SOUP: Spatial-Temporal Demand Forecasting and Competitive Supply in Transportation

Bolong Zheng, Qi Hu, Lingfeng Ming, Jilin Hu, Lu Chen, Kai Zheng, Christian S. Jensen, *Fellow, IEEE*

Abstract—We consider a setting with an evolving set of requests for transportation from an origin to a destination before a deadline and a set of agents capable of servicing the requests. In this setting, an authority assigns agents to requests such that the average idle time of the agents is minimized. An example is the scheduling of taxis (agents) to meet incoming passenger requests for trips while ensuring that the taxis are empty as little as possible. We address the problem of spatial-temporal demand forecasting and competitive supply (SOUP) in two steps. First, we build a granular model that provides spatial-temporal predictions of requests. Specifically, we propose a Spatial-Temporal Graph Convolutional Sequential Learning (ST-GCSL) model that predicts requests across locations and time slots. Second, we provide means of routing agents to request origins while avoiding competition among the agents. In particular, we develop a demand-aware route planning (DROP) algorithm that considers both the spatial-temporal predictions and the supply-demand state. We report on extensive experiments with real-world data that offer insight into the performance of the solution and show that it is capable of outperforming the state-of-the-art proposals.

Index Terms—Spatial-temporal request forecasting, graph convolutional networks, route planning

1 Introduction

The near-ubiquitous deployment of smartphones has enabled transportation network companies such as Didi Chuxing [1], Uber [3], and Lyft [2] to operate ride-hailing platforms that enable the servicing of transportation requests by means of fleets of drivers. In this setting, drivers accept requests and move to the origins of requests to complete the requests. Such platforms have reduced significantly the amounts of time drivers are idle and the amounts of time spent waiting for service by prospective passengers, thus improving the traffic efficiency of a city. In this setting, historical requests provide insight into the movement patterns of passengers and drivers, which is beneficial for many applications such as traffic prediction, transportation supply scheduling, and route planning.

We study the problem of spatial-temporal demand forecasting and competitive supply (SOUP) in the context of transportation services, which encompasses spatial-temporal forecasting of service requests as well as the planning of routes for agents to active requests in a manner that minimizes the average idle time of all agents. In addition to drivers (agents) looking for passengers (requests), this kind of competitive assignment problem occurs in urban transportation settings, e.g., drivers looking for parking and drivers looking for electric charging stations.

Our focus is on a population of drivers servicing an evolving set of requests for transportation from an origin to a destination within a given time window. The drivers are often called taxis. Most existing proposals on crowdsourced taxis focus either on how to better match taxis with service requests to maximize global revenue [21], [38], [46] or on how to learn taxi and passenger movement patterns from trajectory data to guide route planning [9], [26], [29], [42]. Once a taxi drops off a passenger and completes a request, no further instructions are provided to the taxi to reduce the time it is idle before servicing the next request. Rather, taxis may either stay stationary or may move towards regions with expected high demand, which may lead to competition. We aim to develop a data-driven solution that assigns a route to a taxi as soon as the taxi becomes idle such that the average time taxis are idle is minimized.

Overall, we address two sub-problems:

1) **Dynamic request patterns.** In order to help agents service new requests quickly, we need to know the request patterns across the road network of a city. We first choose carefully a spatial granularity for partitioning a road network and a temporal granularity for partitioning time in order to achieve accurate predictions of requests. We then build a corresponding model that predicts future requests.
2) **Competition among agents.** If all agents tend to move towards hot regions to find new requests, they will compete if the supply-demand ratio is high, which causes the so-called "herding" effect. To eliminate this effect, we develop a route planning strategy that assigns agents to destinations with supply-demand balance.

The framework we propose consists of an offline component and an online component. The former comprises

- *Kai Zheng is the corresponding author.*
- *Bolong Zheng, Qi Hu, Lingfeng Ming are with Huazhong University of Science and Technology.*
  E-mail: {bolongzheng, huqi11, lingfengming}@hust.edu.cn
- *Jilin Hu and Christian S. Jensen are with Aalborg University.*
  E-mail: {hujilin, csj}@cs.aau.dk
- *Lu Chen is with Zhejiang University.*
  E-mail: {luchen}@zju.edu.cn
- *Kai Zheng is with University of Electronic Science and Technology of China.*
  E-mail: {zhengkai}@uestc.edu.cn

Manuscript received April 19, 2005; revised September 17, 2014.

TABLE 1
Summary of Notations

Notation	Definition
$G = (V, E, W)$	A road network
$\mathcal{A} = \{a_i\}$	A set of mobile agents
$\Omega = \{\omega_j\}$	A set of requests
$\mathcal{I}_i = \{I_{ik}\}$	The set of idle times of agent a_i
$R = \{r_i\}$	A set of regions on road network
$T = \{t_i\}$	A set of time slots during a day
r^*	The search route
$\mathcal{G}^{geo} = (R, A^{geo})$	A geographical region correlation graph
$\mathcal{G}^{sem} = (R, A^{sem})$	A semantic region correlation graph
A^{geo}, A^{sem}	The adjacency matrix of $\mathcal{G}^{geo}$ and $\mathcal{G}^{sem}$
C	The number of channels for network input
D^i_j	The number of requests in r_i in time slot t_j
$\mathbf{D}^i$	The request sequence of region r_i
$\mathbf{D}_j$	The request vector at time slot t_j
$\hat{\mathbf{D}}_{t+1}$	The predicted request vector
$\mathbf{\Psi}_t$	The context feature vector at time slot t
Z, Z'	The input and output of an STCM
c_f	The final context features
$\mathbf{\Gamma}^l$	The temporal convolution kernel at l-th layer
$\mathbf{\Theta}^l$	The spectral kernel of GC at l-th layer

an end-to-end deep learning model, called spatial-temporal graph convolutional sequential learning (ST-GCSL), that is capable of predicting requests at different locations and times. The online component comprises a demand-aware route planning (DROP) algorithm that exploits both the available spatial-temporal information on requests and the supply-demand state to guide idle agents.

The major contributions are summarized as follows:

- We propose a spatial-temporal graph convolutional model ST-GCSL that captures spatial, temporal, and short-term spatial-temporal dependencies and accurately predicts future requests.
- We develop DROP to assign routes that take into account both the available spatial-temporal request information and the supply-demand state.
- We report on experiments that suggest that the proposed ST-GCSL and DROP outperform state-of-the-art baseline methods.

The rest of the paper is organized as follows. We detail the problem addressed in Section 2. In Section 3, we present the multi-level partitioning and the ST-GCSL algorithm. Section 4 presents the DROP algorithm. The experimental study is covered in Section 5. The related work is the topic of Section 6. Finally, Section 7 concludes the paper.

2 Preliminaries

We proceed to introduce the background settings and to formalize the SOUP problem. Frequently used notation is summarized in Table 1.

2.1 Settings

The problem setting encompasses four types of entities: a road network, mobile agents (taxis), requests (passengers), and an assignment authority.

Definition 1 (Road Network)**.** *A road network is defined as a weighted directed graph $G = (V, E, W)$, where V is the set of nodes, E is the set of edges, and W is the set of edge weights. Each edge $e(u, v) \in E$ that starts from node u to v has a positive weight $w(u, v) \in W$, i.e., travel time on the edge.*

Definition 2 (Mobile Agent)**.** *We assume a population of mobile agents $\mathcal{A} = \{a_i\}$, which is introduced into the system at once at the beginning. Each a_i has an original location l_i in the road network and is labeled as empty. The population of agents is fixed throughout the operation and has cardinality $|\mathcal{A}|$.*

Definition 3 (Request)**.** *We assume an evolving set of requests $\Omega = \{\omega_j\}$ that are introduced into the system in a streaming fashion. Each request $\omega_j = (o, d, t^o, t^*)$ has an origin o, a destination d, an introduction time t^o, and a maximum life time (MLT) t^*. A request that is not serviced within t^* time units after t^0 is automatically removed from the system, an outcome that we call request expiration.*

When a request enters the system, the agent that meets the following conditions is assigned to the request by the assignment authority:

1) The agent is *empty*.
2) The agent is the agent that is *nearest* to the request.
3) The *shortest-travel-time* from the agent to the request enables the agent to reach the request before it expires.

Once an agent is assigned to a request, the agent is labeled as occupied, and the request is removed from the system. Then the agent moves to the request (for pick-up) and then to the request destination (for drop-off), both along the *shortest-travel-time path* in the road network. When the agent arrives at the destination, it is labeled as empty. If no agent meets the above conditions, the request remains in the system until an agent meets the conditions or the request expires.

Once an agent becomes empty, the assignment authority plans a so-called **search route** r^* for it to serve potential requests. If the agent finishes traversing its search route without having been assigned a new request, it is assigned a new search route.

It is worth noting that an agent neither knows when and where requests will appear nor has any information about other agents. They are not allowed to communicate or collaborate with each other.

Definition 4 (Idle Time)**.** *The idle time of an agent is the amount of time from when the agent is labeled as empty to when it is assigned to a request. An agent may experience multiple idle times in a day, each corresponding to traveling on a search route. We denote $\mathcal{I}_i = \{I_{ik}\}$ as the set of idle times of agent a_i's search routes, where I_{ik} is the idle time of a_i's k-th search route r^*_{ik}.*

In order to reduce the idle time when planning a search route for an idle agent, we build an accurate request data model that the assignment authority can use to make decisions. The data model is a software module that is shared with all agents and that is used to represent request patterns and predict future requests.

2.2 Problem Statement

The SOUP problem consists of two tasks: (1) spatial-temporal demand forecasting; and (2) competitive supply.

