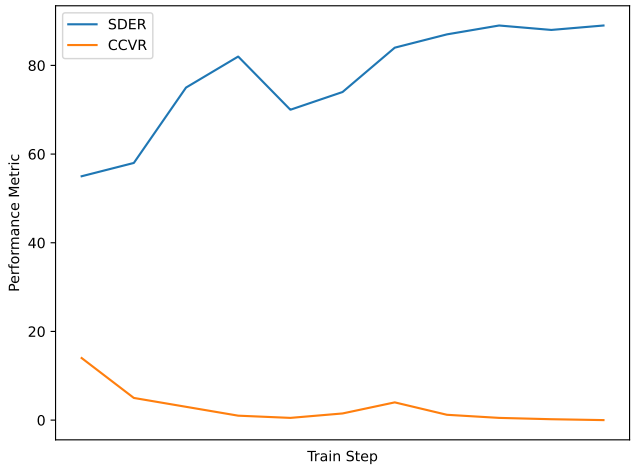



**Figure 6.** Convergence trend of SDER and CCVR over training.

Table 9 compares KM-guided optimization (KMGO) and constraint-guided optimization (CGO). CGO retains zero CCVR without KM-generated labels, although its SDER is generally lower than KMGO. Removing the constraint loss from CGO raises CCVR to 11.24–21.33% and also reduces SDER. This ablation supports the role of the constraint penalties in encouraging capacity-aware decisions under limited supervision.

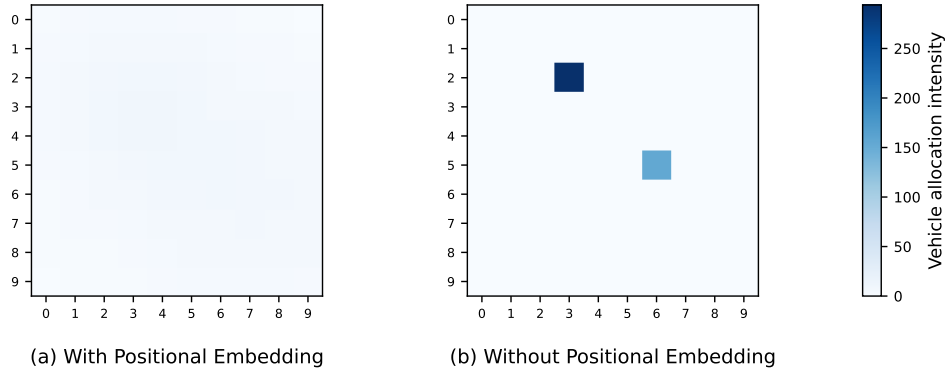



**Figure 7.** Impact of removing positional encoding.

**Table 9.** Loss function ablation experimental results. Bold values highlight the best SDER results for each instance.

| Configuration | Instance 1 | | Instance 2 | | Instance 3 | | Instance 4 | |
|---|---|---|---|---|---|---|---|---|
| | SDER | CCVR | SDER | CCVR | SDER | CCVR | SDER | CCVR |
| KMGO | **92.61%** | 0% | **91.92%** | 0% | **90.85%** | 0% | 91.13% | 0% |
| CGO | 89.23% | 0% | 87.91% | 0% | 89.14% | 0% | **91.44%** | 0% |
| CGO-$L_{\text{con}}$ | 81.33% | 15.72% | 82.86% | 21.33% | 84.15% | 16.51% | 82.49% | 11.24% |

Figure 8 further visualizes the loss-function ablation results across the four evaluation instances, highlighting the SDER differences among the configurations and the capacity violations observed when the constraint loss is removed.