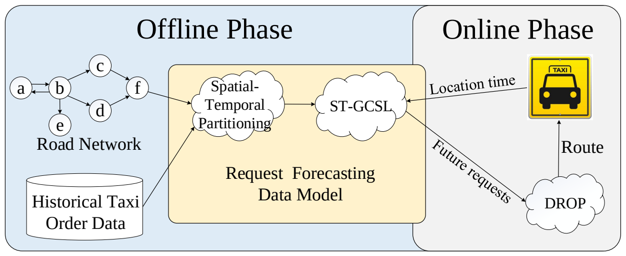


Fig. 1. The SOUP framework includes offline and online stages. In the offline stage, we build a request forecasting data model that predicts future requests. In the online stage, we compute search routes for idle agents based on predicted requests.

Task 1 (Spatial-Temporal Demand Forecasting). *Given a historical set of requests Ω, we aim to build a request data model to predict the numbers of requests at different locations and times.*

Task 2 (Competitive Supply). *Given a request data model, a set of agents $\mathcal{A}$ with original locations, and a stream of requests Ω, we aim to plan search routes for taxis to serve potential requests and avoid the competition such that the average idle time of taxis in Eq. 1 is minimized.*

$$\frac{1}{\sum_{a_i \in \mathcal{A}} |\mathcal{I}_i|} \sum_{a_i \in \mathcal{A}} \sum_{I_{ik} \in \mathcal{I}_i} I_{ik} \tag{1}$$

2.3 Framework Overview

To solve the above problem, we design a search route recommendation framework, as shown in Fig. 1. The processing pipeline includes offline and online components. For the former, we propose the ST-GCSL algorithm to train a request data model based on historical request data, i.e., taxi order data. The data model builds a partitioning of the road network and predicts future requests for the partitions. For the latter component, we propose the DROP algorithm that computes a search route for an agent based on the location and time when it becomes idle and the supply-demand state in the near future.

Once an agent is assigned to a request, it travels to the request for pickup and travels to the request's destination for dropoff. Afterwards, the assignment authority provides the agent with a search route. We limit our discussion to the forecasting and search processes, and we omit the details on the assignment of agents to requests.

3 Spatial Temporal Request Forecasting

We proceed to introduce an end-to-end deep learning model, called spatial-temporal graph convolutional sequential learning (ST-GCSL), to predict where and when requests are likely to appear. First, we introduce a spatial-temporal partitioning, based on which we build a region correlation graph. Then, we present the details of ST-GCSL. Unlike studies [5], [12], [36] that ignore either spatial or short-term spatial-temporal dependencies, ST-GCSL captures all the spatial, temporal, short-term spatial-temporal dependencies, and context features to improve the prediction accuracy.

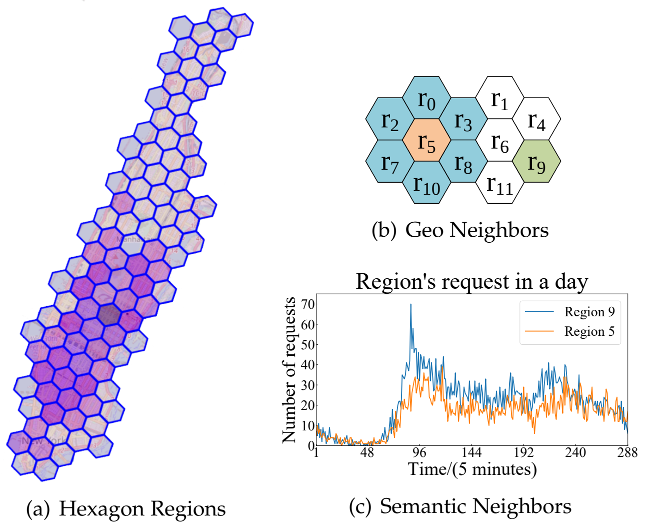


Fig. 2. (a) The hexagon regions of Manhattan. Dark colors indicate high demands. (b) The geographical neighbors of r_5 are r_0, r_2, r_3, r_7, r_8, and r_{10}. (c) Since r_9 and r_5 have similar request patterns, they are semantic neighbors.

3.1 Spatial-Temporal Partitioning

As shown in Fig. 2(a), we partition the road network into N hexagons with Uber H3 library[1] and denote the set of regions by $R = \{r_1, r_2, ..., r_N\}$. We partition a day into M time slots and denote the set of time slots by $T = \{t_1, t_2, \dots, t_M\}$. Let $D_j^i \in \mathbb{R}$ denote the number of requests in region r_i in time slot t_j, we have:

$$D_j^i = |\{\omega | \ \omega.o \in r_i \wedge \omega.t^o \in t_j\}|. \tag{2}$$

From the historical request dataset, we obtain a request sequence $\langle \mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_M \rangle$, where $\mathbf{D}_j = \{D_j^1, D_j^2, \dots, D_j^N\}$ denotes the numbers of requests of all the N regions in time slot t_j, $1 \le j \le M$.

3.2 Region Correlation Graph

Since the spatial dependency between regions can be captured accurately by a topology structure rather than by the Euclidean space [5], we transform the request prediction problem into a graph node prediction problem. Therefore, before we introduce the details of ST-GCSL, we first explain how to build two region correlation graphs based on a historical request dataset.

We follow the existing study [4] to derive two region correlation graphs: (1) a geographical region correlation graph $\mathcal{G}^{geo} = (R, A^{geo})$, and (2) a semantic region correlation graph $\mathcal{G}^{sem} = (R, A^{sem})$, where the set of nodes R is the region set introduced earlier, A^{geo} is the edge set of $\mathcal{G}^{geo}$ in the form of an adjacency matrix, and A^{sem} is the edge set of $\mathcal{G}^{sem}$ in the form of an adjacency matrix.

Specifically, as shown in Fig. 2(b) and Fig. 2(c), to define the adjacency matrices, we consider two types of region neighbors, i.e., **Geographical Neighbor** and **Semantic Neighbor**, based on whether two regions are geographically close or have similar request patterns.

1. https://github.com/uber/h3-java

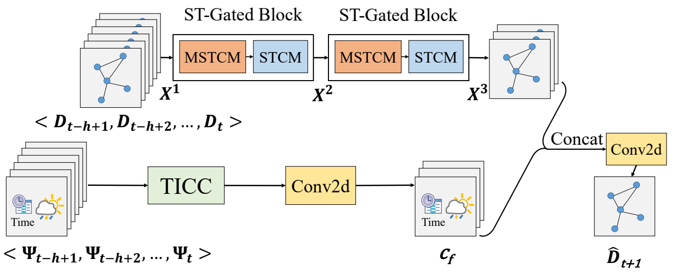


Fig. 3. ST-GCSL consists of two parallel chains that process the historical request sequence and the context feature sequence, respectively.

- The geographical neighbor is based on the first law of geography [33], "near things are more related than distant things," which is used to extract spatial dependencies between a region and its adjacent regions. The geographical adjacency matrix A^{geo} is defined as follows.

$$A^{geo}_{ij} = \begin{cases} 1 & \text{if } r_i, r_j \text{ are geographical neighbors} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

- The semantic neighbor is used to extract semantic correlations among regions with similar request patterns. The intuition is that distant regions may have similar request patterns if they have similar points of interest (POI). We adopt the Pearson Correlation Coefficient [4] to quantify the request pattern similarity between regions. Let $\mathbf{D}^i = \{D^i_1, D^i_2, \dots, D^i_M\}$ represent the request sequence of region r_i in the training data. The semantic similarity between r_i and r_j is defined as follows.

$$Sim(r_i, r_j) = Pearson(\mathbf{D}^i, \mathbf{D}^j) \quad (4)$$

We consider regions r_i and r_j as semantic neighbors if $Sim(r_i, r_j) \geq \epsilon$, where ϵ is a threshold. The semantic adjacency matrix A^{sem} is defined as follows.

$$A^{sem}_{ij} = \begin{cases} 1 & \text{if } r_i, r_j \text{ are semantic neighbors} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

3.3 ST-GCSL Model

The structure of ST-GCSL is shown in Fig. 3. Two parallel chains process the historical request sequence and the context feature sequence, respectively. Consider the current time t.

1) In the upper chain, the input is the historical request sequence of the h most recent time steps, $\langle \mathbf{D}_{t-h+1}, \mathbf{D}_{t-h+2}, \dots, \mathbf{D}_t \rangle$, which is processed by two Spatial-Temporal Gated Blocks (ST-Gated Blocks).
2) In the lower chain, the input is the corresponding context feature sequence, $\langle \mathbf{\Psi}_{t-h+1}, \mathbf{\Psi}_{t-h+2}, \dots, \mathbf{\Psi}_t \rangle$, which is processed by Toeplitz Inverse Covariance-Based Clustering (TICC) [15] and 2D convolution (Conv2d).

The results of these two chains are concatenated and convoluted to return the predicted result at the next time step, i.e., $\hat{\mathbf{D}}_{t+1} \in \mathbb{R}^N$.

3.3.1 *Spatial-Temporal Gated Block (ST-Gated Block)*

The ST-GCSL model has two ST-Gated Blocks, each of which consists of two components: The Multiple Spatial-Temporal Convolutional Module (MSTCM) and The Spatial-Temporal Convolutional Module (STCM).

1) **MSTCM** is stacked by several STCMs (that we call inter-STCMs) and is used to capture short-term spatial-temporal dependencies.
2) **STCM** is used to capture long-term spatial-temporal dependencies. To distinguish it from the STCMs in MSTCM, we call it the outer-STCM.

For the l-th ST-Gated Block, where $l \in \{1, 2\}$, the input is the feature of the request sequence $X^l \in \mathbb{R}^{h_l \times N \times C^l}$, where h_l is the length of time steps, N is the number of nodes (regions), and C^l is the feature dimensionality. Initially, we have $C^l = 1$ when $l = 1$. The output is a new representation of the feature $X^{l+1} \in \mathbb{R}^{[h_l - 2(k-1)] \times N \times C^{l+1}}$, where k is the kernel size of convolution operation. Note that we use two ST-Gated Blocks, so the time step length h decreases to $h - 4(k-1)$.