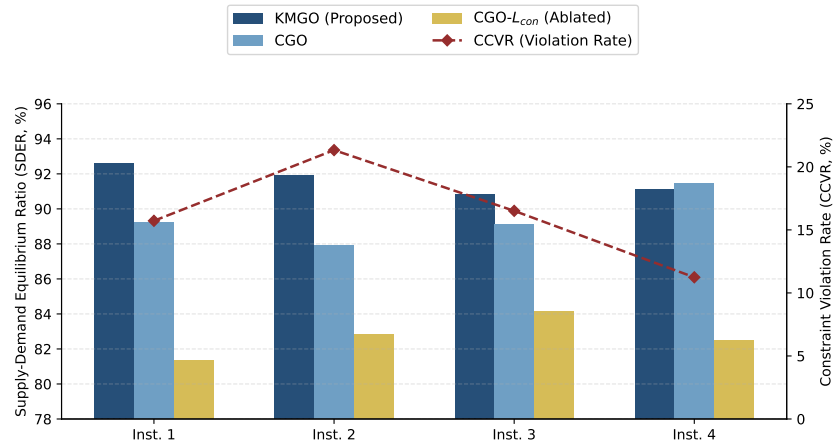



**Figure 8.** Performance comparison of loss function ablation experiment.

Table 7 compares KMGO and CGO with the KM reference in terms of idle distance, a revenue surrogate, and a distance-based waiting-time surrogate. The learned methods do not strictly dominate KM; instead, their values remain close to the reference across the reported instances. Lower idle distance and the waiting-distance surrogate are preferred, whereas higher revenue surrogate is preferred. The comparison supports the interpretation of the neural model as an approximation that trades small objective deviations for substantially lower online computation time. In addition to the instance-level comparison in Table 7, Figure 9 provides a complementary comparison with representative dispatching baselines under peak and off-peak conditions.

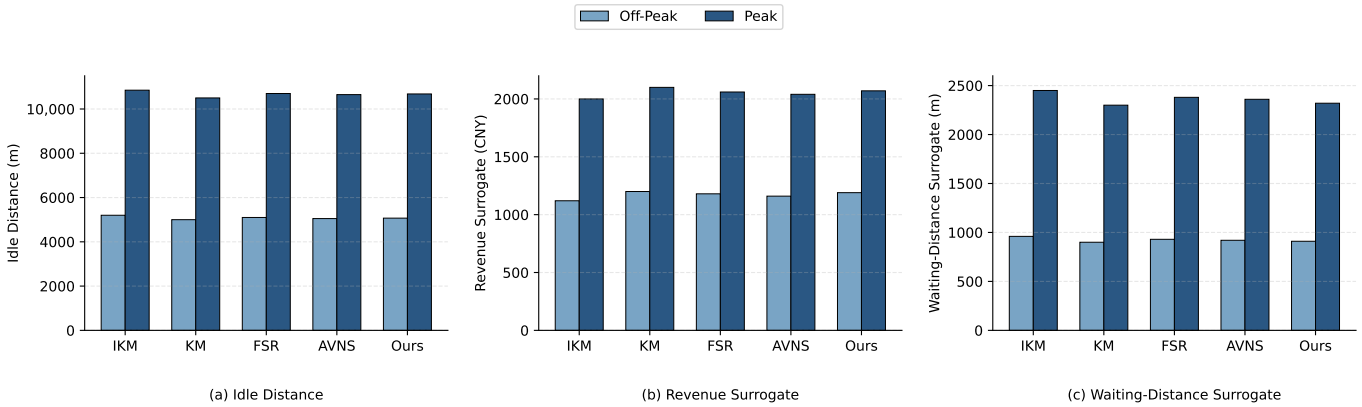



**Figure 9.** Comparison of idle distance (m), revenue surrogate (CNY), and waiting-distance surrogate (m) across dispatching methods under peak and off-peak conditions.

The KM algorithm has polynomial complexity for standard assignment, but its runtime can still be substantial on the reported large instances. The learned model moves most computation to offline training and performs a forward pass during online dispatch. CGO further reduces dependence on KM-generated training labels by using the decision objective and constraint penalties.

For the runtime comparison in Table 10, KM and E2EDM were evaluated on the same 12 dispatching instances, with identical vehicle configurations, regional demand, and capacity constraints in each paired comparison. KM was executed on the Intel Core i7-14600KF CPU, whereas the pretrained E2EDM dispatcher ran on the NVIDIA RTX 4060 Ti GPU. The reported E2EDM time includes recurrent Transformer forward inference and vehicle-wise argmax decoding to produce discrete assignments; it excludes offline model training, demand forecasting, and historical-data preprocessing, and therefore does not represent full prediction–dispatch pipeline latency. Since KM and E2EDM were executed on different processing devices, the measured time ratio reflects their respective implementations and hardware environments rather than a hardware-normalized algorithmic speedup.