STCM. STCM is the key module of an ST-Gated Block. As shown in Fig. 4(a), STCM consists of three operations: Conv2d, Graph Convolution (GC), and Gated Linear Unit (GLU) [11]. The input of STCM is $Z \in \mathbb{R}^{q \times N \times C}$. Specifically, for outer-STCM, we have $q = h_l - (k-1)$. For inter-STCMs, we have $q = m$, where m is the short-term time step length. The output is $Z' \in \mathbb{R}^{[q-(k-1)] \times N \times C'}$.

The input is first processed by Conv2d as follows.

$$B = \mathbf{\Gamma} * Z, \quad (6)$$

where $*$ denotes the convolution operator. $\mathbf{\Gamma}$ represents a total of $2 \times C^f$ convolution filters, each of which has kernel size $k \times 1$, stride number 1, and no padding. The output of Conv2d is $B \in \mathbb{R}^{[q-(k-1)] \times N \times 2C^f}$, which is equally split into B_1 and B_2 along the feature dimension, i.e., $B_1, B_2 \in \mathbb{R}^{[q-(k-1)] \times N \times C^f}$. Then, B_1 and B_2 are fed into two GC layers separately.

In order to better extract the spatial dependency, we consider two region correlation graphs. Let us take the geographical region correlation graph $\mathcal{G}^{geo} = (R, A^{geo})$ for example, the graph convolution operation is formulated as follows [23].

$$\mathbf{\Theta}^{geo} \star B_\mu = \tilde{P}^{-\frac{1}{2}} \tilde{A}^{geo} \tilde{P}^{-\frac{1}{2}} B_\mu W_\mu, \quad (7)$$

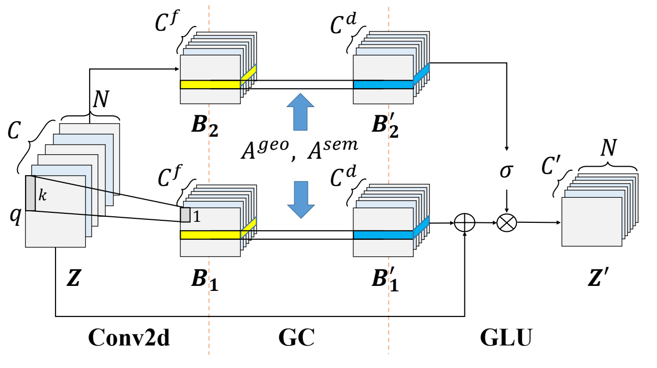
where $\star$ denotes the graph convolution operator and $\mathbf{\Theta}^{geo}$ is a graph convolution filter. $\tilde{A}^{geo} = I + A^{geo} \in \mathbb{R}^{N \times N}$ is the adjacency matrix with self-looping. $\tilde{P} \in \mathbb{R}^{N \times N}$ is the degree matrix with $\tilde{P}_{ii} = \sum_j \tilde{A}^{geo}_{ij}$ and $\tilde{P}_{ij} = 0, \forall i \neq j$. B_μ is the input, W_μ is the weight parameters, where $\mu \in \{1, 2\}$. The graph convolution operation for the semantic region correlation graph is similar. Finally, the output of GC is as follows.

$$B'_\mu = \eta_1 \cdot \mathbf{\Theta}^{geo} \star B_\mu + \eta_2 \cdot \mathbf{\Theta}^{sem} \star B_\mu, \quad (8)$$

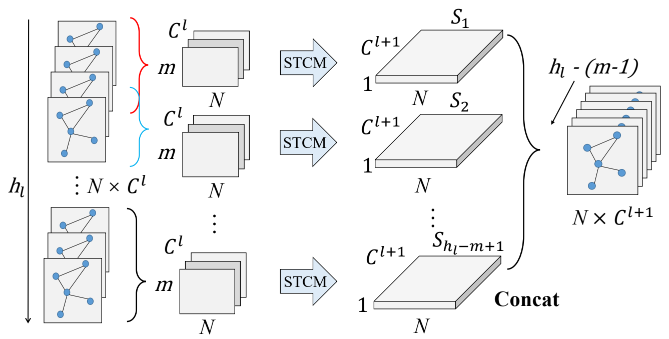
where η_1 and η_2 are trainable parameters.

Furthermore, we use a GLU to model the complex non-linearity in request forecasting, which is formulated as follows.

$$Z' = \left[B'_1 + Z\right] \otimes \sigma \left[B'_2\right], \quad (9)$$



(a) Spatial-Temporal Convolutional Module (STCM)



(b) Multiple Spatial-Temporal Convolutional Module (MSTCM)

Fig. 4. Spatial-Temporal Gated Block

where σ is the Sigmoid function, and $\otimes$ denotes the Hadamard product. On the left half, a residual connection is utilized to avoid network degradation. The right half is a gate that regulates the information flow. Therefore, we have the output $Z^{'} \in \mathbb{R}^{[q-(k-1)]\times N \times C^{'}}$.

Generally, an STCM utilizes Conv2d to extract temporal dependencies and uses GC to capture spatial dependencies. Therefore, an STCM extracts short-term or long-term spatial-temporal dependencies based on the input time step size. Specifically, the inter-STCM is to extract short-term spatial-temporal dependencies with $q = m$, while the outer-STCM is to extract long-term spatial-temporal dependencies with $q = h_l - (k-1)$. Note that m is smaller than $h_l - (k-1)$.

MSTCM. As shown in Fig. 4(b), MSTCM is stacked by multiple inter-STCMs along the time axis to capture the short-term spatial-temporal dependencies. For an MSTCM in the l-th ST-Gated Block, the input $X^l \in \mathbb{R}^{h_l \times N \times C^l}$ is split into m-Gram slices. Each slice $X_i^l \in \mathbb{R}^{m \times N \times C^l}$ is the input to the i-th inter-STCM, where $1 \leq i \leq h_l - m + 1$. The output of MSTCM is $S = [S_1, S_2, \cdots, S_{h_l-m+1}]$, where $[\cdot,\cdot]$ is the concatenation operator, and $S_i \in \mathbb{R}^{[m-(k-1)]\times N \times C^{l+1}}$ is the output of the i-th inter-STCM. According to Eqs. 6, 7, 8, and 9, the operation of each inter-STCM is formulated as follows.

$$S_i = [\mathbf{\Theta}_1^l \star (\mathbf{\Gamma}_1^l * X_i^l) + X_i^l] \otimes \sigma[\mathbf{\Theta}_2^l \star (\mathbf{\Gamma}_2^l * X_i^l)]. \quad (10)$$

For the ease of representation, we use $m = k$ in all MSTCMs, so that $S_i \in \mathbb{R}^{1 \times N \times C^{l+1}}$.

To improve the processing efficiency, these inter-STCMs are executed in parallel such that MSTCM captures the short-term spatial-temporal dependencies of the entire input simultaneously.

3.3.2 Clustering Context Feature Sequence

Due to the strong correlation between the request availability pattern and the extra features, e.g., weekdays, weekends, or rainy days, we introduce them as context features to improve the accuracy of request forecasting. As shown in Fig. 3, we employ clustering on the context features and determine which cluster the context features belong to. Then we add the cluster information as an additional feature.

The context feature $\mathbf{\Psi}_t$ at time step t is a five-tuple (time of day, day of week, weather, holiday, events), which is clustered via TICC. Interested readers may refer to reference [15] for details. Thus, the context features of all h time steps are encoded into a cluster label vector $c \in \mathbb{R}^{h\times 1}$. To concatenate with the output from the upper chain, c is processed by the following two steps on the spatial and temporal dimensions. First, c is duplicated into a tensor $c_d \in \mathbb{R}^{h \times N \times 1}$ to expand the spatial dimension. Then, c_d is processed by a convolution operation with a filter, whose kernel size is $[4(k-1)+1] \times 1$, so that the output is $c_f \in \mathbb{R}^{[h-4(k-1)]\times N \times 1}$.

4 Demand-Aware Route Planning

We proceed to detail the demand-aware route planning (DROP) algorithm that guides idle agents based on the predicted request patterns such that the supply and demand are balanced across regions. Specifically, we first measure the supply-demand state based on the idle rate of agents. Then, we compute a weighted score for each candidate region that measures its degree of popularity. After that, we sample a destination region based on the weighted scores. Finally, we select a destination node in the destination region and provide the shortest-travel-time path to this node.

4.1 Supply-Demand Analysis

To better understand the characteristics of real-time supply-demand states, we visualize the real-time locations of requests and idle agents with the COMSET[2] simulator, as shown in Fig. 5. The New York TLC Trip Record YELLOW Data[3] on June 1, 2016 is used as the simulation data, and we visualize two moments that correspond to peak and non-peak hours.

In Fig. 5(a), we observe that at 12:00 (non-peak hour), the number of requests is small, while most agents locate in the downtown area. Therefore, we need to spread agents across the network as much as possible to eliminate the "herding effect" such that most requests can be served. Fig. 5(b) shows that the number of requests exceeds by far the number of idle agents at 20:00 (peak hour), meaning that the demand by far exceeds the supply, and the request density in the midtown area is significantly higher than those in other

2. https://github.com/JeroenSchols/COMSET-GISCUP
3. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

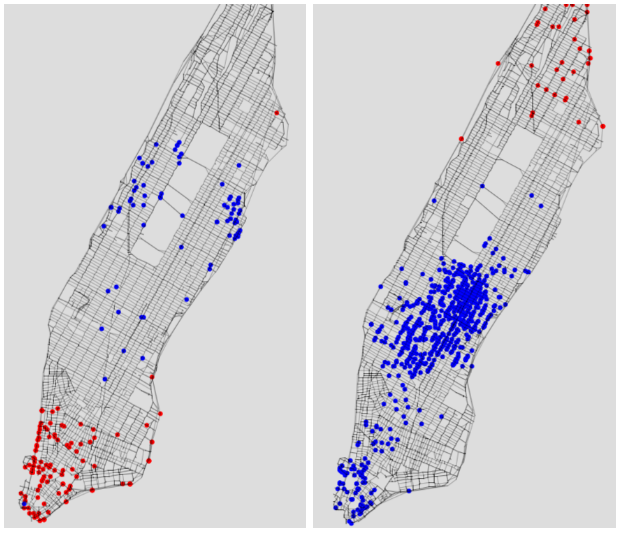

(a) 12:00, non-peak hours (b) 20:00, peak hours

Fig. 5. The real-time locations of requests and idle agents in peak and non-peak hours (red dot: idle agent, blue dot: request).

areas. Therefore, to reduce the agent idle time, we need to guide all agents to the midtown area, rather scatter them across the road network.

To measure different supply-demand states, we use the idle rate of agents, which is the number of idle agents over the agent cardinality $|\mathcal{A}|$. Intuitively, a high idle rate of agents indicates that the supply exceeds the demand, and a low idle rate of agents reveals the otherwise.

4.2 Computing Region Weighted Scores

We utilize the results predicted by ST-GCSL to compute a weighted score for each region. Specifically, we first evaluate the popularity of each region based on its numbers of request origins and destinations, and we then compute region weighted scores for the candidate regions.

Region Popularity Evaluation. Intuitively, the more request origins that locate in a region, the more popular the region is. During peak hours, we only need to send agents to popular regions to serve requests. However, during non-peak hours, we have to consider the competition among agents. If a region is the destination of many requests, newly available agents will soon appear in that region. As a result, part of the requests in this region can be served quickly by these agents [19]. Therefore, the popularity of r_i during t_j is computed as follows.

$$\mathcal{P}_{i,j} = o_{i,j} - ir \times d_{i,j} \tag{11}$$

where $o_{i,j}$ and $d_{i,j}$ represent the predicted number of request origins and destinations in r_i during time slot t_j, respectively, and ir is the idle rate of agents at the time slot t_j. Eq. 11 balances the positive effect of request origins and the negative effect of request destinations by considering the real-time supply-demand state.

Region Weighted Score. Since an agent may spend several time slots moving to the destination, the popularity of the destination region is not consistent in this process. Thus, we use the average popularity of the region in this process as the region weighted score. Furthermore, as agents prefer to serve nearby requests rather than distant requests [22], we borrow a concept called discount factor from Markov Decision Process (MDP) [31] to attenuate the region weighted score. For a region r_i and time slot t_j, the region weighted score $\mathcal{W}_{i,j}$ is computed as follows.

$$\mathcal{W}_{i,j} = \frac{\sum_{k=j}^{j+\Delta-1} \gamma^{k-j} \times \mathcal{P}_{i,j}}{\Delta}, \tag{12}$$

where Δ is the number of estimated time slots it takes an agent to move to the center of r_i, and γ is the discount factor.

In addition, agents can be assigned to serve requests not only in the current region, but also in nearby regions. We conduct a spatial expansion [35] by taking the scores of nearby regions into consideration. With $R_1(r_i)$ representing the first-order neighbors of r_i, a new region weighted score $\mathcal{W}^*_{i,j}$ is computed as follows.

$$\mathcal{W}^*_{i,j} = \alpha \times \mathcal{W}_{i,j} + (1-\alpha) \times \frac{\sum_{r_k \in R_1(r_i)} \mathcal{W}_{k,j}}{|R_1(r_i)|} \tag{13}$$

The first term in Eq. 13 is the previous score of r_i, and the second term is the average score of r_i's first-order neighbors, and parameter α is used to balance their weights.

4.3 Route Planning

The route planning algorithm DROP computes a search route for an agent when the agent becomes idle. It is worth noting that the routing strategy is state-dependent, i.e., the supply-demand state is considered when selecting the destination region.

Candidate Regions Generation. To limit the length of the search route and to reduce the search space, we only consider the L-order neighbors [35] of the agent. For the agent's current region r, let $R_L(r)$ represent a set of regions, where each region is the L-order neighbors of r. Note that $R_0(r) = \{r\}$. We add all regions from $R_0(r)$ to $R_L(r)$ to form the candidate regions $R^* = \cup_{i=0}^{L} R_i(r)$.

Destination Region Determination. In order to dispatch agents according to the request distribution, we first compute a weighted score for each region in R^*. From Fig. 5, we observe that the number of candidate regions depends on the real-time supply-demand state, and the real-time supply-demand state is indicated by the idle rate of agents. Therefore, we keep the top $\beta \cdot ir$ percents of regions with the highest weighted scores in R^*, and remove the remaining regions from R^*, where β is a user-specified parameter that is studied in experiments. Then, we employ a weighted sampling algorithm [19] to sample a destination region from R^*. For a candidate region r_i, the probability of r_i being sampled is:

$$\Pr(r_i) = \frac{\mathcal{W}^*_{i,j}}{\sum_{r_k \in R^*} \mathcal{W}^*_{k,j}} \tag{14}$$

Planning Search Route. To determine the destination, we first select the top 15% nodes with most requests in the destination region as potential destinations. Then, we perform weighted sampling again to find a node as the destination. Finally, a search route to this node along the shortest-travel-time path is assigned to the agent. Note that

TABLE 2
Dataset Statistics

Dataset	# of Orders	# Edges	# Nodes	Size (GB)
New York	69,406,526	9,542	4,360	10.1
Haikou	12,374,094	8,034	3,298	2.28
Chengdu	7,065,938	12,494	3,948	0.83

the destination selection improves the performance to a limited degree. Instead of training the ST-GCSL on the nodes of the road network, we simply use the average number of requests in the historical dataset as the weighted score of each node. Moreover, the 15% of nodes in each region at each time slot can be precomputed to reduce the time cost of route planning.

5 Experimental Study

The ST-GCSL model is implemented in Python3 with Pytorch using a Tesla P100 GPU card, and the DROP algorithm is implemented in Java. The experiments are run on a Windows machine with an Intel 2.8GHz CPU and 12GB memory.

5.1 Experimental Settings

Dataset Description. We use three real datasets, called New York, Haikou[4] and Chengdu[5], to evaluate our method. Table 2 summarizes statistics of the datasets. A request corresponds to a trip record.

- **New York:** The New York TLC Trip Record YELLOW Data includes the records with both pick-up and drop-off information of Yellow Cab taxis in New York City. We extract the data from January through June 2016 for evaluation. The weather data is extracted from New York Central Park[6].
- **Haikou:** The Haikou dataset contains taxi order data from Haikou city for the period from May to October 2017, including the coordinates of origins and destinations, as well as the order type, the travel category, and the number of passengers. The weather data[7] is extracted as the context features.
- **Chengdu:** The Chengdu dataset contains trips in Chengdu during November, 2016. The weather data[8] is extracted as the context features.

Methods for Comparison. For the request forecasting, we compare ST-GCSL with the following methods:

- **HA**: A historical average model that treats the average number of requests as the predicted value.
- **VAR** [16]: A Vector Auto-Regression model that is used to analyze multivariate time series data.
- **LSTM** [17]: A Long Short-Term Memory Network, a typical time series forecasting method.
- **DCRNN** [24]: A Diffusion Convolutional Recurrent Neural Network that models spatial-temporal dependency by integrating graph convolution into gated recurrent units.
- **STGCN** [41]: Spatio-Temporal Graph Convolutional Networks that capture temporal dependency and spatial dependency by using 2D convolutional networks and graph convolutional network, respectively.
- **STG2Seq** [5]: A Spatial-Temporal Graph to Sequence Model that uses multiple gated graph convolution module with two attention mechanisms to capture spatial-temporal dependency.
- **Graph WaveNet** [36]: The Graph WaveNet combines graph convolution with dilated causal convolution to capture spatial-temporal dependency.
- **ASTGCN** [13]: An Attention based Spatial-Temporal Graph Convolution Network that utilizes a spatial-temporal attention mechanism to learn the dynamic spatial-temporal correlations.
- **GMAN** [49]: A graph multi-attention network adapts an encoder-decoder architecture with multiple spatial-temporal attention blocks to model spatial-temporal correlations.
- **AGCRN** [6]: An Adaptive Graph Convolutional Recurrent Network designs a node adaptive parameter learning module and a data adaptive graph generation module to capture spatial-temporal correlations.
- **ASTGNN** [14]: An Attention based Spatial-Temporal Graph Neural Network follows an encoder-decoder structure whose architecture is similar to that of Transformer [34].

To enable a fair comparison, we use the same loss function in all models, defined as follows:

$$Loss(\theta) = ||\mathbf{D}_{t+1} - \hat{\mathbf{D}}_{t+1}||_2^2,$$

where $\mathbf{D}_{t+1}, \hat{\mathbf{D}}_{t+1}$ denote the real request number and the predicted value, respectively. We normalize the context features data to the unit interval using Max-Min normalization and assign them to 10 clusters using the TICC algorithm. The request number is preprocessed using Mean-Std normalization.

For the route planning problem, we compare DROP with four baselines:

- **RD** [37]: Random Destination randomly chooses a node as the destination and then uses the shortest-travel-path between the current location and the destination as the search route.
- **SmartAgent** [22]: SmartAgent uses non-negative matrix factorization (NMF) to model and predict the spatio-temporal distributions of requests, and then chooses destinations using a greedy heuristic.
- **TripBandAgent** [8]: TripBandAgent optimizes the taxi search strategy using reinforcement learning (RL).
- **STP** [19]: Spatial-Temporal Partitioning divides the search space into regions and computes weights for planning routes.