As shown in Table 10, KM requires an average of 242.47 s per instance, whereas the proposed model requires 2.45 s. This corresponds to a 98.99% reduction in average online computation time, or approximately a 98.80-fold speedup. The result supports the use of the neural dispatcher as a seconds-level approximation for high-frequency operation; it does not imply millisecond-level latency or exponential KM complexity.

**Table 10.** Online dispatching computation-time comparison for paired instances (E2EDM forward inference and argmax decoding included; offline training and demand prediction excluded).

| Instance | Time (s) | |
|---|---|---|
| | **KM** | **Ours** |
| 1 | 85.53 | 1.81 |
| 2 | 115.32 | 2.85 |
| 3 | 180.41 | 2.72 |
| 4 | 502.51 | 4.33 |
| 5 | 287.09 | 2.21 |
| 6 | 181.16 | 1.95 |
| 7 | 115.44 | 1.98 |
| 8 | 278.10 | 2.32 |
| 9 | 180.70 | 1.14 |
| 10 | 252.44 | 2.36 |
| 11 | 236.81 | 2.62 |
| 12 | 494.13 | 3.16 |
| Average | 242.47 | 2.45 |

### *5.3. Transportation System Implications*

The results suggest four implications for ride-hailing system design. First, prediction models should be assessed not only by MAE or RMSE, but also by how their outputs affect downstream allocation. Second, a trained neural dispatcher can act as a fast online approximation when repeatedly solving a reference optimizer is too slow for the decision frequency. Third, explicit capacity penalties are useful for discouraging regional over-allocation, as indicated by the CCVR ablation. Fourth, the three-city evaluation suggests adaptation potential across heterogeneous demand settings, but transfer to unseen cities still requires dedicated validation. For deployment, the observed MAE–RMSE trade-off suggests monitoring block-level large-error cases, particularly during demand surges, and evaluating lighter graph configurations under the same dispatch constraints before choosing a configuration.

## 6. Conclusions

This study proposed a decision-focused end-to-end trainable framework for scalable vehicle dispatching in ride-hailing systems. The framework integrates multi-graph spatiotemporal demand representation with a recurrent Transformer-based dispatching module and incorporates constraint-aware loss terms to guide capacity-aware vehicle allocation. Experiments on three real-world ride-hailing datasets show competitive dispatching quality and zero CCVR for the proposed model in the reported city-level comparison. The model reduces the reported average online dispatching inference time from 242.47 s for KM to 2.45 s, supporting its role as an efficient approximation for high-frequency operation. The prediction results also show that a dedicated forecasting model can achieve lower standalone error, reinforcing the decision-focused rather than prediction-dominance interpretation.

**Limitations and future work.** The present study uses grid-level distance and historical demand and does not model uncertainty, road-network travel time, battery state of charge, charging-station availability, energy consumption, or heterogeneous fleet constraints. It also evaluates logged multi-city datasets rather than an online simulator, digital twin, or field deployment, and it does not establish transfer performance on unseen cities. Future work should address these limitations through uncertainty-aware prediction, road-network and energy models, richer fleet constraints, and online closed-loop evaluation. The present results are single-run estimates without multi-seed variance reporting, so small differences should be interpreted cautiously. Future work will include random-seed stability analyses, sensitivity tests for the operational objective weights, and comparisons of independent, frozen-predictor, and joint training using identical architectures. We will also evaluate IKM, FSR, and AVNS under controlled runtime protocols and profile time and memory scaling with fleet size, given the quadratic attention-memory growth of full self-attention.