Parameter Settings. For request forecasting, we set the duration of time slot t to 5 minutes for New York, 10 minutes for Haikou, and 10 minutes for Chengdu, respectively. The number of regions N is 99 for New York, 30 for Haikou,

4. https://outreach.didichuxing.com/app-vue/HaiKou?id=999
5. https://outreach.didichuxing.com/competition/kddcup2020
6. http://www.meteomanz.com/index?l=1&cou=4030&ind=72506
7. https://data.cma.cn/
8. https://qq.ip138.com/weather/sichuan/chengdu_lishi.htm

TABLE 3
Parameters (Default value is highlighted)

Parameters	Values
Agent cardinality	**5000**, 5500, 6000, 6500, 7000 (New York) **1000**, 1100, 1200, 1300, 1400, 1500 (Haikou) **2000**, 2500, 3000, 3500, 4000 (Chengdu)
MLT (min)	5, 6, 7, 8, 9, **10**
α	0.6, **0.7**, **0.8**, 0.9, 1.0
β	2, 2.25, **2.5**, 2.75, 3 (New York) 3, **3.25** (Haikou), **3.5** (Chengdu), 3.75, 4
γ	0.6, **0.7**, 0.8, 0.9, 1.0

and 59 for Chengdu, respectively. The batch size of the random gradient descent is 32 and the dropout rate is 0.2. The number of filters in the first and second ST-Gated block are 32 and 64, respectively. We use the Adam optimizer with default learning rate 0.001 for training, and we multiply the learning rate with 0.7 every five epochs. Parameter h is set to 12, while k and m are set to 3. Threshold ϵ is selected from $\{0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0\}$.

For route planning, we summarize the parameter settings of DROP in Table 3. When we vary a parameter, other parameters are set to their default values.

Evaluation Metrics. For the request forecasting problem, we use three well-adopted metrics: Mean Average Percentage Error (MAPE), Mean Absolute Error (MAE), and Rooted Mean Square Error (RMSE). For the route planning problem, we use three evaluation metrics:

1) The average agent idle time (highest priority).
2) The average request waiting time, i.e., the period of time from its introduction to its pick-up or expiration.
3) The percentage of expired requests.

5.2 Request Forecasting Performance Evaluation

We first compare ST-GCSL with the baseline models. Then, we conduct an ablation study on ST-GCSL. Further, we analyze the effect of parameter ϵ. Finally, we compare our model with the baselines in terms of the multi-step prediction accuracy and the training costs. For the data of each month, we use the last 10 days for validation and testing (i.e., 5 days for validation and 5 days for testing) and the remaining days for training.

Comparison With Baseline Models. Table 4 shows the forecasting test errors for the different methods. We have the following observations:

1) The classical methods, including HA and VAR, exhibit poor performance. The reason is that they are incapable of modeling non-linear spatial-temporal dependencies.
2) In general, the deep learning methods perform better. LSTM only takes temporal dependencies into consideration, while GCN-based methods (i.e., DCRNN, STGCN, STG2Seq, Graph WaveNet, ASTGCN, GMAN, AGCRN, and ASTGNN) use two modules to model temporal and spatial dependencies separately. So they perform better than LSTM. Although attention-based methods such as ASTGCN, GMAN, and ASTGNN, have achieved better performance on traffic forecasting problems, they show limited improvements on the request forecasting problem.
3) ST-GCSL achieves the best performance on all datasets, especially in MAE and RMSE. The method takes short-term and long-term spatial-temporal dependencies into account and enables to capture temporal, spatial, and spatial-temporal dependencies, while other methods disregard either spatial or spatial-temporal dependencies.

Ablation Study. To evaluate the effect of the different components of our model, we compare different variants of ST-GCSL, including:

1) Removing the context features.
2) Replacing the GLU with the ReLU activation function.
3) Removing the outer-STCM from ST-Gated blocks.
4) Removing the MSTCM from ST-Gated block.
5) Removing semantic neighbors from the region correlation graph.
6) Removing geographical neighbors from the region correlation graph.

The results are shown in Table 5. We have four observations.

1) Without the context features, the model has poor performance, indicating that the context features contain information useful for prediction.
2) The model with the GLU performs better than the model with the ReLU activation function. This is because the module with GLU has twice as many parameters as ReLU, enabling it capture more complex spatial-temporal dependencies. Also, the gate in GLU is better at controlling the output than is ReLU.
3) Removing the outer-STCM or the MSTCM from ST-Gated Blocks may disregard temporal information and spatial-temporal dependencies to some extent. Specifically, removing the outer-STCM may disregard long-term temporal dependencies while removing the MSTCM may miss short-term spatial-temporal dependencies. Although removing the STCM yields better performance on Chengdu, it is normal to obtain a slight change of results due to data sparsity.
4) Removing one type of neighbors weakens the performance, indicating that it is valuable to consider two types of neighbors.

Overall, the results indicate that our proposed network structure is capable of outperforming the competitors.

Effect of Parameter ϵ. According to the results in Table 5, it is beneficial to consider semantic neighbors. Since parameter ϵ controls the number of semantic neighbors considered, we study the effect of ϵ. Table 6 shows the prediction error with respect to ϵ. We can see that when ϵ is 0.5, 0.8, and 0.7, ST-GCSL achieves the best performance on New York, Haikou, and Chengdu. The reason is that it considers many regions with weak dependency when ϵ is smaller; and when ϵ is larger, it only considers the regions with large similarity, which yields results similar to those obtained when disregarding semantic neighbors. Therefore, ϵ is set to a medium-large value.

Multi-step Prediction Comparison. ST-GCSL is able to conduct multi-step prediction, as can DRCNN, STGCN, STG2Seq, Graph WaveNet, ASTGCN, GMAN, and AGCRN. So we compare ST-GCSL with these methods at request forecasting in 2 and 3 time steps. Table 8 presents the results

TABLE 4
Comparison with Baseline Models

Method	New York			Haikou			Chengdu		
	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE
HA	43.71	3.781	5.993	46.36	4.563	7.047	52.29	8.01	16.048
VAR	30.06	3.525	4.761	36.89	3.464	5.374	41.06	5.344	9.282
LSTM	26.43	3.407	4.723	33.69	3.446	5.328	41.27	5.671	9.396
DCRNN	23.88	3.064	4.718	31.20	3.367	5.314	30.21	4.381	7.052
STGCN	23.66	3.038	4.661	30.01	3.378	5.431	30.07	4.399	7.179
STG2Seq	23.65	3.232	5.176	31.10	3.376	5.308	34.14	5.662	10.333
Graph WaveNet	23.52	2.976	4.605	29.44	3.250	5.145	**29.21**	4.242	6.878
ASTGCN	24.02	3.045	4.667	31.81	3.587	5.408	34.02	4.943	8.133
GMAN	24.52	3.162	5.014	29.56	3.532	5.768	32.99	6.991	14.170
AGCRN	25.11	2.240	3.973	32.45	3.001	4.946	34.67	3.875	6.895
ASTGNN	24.64	3.024	4.595	30.32	3.305	5.274	35.59	4.649	7.677
ST-GCSL	**23.29**	**1.946**	**3.591**	**29.33**	**2.716**	**4.593**	31.23	**3.811**	**6.757**

TABLE 5
Comparison with Variants of ST-GCSL

Removed Component	New York			Haikou			Chengdu		
	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE
Context features	23.42	1.950	3.599	29.75	2.745	4.615	31.61	3.827	6.787
GLU	23.53	1.962	3.595	29.80	2.757	4.675	31.79	3.837	6.806
STCM	23.41	1.950	3.609	30.07	2.763	4.662	**30.78**	**3.725**	**6.501**
MSTCM	23.43	1.961	3.608	30.48	2.818	4.676	31.69	3.842	6.832
Semantic-Neighbor	23.79	1.971	3.625	30.20	2.794	4.673	31.93	3.857	6.844
Geographical-Neighbor	23.73	1.963	3.618	29.90	2.768	4.630	32.28	3.869	6.829
ST-GCSL	**23.29**	**1.946**	**3.591**	**29.33**	**2.716**	**4.593**	31.23	3.811	6.757

TABLE 6
MAPE When Varying ϵ

Dataset	RMSE						
	0.2	0.4	0.5	0.6	0.7	0.8	1.0
New York	3.606	3.592	**3.591**	3.611	3.603	3.604	3.625
Haikou	4.645	4.611	4.648	4.629	4.618	**4.593**	4.673
Chengdu	6.761	6.778	6.764	6.802	**6.757**	6.805	6.844

TABLE 7
Training Time Comparison

Method	New York		Haikou		Chengdu	
	Training (s/epoch)	Inference (s)	Training (s/epoch)	Inference (s)	Training (s/epoch)	Inference (s)
DCRNN	23.08	4.33	10.65	3.18	9.36	2.89
STGCN	2.56	3.39	0.94	0.80	0.76	1.06
STG2Seq	18.96	1.65	5.81	0.62	8.06	0.94
Graph WaveNet	11.65	0.96	3.81	0.43	4.14	0.41
ASTGCN	9.43	1.05	2.55	0.22	3.21	0.37
GMAN	22.90	1.88	4.50	0.35	7.35	0.61
AGCRN	15.08	1.53	4.80	0.43	4.75	0.46
ASTGNN	54.05	24.44	9.12	3.88	13.31	4.91
ST-GCSL	13.24	1.36	4.25	0.42	5.21	0.45

for all metrics on all datasets. As we can see, ST-GCSL exhibits the best performance.

Training Time Analysis. We compare ST-GCSL with the baseline models with respect to the training time of the multi-step prediction. For fairness, we measure the epoch time cost of each method on training data with same batch size. For inference, we record the total time cost of each method on the validation data. The results in Table 7 indicate the following.

1) STGCN is the fastest by applying the convolution along time axis and using minimal GC operations. ASTGNN is the slowest due to a large number of GC operations in the encoder-decoder structure. DCRNN is also slow due to its integration of GC into the gated recurrent unit, which means that the GC operation is executed at each time step.
2) In general, the training time of all methods grows linearly with the number of data-points and the CNN-based prediction methods including STG2Seq, Graph WaveNet, ASTGCN, GMAN, AGCRN, and ST-GCSL, achieve a better training and inference performance.
3) ST-GCSL achieves satisfactory training and inference performance. Although the parallel operation of MSTCM in ST-GCSL is similar with that of STG2Seq, the design of outer-STCM decreases the number of GC operations, which makes ST-GCSL more efficient.

The experimental findings thus indicate that ST-GCSL not only achieves the best performance, but also can be trained efficiently.

5.3 Route Planning Performance Evaluation

For New York and Haikou datasets, we randomly select two days data from each month to form the testing dataset. For Chengdu dataset, since it only contains one month's trip data, we take the data of the first three weeks for training, and the data of the last week for testing. To better show the performance improvement, we report the average improvement compared to RD in Figs. 6 to 11 on previously mentioned three metrics. The experimental results of different parameter settings are shown in Table 9. As the accuracy improvement of ST-GCSL over other prediction models is not significant, DROP using other prediction models may have similar performance with using ST-GCSL, so we do not replace ST-GCSL with other models for ablation study.

Varying the Agent Cardinality. Figs. 6, 7, and 8 show the performance of all methods when varying the agent cardinality. We make three observations.

TABLE 8
Multi-step Prediction

Method	New York (step 2/3)			Haikou (step 2/3)			Chengdu (step 2/3)		
	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE
DCRNN	24.20/24.54	3.121/3.187	4.871/5.023	31.54/32.66	3.465/3.610	5.546/5.762	30.15/30.37	4.446/4.520	7.259/7.450
STGCN	24.21/24.69	3.114/3.185	4.841/4.977	30.88/32.08	3.508/3.688	5.668/5.966	30.55/30.97	4.518/4.638	7.441/7.697
STG2Seq	24.61/25.75	3.581/3.961	6.012/6.873	33.19/35.13	3.578/3.781	5.618/5.937	35.62/36.81	6.516/7.254	12.422/14.293
Graph WaveNet	23.72/24.08	3.043/3.098	4.746/4.866	30.04/30.24	3.317/3.341	5.28/5.324	**29.34**/**29.65**	4.275/4.322	6.937/7.704
ASTGCN	24.49/24.83	3.122/3.174	4.796/4.925	32.37/32.81	3.674/3.756	5.587/5.751	34.95/35.29	5.070/5.146	8.348/8.507
GMAN	24.61/24.68	3.176/3.184	5.041/5.061	29.73/**29.91**	3.542/3.558	5.804/5.820	33.19/33.22	7.033/7.123	14.229/14.269
AGCRN	25.41/25.63	2.288/2.341	4.052/4.134	30.20/30.77	3.019/3.077	4.982/5.081	34.68/34.72	3.913/3.968	6.985/7.093
ASTGNN	24.95/25.14	3.089/3.3135	4.756/4.869	31.20/31.77	3.411/3.531	5.490/5.715	36.45/36.49	4.665/4.647	7.708/7.718
ST-GCSL	**23.68**/**24.03**	**2.004**/**2.039**	**3.691**/**3.766**	**29.59**/30.08	**2.767**/**2.846**	**4.723**/**4.841**	31.58/32.36	**3.842**/**3.927**	**6.863**/**7.002**