**Author Contributions:** Conceptualization, Y.G. and Y.Q.; methodology, Y.G., Y.Q. and H.W. (Haoran Wang); software, H.W. (Haoran Wang) and Y.Q.; validation, H.W. (Haoran Wang) and H.W. (Hongao Wang); formal analysis, Y.G. and H.W. (Hongao Wang); investigation, H.W. (Haoran Wang) and Y.Q.; resources, Y.Q.; data curation, H.W. (Haoran Wang) and H.W. (Hongao Wang); writing—original draft preparation, Y.G.; writing—review and editing, Y.Q. and H.W. (Haoran Wang); visualization, H.W. (Haoran Wang) and H.W. (Hongao Wang); supervision, Y.Q.; project administration, Y.G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the Natural Science Foundation of Zhejiang Province (grant number LMS25G010003) and the Scientific Research Project of Zhejiang Provincial Department of Education (grant number Y202454572).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are not publicly available because they involve real ride-hailing transaction records and are protected by a data-use agreement with the data provider.

**Acknowledgments:** The authors acknowledge the support described in the Funding section.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Martínez-Díaz, M.; Soriguera, F. Autonomous vehicles: Theoretical and practical challenges. *Transp. Res. Procedia* **2018**, *33*, 275–282. [CrossRef]
2. Olayode, I.O.; Du, B.; Severino, A.; Campisi, T.; Alex, F.J. Systematic literature review on the applications, impacts, and public perceptions of autonomous vehicles in road transportation system. *J. Traffic Transp. Eng. (Engl. Ed.)* **2023**, *10*, 1037–1060. [CrossRef]
3. Guo, Y.; Li, W.; Xiao, L.; Choudhary, A.; Allaoui, H. Enhancing efficiency and interpretability: A multi-objective dispatching strategy for autonomous service vehicles in ride-hailing. *Comput. Ind. Eng.* **2024**, *194*, 110385. [CrossRef]
4. Al-Kanj, L.; Nascimento, J.; Powell, W.B. Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. *Eur. J. Oper. Res.* **2020**, *284*, 1088–1106. [CrossRef]
5. Li, B.; Zhao, M.; Chen, N.; Li, Z.; Sun, H. The Ride-Hailing and Pricing Strategy Decision with the Application of Autonomous Vehicles. *Manag. Decis. Econ.* **2026**, *47*, 286–308. [CrossRef]
6. Yue, X.; Liu, Y.; Shi, F.; Luo, S.; Zhong, C.; Lu, M.; Xu, Z. An End-to-End Reinforcement Learning Based Approach for Micro-View Order-Dispatching in Ride-Hailing. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24, Boise, ID, USA, 21–25 October 2024; pp. 5054–5061. [CrossRef]
7. Guo, Y.; Zhang, Y.; Yu, J.; Shen, X. A Spatiotemporal Thermo Guidance Based Real-Time Online Ride-Hailing Dispatch Framework. *IEEE Access* **2020**, *8*, 115063–115077. [CrossRef]
8. Wen, D.; Li, Y.; Lau, F.C.M. A Survey of Machine Learning-Based Ride-Hailing Planning. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 4734–4753. [CrossRef]
9. Ge, H.; Hu, X.; Cheng, M. DHDRDS: A Deep Reinforcement Learning-Based Ride-Hailing Dispatch System for Integrated Passenger–Parcel Transport. *Sustainability* **2025**, *17*, 4012. [CrossRef]