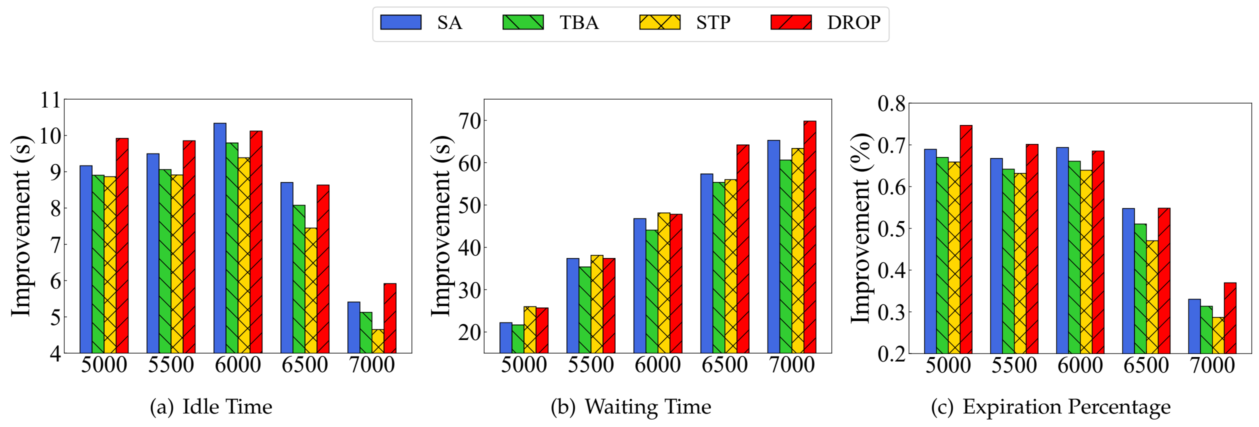


Fig. 6. Comparison with Baselines on New York Varying the Agent Cardinality

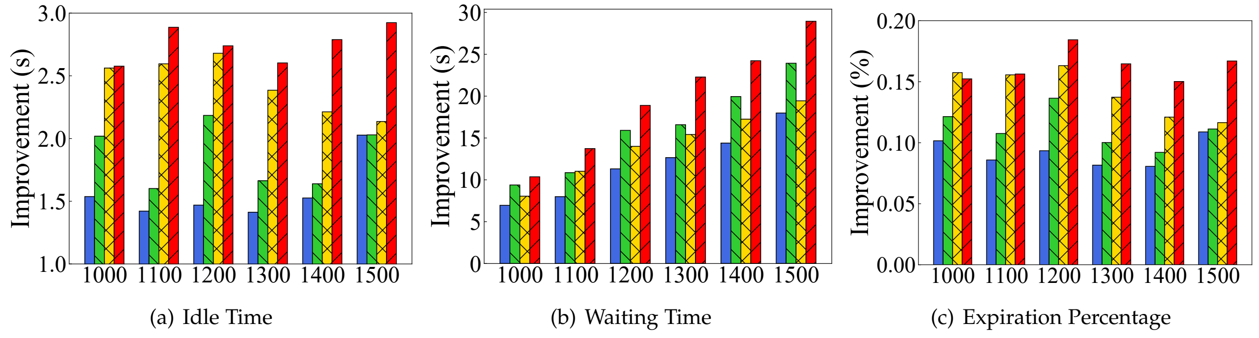


Fig. 7. Comparison with Baselines on Haikou Varying the Agent Cardinality

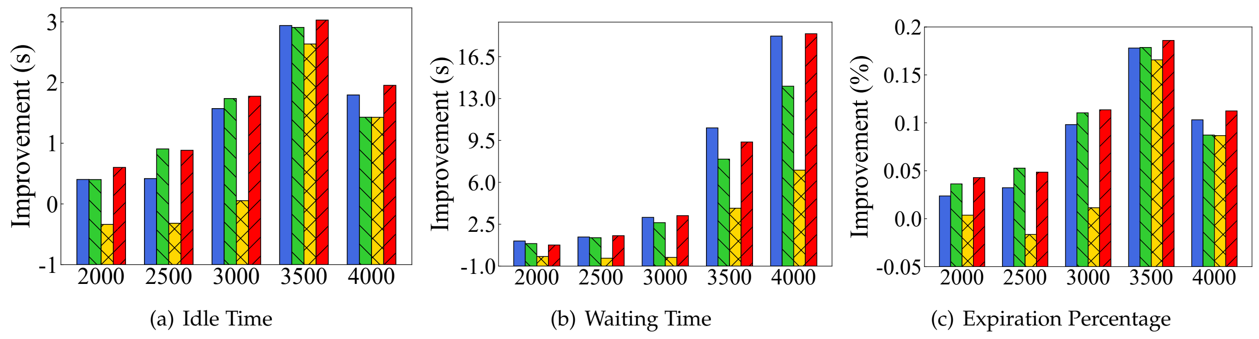


Fig. 8. Comparison with Baselines on Chengdu Varying the Agent Cardinality

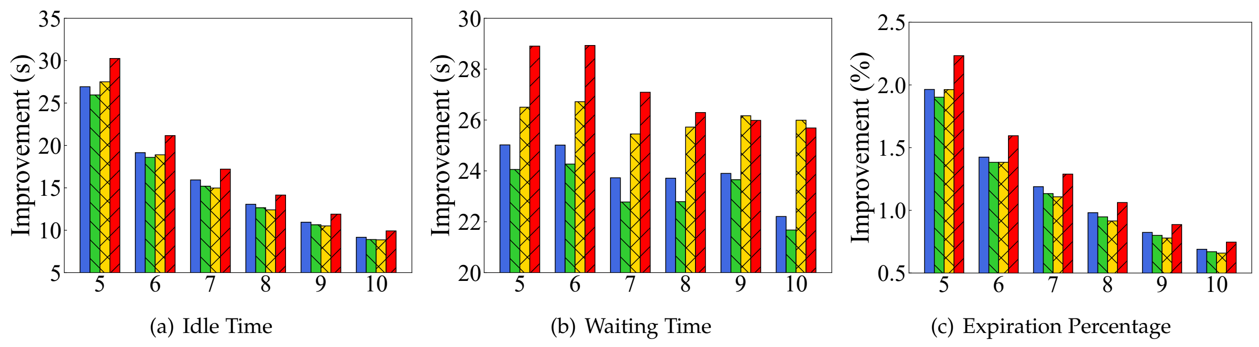


Fig. 9. Comparison with Baselines on New York Varying the MLT

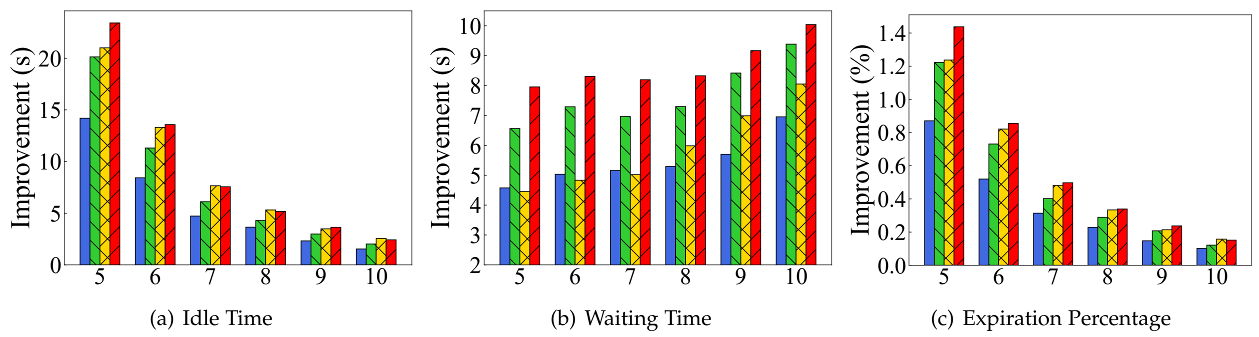


Fig. 10. Comparison with Baselines on Haikou Varying the MLT

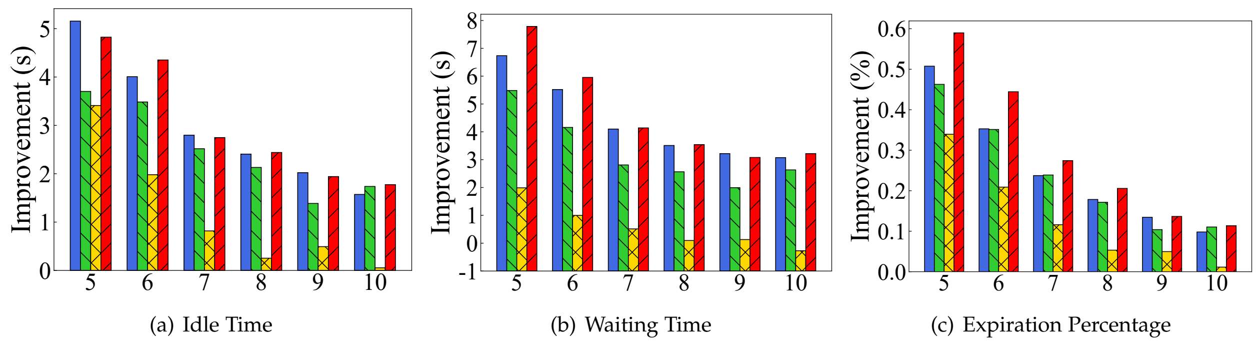


Fig. 11. Comparison with Baselines on Chengdu Varying the MLT

1) The idle time improvement of DROP is the highest. For instance, when the agent cardinality is 5000 on New York, the average idle time of RD is 368.8 seconds, and the improvement of the idle time for SA, TBA, and STP are 9.2, 8.9, and 8.8 seconds, while the improvement achieved by DROP is 9.9 seconds. The reason is that the predicted results of ST-GCSL are close to the real request distribution, and DROP guides agents to popular regions.
2) DROP outperforms the baseline methods in terms of request waiting time. With the increase of agent cardinality, the improvement of DROP increases compared to those of the other methods. This is because DROP can guide more agents to locations near requests.
3) DROP has the highest expiration percentage improvement compared to the baseline methods in all cases, since it distributes agents according to the distribution of requests.

Varying the MLT. Figs. 9, 10, and 11 show the performance of all methods when varying the MLT. We make the following observations.

1) DROP outperforms the other methods with respect to idle time, especially when MLT is small. For example, when MLT is 5 minutes on New York, the average idle time of RD is 400.1 seconds, and the improvement of DROP is 30.2 seconds, while those of SA, TBA, and STP are 26.9, 26, and 27.5 seconds, respectively. This is because when MLT is small, the requests can only be assigned to nearby agents, and the routing strategy of DROP can direct agents to locations closer to requests than can the other methods.
2) DROP has the best performances in terms of waiting time and expiration percentage on the three datasets.

Parameter Study. We compare the performance of DROP under different parameter settings shown in Table 9. We observe that the performance changes only little across these settings. Note that as the three datasets are of different sizes, there are differences in the parameter settings. Specifically, the parameter β that regulates the supply-demand state and the number of candidate regions, varies from 2 to 3 in New York, while it varies from 3 to 4 in Haikou and Chengdu.

6 Related Work

6.1 Taxi Demand Prediction

Traffic demand prediction is a critical aspect when aiming to achieve an efficient transportation system. Many studies employ convolutional neural networks (CNNs). For example, ConvLSTM [30] combines CNNs and recurrent neural networks (RNNs) to model spatial and temporal dependencies, respectively, which is an extension of fully-connected LSTMs [17]. ST-ResNet [45] models the temporal closeness, period, and trend properties of crowd traffic based on existing studies [43], [44]. DMVST-Net [40] employs graph embedding as an external feature to improve forecast accuracy based on localized spatial and temporal views.

However, these CNN-based methods only model the Euclidean relations among grid regions and ignore non-Euclidean relations. In contrast, graph convolutional networks (GCNs) can extract local features from non-Euclidean structures, resulting in improved performance. For instance, One study [18] predicts the travel cost, while GCWC [10] fills in missing stochastic weights of speed via GCN, and their techniques can be applied to demand prediction. DCRNN [24] models the spatial-temporal dependency by integrating graph convolution into the gated recurrent units. The architecture is like that of ConvLSTM, but replaces the Conv2d with GC. STGCN [41] captures temporal dependency and spatial dependency by using 2D convolutional networks and graph convolutional networks, respectively. STG2Seq [5] uses multiple gated graph convolution modules with two attention mechanisms to capture spatial-temporal dependency, while Graph WaveNet [36] learns a self-adaptive adjacency matrix and combines graph convolution with dilated casual convolution to capture spatial-temporal dependencies. STMGCN [47] employs a multi-graph to extract spatial dependency and uses RNNs to capture temporal dependency. ASTGCN [13], GMAN [49], and ASTGNN [14] are attention based spatial-temporal graph convolution networks that utilize a spatial-temporal attention mechanism to learn the dynamic spatial-temporal correlations, while AGCRN [6] designs a node adaptive parameter learning module and a data adaptive graph generation module to capture the dynamic spatial-temporal correlations.

The GCN-based methods are more flexible and progressive than the CNN-based methods, but there are also differences between them. STGCN, Graph WaveNet, and STMGCN use separate components to capture temporal and spatial dependency, while the GC operation in DCRNN and STG2Seq is executed by short-term time steps, meaning that they capture spatial and temporal dependency simultaneously, which also leads to much more computation. Further, based on the three temporal and spatial dependencies in the traffic forecasting problem we study, the above methods all miss spatial-temporal dependency to some extent. Unlike those methods, our method captures all three types of dependencies and can be trained efficiently.

6.2 Taxi Dispatching

Existing studies on taxi dispatching generally concern either order matching or route recommendation. Order matching aims to match idle taxis with appropriate passengers, often to maximize global revenue. One study [38] considers both instant passenger satisfaction and expected future gain in a unified decision-making framework, and then optimizes long-term platform efficiency through reinforcement learning. Another study [32] models the ride dispatching problem as a Semi Markov Decision Process and proposes Cerebellar Value Networks (CVNet) with a novel distributed state representation layer to learn the best dispatch policy. Moreover, some studies [7], [10] introduce the idea of game theory into the matching problem and model the matching of taxis and passengers as a process to reach a stable Nash equilibrium.

Considering route recommendation, existing proposals can be divided into combination optimization methods and deep reinforcement learning (DRL) methods. One study [39] models the driver repositioning task as a classical Minimum Cost Flow (MCF) problem and then solves it by combination optimization. MCF-FM [27] develops a continuous order dispatch strategy for effective fleet management. Next, DRL is used widely to deal with taxi dispatching and repositioning problems. For instance, MOVI [28] uses CNNs to extract supply and demand distribution features and adopts a distributed DQN policy to learn the best dispatch action for each vehicle. One study [25] considers geographic context and collaborative context to remove invalid actions, proposing a contextual actor-critic multi-agent reinforcement learning algorithm to adapt to dynamic traffic. Another study [48] uses a deep Q-network with action sampling policy (AS-DQN) to learn an optimal dispatching policy. However, these DRL-based methods require that all taxis to be dispatched at a same time slot to achieve coordination among taxis, which is impossible in practise. Moreover, these methods ignore the real-time supply and demand states. In contrast, DROP guides idle taxis based on the real-time supply and demand distributions. A demo of SOUP is described elsewhere [20].

7 Conclusion

We study the problem of spatial-temporal demand forecasting and competitive supply (SOUP) in transportation. We propose the ST-GCSL model for request prediction and develop the DROP algorithm to guide idle agents. We report on experimental studies that show that ST-GCSL is capable of outperforming the baseline models considered. Specifically, DROP outperforms all the competitors and reduces the idle time, waiting time, and expiration percentage by 9.9s, 25.7s, and 0.8% compared with the RD method on the New York dataset.

Acknowledgments

This research is supported in part by the NSFC (Grants No. 61902134, 62011530437, 61972069, 61836007,

TABLE 9
Performance of Different Parameter Settings

Parameters		New York			Haikou			Chengdu		
		IT (s)	WT (s)	Exp (%)	IT (s)	WT (s)	Exp (%)	IT (s)	WT (s)	Exp (%)
α	0.6	359.021	253.156	7.947	436.396	**295.493**	8.305	476.682	345.712	27.494
	0.7	**358.827**	**252.906**	**7.932**	**436.174**	295.734	**8.288**	476.682	**345.659**	27.498
	0.8	358.960	253.391	7.937	436.461	295.874	8.319	476.736	345.866	27.496
	0.9	359.064	253.529	7.950	436.547	296.373	8.321	476.581	345.669	27.503
	1.0	358.993	253.905	7.945	436.798	296.373	8.344	**476.627**	345.719	**27.484**
β	2/3	359.161	253.115	7.954	436.575	296.038	8.320	477.018	345.983	27.518
	2.25/3.25	358.957	252.231	7.934	436.547	295.897	8.315	476.912	345.853	27.510
	2.5/3.5	**358.848**	**251.800**	**7.930**	436.577	295.650	8.305	476.754	345.900	**27.498**
	2.75/3.75	358.977	251.806	7.945	436.297	295.558	**8.297**	**476.749**	345.942	27.506
	3/4	359.021	253.572	7.947	**436.167**	**295.451**	8.302	476.949	**345.894**	27.521
γ	0.6	359.059	253.738	7.945	436.429	**295.441**	8.315	476.781	345.765	27.500
	0.7	358.810	**251.811**	7.928	436.563	295.445	8.309	**476.714**	**345.751**	27.503
	0.8	358.828	252.907	7.933	**436.352**	295.866	**8.303**	476.804	345.854	27.512
	0.9	**358.759**	252.946	**7.926**	436.724	296.121	8.328	476.719	345.835	**27.494**
	1.0	358.814	252.964	7.934	436.567	295.900	8.319	476.893	346.217	27.522

and 61832017), the Hubei Natural Science Foundation (Grant No. 2020CFB871), and the Fundamental Research Funds for the Central Universities (HUST: Grants No. 2019kfyXKJC021, 2019kfyXJJS091).

REFERENCES

[1] Didi: https://www.xiaojukeji.com, 2020.

[2] Lyft: https://www.lyft.com, 2020.

[3] Uber: https://www.uber.com, 2020.

[4] L. Bai, L. Yao, S. S. Kanhere, X. Wang, W. Liu, and Z. Yang. Spatio-temporal graph convolutional and recurrent networks for citywide passenger demand prediction. In *CIKM*, pages 2293–2296, 2019.

[5] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng. Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In *IJCAI*, pages 1981–1987, 2019.

[6] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *NeurIPS*, 2020.

[7] R. Bai, J. Li, J. A. D. Atkin, and G. Kendall. A novel approach to independent taxi scheduling problem based on stable matching. *JORS*, 65(10):1501–1510, 2014.

[8] F. Borutta, S. Schmoll, and S. Friedl. Optimizing the spatio-temporal resource search problem with reinforcement learning (GIS cup). In *SIGSPATIAL/GIS*, pages 628–631, 2019.

[9] A. Braverman, J. G. Dai, X. Liu, and L. Ying. Empty-car routing in ridesharing systems. *Operations Research*, 67(5):1437–1452, 2019.

[10] P. Cheng, L. Chen, and J. Ye. Cooperation-aware task assignment in spatial crowdsourcing. In *ICDE*, pages 1442–1453, 2019.

[11] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941, 2017.

[12] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *AAAI*, pages 3656–3663, 2019.

[13] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, pages 922–929. AAAI Press, 2019.

[14] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[15] D. Hallac, S. Vare, S. P. Boyd, and J. Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *IJCAI*, pages 5254–5258. ijcai.org, 2018.

[16] J. D. Hamilton. *Time series analysis*, volume 2. Princeton New Jersey, 1994.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] J. Hu, C. Guo, B. Yang, C. S. Jensen, and H. Xiong. Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks. In *ICDE*, 2020.

[19] Q. Hu, L. Ming, C. Tong, and B. Zheng. An effective partitioning approach for competitive spatial-temporal searching (GIS cup). In *SIGSPATIAL/GIS*, pages 620–623, 2019.

[20] Q. Hu, L. Ming, R. Xi, L. Chen, C. S. Jensen, and B. Zheng. SOUP: A fleet management system for passenger demand prediction and competitive taxi supply. In *ICDE*, pages 2657–2660, 2021.

[21] J. Jin, M. Zhou, W. Zhang, M. Li, Z. Guo, Z. Qin, Y. Jiao, X. Tang, C. Wang, J. Wang, G. Wu, and J. Ye. Coride: Joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *CIKM*, pages 1983–1992, 2019.

[22] J. Kim, D. Pfoser, and A. Züfle. Distance-aware competitive spatiotemporal searching using spatiotemporal resource matrix factorization (GIS cup). In *SIGSPATIAL/GIS*, pages 624–627, 2019.

[23] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[24] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR (Poster)*. OpenReview.net, 2018.

[25] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *KDD*, pages 1774–1783, 2018.

[26] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *IEEE Trans. Automation Science and Engineering*, 13(2):463–478, 2016.

[27] L. Ming, Q. Hu, M. Dong, and B. Zheng. An effective fleet management strategy for collaborative spatio-temporal searching: GIS cup. In *SIGSPATIAL/GIS*, pages 651–654, 2020.

[28] T. Oda and C. Joe-Wong. MOVI: A model-free approach to dynamic fleet management. In *INFOCOM*, pages 2708–2716, 2018.

[29] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong. A cost-effective recommender system for taxi drivers. In *KDD*, pages 45–54, 2014.

[30] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*, pages 802–810, 2015.

[31] R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction*. MIT Press, 1998.

[32] X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A deep value-network based approach for multi-driver order dispatching. In *KDD*, pages 1780–1790, 2019.

[33] W. R. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46(sup1):234–240, 1970.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

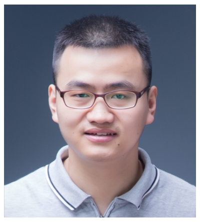
[35] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu. Deep reinforcement learning with knowledge transfer for online rides order dispatching. In *ICDM*, pages 617–626, 2018.

[36] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, pages 1907–1913. ijcai.org, 2019.
[37] B. Xu, Y. Pao, and U. Khurana. Sigspatial gis cup, 2019.
[38] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *KDD*, pages 905–913, 2018.
[39] Z. Xu, C. Men, P. Li, B. Jin, G. Li, Y. Yang, C. Liu, B. Wang, and X. Qie. When recommender systems meet fleet management: Practical study in online driver repositioning system. In *WWW*, pages 2220–2229, 2020.
[40] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI*, pages 2588–2595, 2018.
[41] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, pages 3634–3640. ijcai.org, 2018.
[42] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *TKDE*, 25(10):2390–2403, 2013.
[43] J. Zhang, Y. Zheng, and D. Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017.
[44] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi. Dnn-based prediction model for spatio-temporal data. In *SIGSPATIAL/GIS*, pages 92:1–92:4, 2016.
[45] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artif. Intell.*, 259:147–166, 2018.
[46] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye. A taxi order dispatch model based on combinatorial optimization. In *KDD*, pages 2151–2159, 2017.
[47] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng. Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *AAAI*, pages 2245–2252, 2019.
[48] B. Zheng, L. Ming, Q. Hu, Z. Lü, G. Liu, and X. Zhou. Supply-demand aware deep reinforcement learning for dynamic fleet management. *ACM TIST.*, 2021.
[49] C. Zheng, X. Fan, C. Wang, and J. Qi. GMAN: A graph multi-attention network for traffic prediction. In *AAAI*, pages 1234–1241. AAAI Press, 2020.

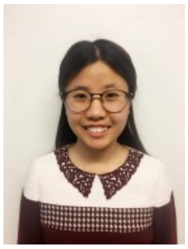
Bolong Zheng is an associate professor in Huazhong University of Science and Technology (HUST). He received his PhD from University of Queensland in 2017. He received bachelor and master degrees in computer science from HUST in 2011 and 2013, respectively. His research interests include spatial-temporal data management.He has published over 40 papers in prestigious journals and conferences in data management field such as SIGMOD, ICDE, VLDB, ACM Transactions and IEEE Transactions.

Qi Hu is pursuing his master degree in computer science at Huazhong University of Science and Technology (HUST), under the supervision of Dr. Bolong Zheng. He received his bachelor degree in Geographic Information Science (GIS) from Wuhan University in 2019. His research interests include spatio-temporal data management, fleet management, and reinforcement learning.

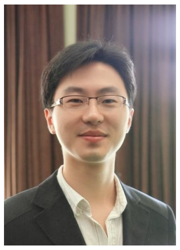
Lingfeng Ming is pursuing his master degree in computer science at Huazhong University of Science and Technology (HUST), under the supervision of Dr. Bolong Zheng. He received his bachelor degree in Energy and Power Engineering from HUST in 2019. His research interests include spatio-temporal data management, fleet management, and reinforcement learning.



Jilin Hu received the PhD degree from Aalborg University, Denmark, in 2019. He is currently an assistant professor at Aalborg University, Denamrk. His research interests include spatio-temporal data management, traffic data analytics, and machine learning. He was a session chair for PVLDB'20. He has been reviewers for several top tier journals, e.g., IEEE TKDE, VLDB Journal, IEEE TNNLS, Neurocomputing, etc. He was also PC members for CVPR'21, AAAI'21, APWeb'20.



Lu Chen received the PhD degree in computer science from Zhejiang University, China, in 2016. She was an assistant professor in Aalborg University for a 2-year period from 2017 to 2019, and she was an associated professor in Aalborg University for a 1-year period from 2019 to 2020. She is currently a ZJU Plan 100 Professor in the College of Computer Science, Zhejiang University, Hangzhou, China. Her research interests include indexing and querying metric spaces, graph databases, and database usability.



Kai Zheng is a Professor of Computer Science with University of Electronic Science and Technology of China. He received his PhD degree in Computer Science from The University of Queensland in 2012. He has been working in the area of spatial-temporal databases, uncertain databases, social-media analysis, and blockchain technologies. He has published over 100 papers in prestigious journals and conferences in data management field such as SIGMOD, ICDE, VLDB Journal, ACM Transactions and IEEE Transactions. He is a senior member of IEEE.

Christian S. Jensen is a Professor of Computer Science at Aalborg University, Denmark. His research concerns data analytics and management with focus on temporal and spatio-temporal data management. Christian is an ACM and an IEEE fellow, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences. He has received several national and international awards for his research, most recently the 2019 TCDE Impact Award. He was Editor-in-Chief of ACM TODS from 2014 to 2020 and an Editor-in-Chief of The VLDB Journal from 2008 to 2014.