10. Bertsimas, D.; Kallus, N. From Predictive to Prescriptive Analytics. *Manag. Sci.* **2020**, *66*, 1025–1044. [CrossRef]
11. Yuan, E.; Chen, W.; Van Hentenryck, P. Reinforcement learning from optimization proxy for ride-hailing vehicle relocation. *J. Artif. Intell. Res.* **2022**, *75*, 985–1002. [CrossRef]
12. Bao, Y.; Gao, J.; He, J.; Oliehoek, F.A.; Cats, O. Timing the Match: A Deep Reinforcement Learning Approach for Ride-Hailing and Ride-Pooling Services. *arXiv* **2025**, arXiv:2503.13200.
13. Yan, L.; Tang, R. Dual-Layer Optimization for Supply–Demand Balance in Urban Taxi Systems: Multi-Agent Reinforcement Learning with Dual-Attention Mechanisms. *Electronics* **2025**, *14*, 2562. [CrossRef]
14. Elmachtoub, A.N.; Grigas, P. Smart "Predict, then Optimize". *Manag. Sci.* **2022**, *68*, 9–26. [CrossRef]
15. Sadeghi Eshkevari, S.; Tang, X.; Qin, Z.; Mei, J.; Zhang, C.; Meng, Q.; Xu, J. Reinforcement Learning in the Wild: Scalable RL Dispatching Algorithm Deployed in Ridehailing Marketplace. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Washington, DC, USA , 14–18 August 2022; pp. 3838–3848. [CrossRef]
16. Liu, Y.; Jia, R.; Ye, J.; Qu, X. How machine learning informs ride-hailing services: A survey. *Commun. Transp. Res.* **2022**, *2*, 100075. [CrossRef]
17. Liu, Y.; Wu, F.; Lyu, C.; Li, S.; Ye, J.; Qu, X. Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. *Transp. Res. Part E Logist. Transp. Rev.* **2022**, *161*, 102694. [CrossRef]
18. Feng, J.; Gluzman, M.; Dai, J.G. Scalable Deep Reinforcement Learning for Ride-Hailing. In Proceedings of the 2021 American Control Conference (ACC), New Orleans, LA, USA, 25–28 May 2021; pp. 3743–3748. [CrossRef]
19. Fan, Q.; van Essen, J.T.; Correia, G.H. A bi-level framework for heterogeneous fleet sizing of ride-hailing services considering an approximated mixed equilibrium between automated and non-automated traffic. *Eur. J. Oper. Res.* **2024**, *315*, 879–898. [CrossRef]
20. Megantara, T.R.; Supian, S.; Chaerani, D.; Bon, A.T. The Application of the Piecewise Linear Method for Non-Linear Programming Problems in Ride-Hailing Assignment Based on Service Level, Driver Workload, and Fuel Consumption. *Mathematics* **2024**, *12*, 2290. [CrossRef]
21. Guo, Y.; Zhu, R.; Li, W.; Boulaksil, Y.; Allaoui, H. Dynamic pick-up point recommendation with multi-modal deep forest and incentive-based adaptive Kuhn-Munkres Algorithm. *Knowl.-Based Syst.* **2025**, *330*, 114543. [CrossRef]
22. van Engelen, M.; Cats, O.; Post, H.; Aardal, K. Enhancing flexible transport services with demand-anticipatory insertion heuristics. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *110*, 110–121. [CrossRef]
23. Alonso-Mora, J.; Samaranayake, S.; Wallar, A.; Frazzoli, E.; Rus, D. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. Natl. Acad. Sci.* **2017**, *114*, 462–467. [CrossRef]

24. Sayarshad, H.R.; Chow, J.Y. Non-myopic relocation of idle mobility-on-demand vehicles as a dynamic location-allocation-queueing problem. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *106*, 60–77. [CrossRef]
25. Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; Ye, J. Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, London, UK, 19–23 August 2018; pp. 905–913. [CrossRef]
26. Sartori Vieira, F.; Sörensen, K.; Vanstenwegen, P. Optimization of a semiflexible demand-responsive feeder bus system using variable neighborhood search. *Int. Trans. Oper. Res.* **2025**, *32*, 3413–3440. [CrossRef]
27. Feng, C.; Mei, Z. Optimization of Shared Autonomous Vehicles Routing Problem: From the View of Parking. *Sustainability* **2023**, *15*, 12313. [CrossRef]
28. de Abreu Rodrigues, G.; Junqueira, L.; Vieira, J.G.V.; da Silva Júnior, O.S.; da Cunha, C.B. Rural last-mile distribution using a hybrid heuristic-multi-criteria decision-making framework. *Expert Syst. With Appl.* **2025**, *289*, 128331. [CrossRef]
29. Benjelloun, R.; Tarik, M.; Jebari, K. Open Competency Optimization with Combinatorial Operators for the Dynamic Green Traveling Salesman Problem. *Information* **2025**, *16*, 675. [CrossRef]
30. Engelhardt, R.; Dandl, F.; Bogenberger, K. Speed-up Heuristic for an On-Demand Ride-Pooling Algorithm. *arXiv* **2020**, arXiv:2007.14877.
31. Machado, B.; Pimentel, C.; de Sousa, A. Integration planning of freight deliveries into passenger bus networks: Exact and heuristic algorithms. *Transp. Res. Part A Policy Pract.* **2023**, *171*, 103645. [CrossRef]
32. Kılıç, D.K.; Nielsen, P.; Miehe, F.; Yafrani, M.E. Trustworthiness analysis of heuristic routing algorithm with special constraints for autonomous vehicles in surveillance and coverage operations. *Int. J. Syst. Sci.* **2025**, *56*, 2717–2734. [CrossRef]
33. Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; Lim, A. Learning Improvement Heuristics for Solving Routing Problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 5057–5069. [CrossRef]
34. Thyssens, D.; Dernedde, T.; Sentanoe, W.; Schmidt-Thieme, L. On Distributional Dependent Performance of Classical and Neural Routing Solvers. *arXiv* **2025**, arXiv:2508.02510.
35. Latinopoulos, C.; Zavvos, E.; Kaklis, D.; Leemen, V.; Halatsis, A. Marine Voyage Optimization and Weather Routing with Deep Reinforcement Learning. *J. Mar. Sci. Eng.* **2025**, *13*, 902. [CrossRef]
36. Zhang, J.; Sun, Y.; Feng, Q.; Zhao, Y.; Wang, Z. Data-Driven Robust Optimization of the Vehicle Routing Problem with Uncertain Customers. *Complexity* **2022**, *2022*, 9064669. [CrossRef]
37. Wang, X.; Li, N.; Jin, X. A Reinforcement Learning Hyper-Heuristic with Cumulative Rewards for Dual-Peak Time-Varying Network Optimization in Heterogeneous Multi-Trip Vehicle Routing. *Algorithms* **2025**, *18*, 536. [CrossRef]
38. Zhang, Y.; Long, Y.; Ni, Y.; Huang, L.; Wang, X.; Liu, J. Hierarchical Optimization via LLM-Guided Objective Evolution for Mobility-on-Demand Systems. *arXiv* **2025**, arXiv:2510.10644.
39. Lee, E.H.; Jeong, J. Accessible taxi routing strategy based on travel behavior of people with disabilities incorporating vehicle routing problem and Gaussian mixture model. *Travel Behav. Soc.* **2024**, *34*, 100687. [CrossRef]
40. Zhang, H.; Wang, G.; Wang, X.; Zhou, Z.; Zhang, C.; Dong, Z.; Wang, Y. NondBREM: Nondeterministic Offline Reinforcement Learning for Large-Scale Order Dispatching. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 401–409. [CrossRef]
41. Nejadshamsi, S.; Bentahar, J.; Wang, C.; Eicker, U. Predicting short-Term bike-Sharing demand at station level: A multi-Task dynamic graph-based spatiotemporal approach. *Knowl.-Based Syst.* **2026**, *333*, 114986. [CrossRef]
42. Singh, A.; Al-Abbasi, A.O.; Aggarwal, V. A Distributed Model-Free Algorithm for Multi-Hop Ride-Sharing Using Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 8595–8605. [CrossRef]
43. Guo, X.; Wang, Q.; Zhao, J. Data-Driven Vehicle Rebalancing with Predictive Prescriptions in the Ride-Hailing System. *IEEE Open J. Intell. Transp. Syst.* **2022**, *3*, 251–266. [CrossRef]
44. Guo, Z.; Yu, B.; Shan, W.; Yao, B. Data-driven robust optimization for contextual vehicle rebalancing in on-demand ride services under demand uncertainty. *Transp. Res. Part C Emerg. Technol.* **2023**, *154*, 104244. [CrossRef]
45. Yang, J.; Chen, L.; Su, Z.; Ma, W.; Zou, Z.; An, K. Decision-focused learning for optimal subsidy allocation in ride-hailing services. *Transp. Res. Part C Emerg. Technol.* **2025**, *180*, 105301. [CrossRef]
46. Kang, Y.; Li, T.; Zhao, L.; Gao, Z.; He, J.; Chen, Z.; Pu, B. Dual dynamic graph attention network driven deep reinforcement learning for flexible job-Shop scheduling. *Knowl.-Based Syst.* **2026**, *335*, 115192. [CrossRef]
47. Liang, E.; Wen, K.; Lam, W.H.K.; Sumalee, A.; Zhong, R. An Integrated Reinforcement Learning and Centralized Programming Approach for Online Taxi Dispatching. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 4742–4756. [CrossRef]
48. Lin, K.; Zhao, R.; Xu, Z.; Zhou, J. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, London, UK, 19–23 August 2018; pp. 1774–1783. [CrossRef]
49. Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; Wang, J. Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, 10–15 July 2018*; Dy, J., Krause, A., Eds.; Proceedings of Machine Learning Research (PMLR), 2018; Volume 80, pp. 5571–5580.

50. Wang, D.; Wang, Q.; Yin, Y.; Cheng, T. Optimization of ride-sharing with passenger transfer via deep reinforcement learning. *Transp. Res. Part E Logist. Transp. Rev.* **2023**, *172*, 103080. [CrossRef]
51. Wang, Y.; Yin, H.; Wu, L.; Chen, T.; Liu, C. Secure Your Ride: Real-Time Matching Success Rate Prediction for Passenger-Driver Pairs. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 3059–3071. [CrossRef]
52. Kang, Y.; Chan, J.; Shao, W.; Salim, F.D.; Leckie, C. Long-Term Fairness in Ride-Hailing Platform. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*; Bifet, A., Krilavičius, T., Miliou, I., Nowaczyk, S., Eds.; Springer: Cham, Switzerland, 2024; pp. 223–239.
53. Seo, Y.H.; Kim, D.K.; Kang, S.; Byon, Y.J.; Kho, S.Y. Rebalancing Docked Bicycle Sharing System with Approximate Dynamic Programming and Reinforcement Learning. *J. Adv. Transp.* **2022**, *2022*, 2780711. [CrossRef]
54. Han, X.; Chong, Y.; Huangfu, Q. Shared Bicycle Scheduling Model Based on Price Incentive Mechanism. *Math. Probl. Eng.* **2018**, *2018*, 3713254. [CrossRef]
55. Li, J.; Allan, V.H. Multiple Agents Dispatch via Batch Synchronous Actor Critic in Autonomous Mobility on Demand Systems. In Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART), Rome, Italy, 24–26 February 2024; Volume 2, pp. 198–209.
56. Hiermann, G.; Schiffer, M. A decomposition-based approach for large-scale pickup and delivery problems. *Eur. J. Oper. Res.* **2026**, *333*, 746–761. [CrossRef]
57. Wang, B.; Wang, Y.; He, X.; Hu, Y.; Yin, B. Multi-graph convolutional clustering network. *IET Signal Process.* **2022**, *16*, 650–661. [CrossRef]
58. Zhang, B.; Niu, D.; Zhang, L.; Zhang, Q.; Li, Z. MSH-DTI: Multi-graph convolution with self-supervised embedding and heterogeneous aggregation for drug-target interaction prediction. *BMC Bioinform.* **2024**, *25*, 275. [CrossRef]
59. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Burstein, J., Doran, C., Solorio, T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; Volume 1, pp. 4171–4186. [CrossRef]
60. Gan, Z.; Cheng, Y.; Kholy, A.; Li, L.; Liu, J.; Gao, J. Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*; Korhonen, A., Traum, D., Màrquez, L., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 6463–6474. [CrossRef]
61. Guo, Y.; Li, W.; Xiao, L.; Allaoui, H. A prediction-based iterative Kuhn-Munkres approach for service vehicle reallocation in ride-hailing. *Int. J. Prod. Res.* **2024**, *62*, 3690–3715. [CrossRef]
62. Johnsen, L.C.; Meisel, F. Interrelated trips in the rural dial-a-ride problem with autonomous vehicles. *Eur. J. Oper. Res.* **2022**, *303*, 201–219. [CrossRef]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.