SOUP

A fleet management system for passenger demand prediction and competitive taxi supply

Hu, Qi; Ming, Lingfeng; Xi, Ruijie; Chen, Lu; Jensen, Christian S.; Zheng, Bolong

Published in:
Proceedings - 2021 IEEE 37th International Conference on Data Engineering, ICDE 2021

DOI (link to publication from Publisher):
10.1109/ICDE51399.2021.00297

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

SOUP: A Fleet Management System for Passenger Demand Prediction and Competitive Taxi Supply

Qi Hu[1], Lingfeng Ming[1], Ruijie Xi[1], Lu Chen[2], Christian S. Jensen[3], Bolong Zheng[1]
[1]Huazhong University of Science and Technology, Wuhan, China
Email: {huqi11, lingfengming, ruijiexi, bolongzheng}@hust.edu.cn
[2]Zhejiang University, Hangzhou, China
Email: luchen@zju.edu.cn
[3]Aalborg University, Aalborg, Denmark
Email: csj@cs.aau.dk

***Abstract*—Online car-hailing services have gained substantial popularity. An effective taxi fleet management strategy should not only increase taxi utilization by reducing taxi idle time, but should also improve passenger satisfaction by minimizing passenger waiting time. We demonstrate a fleet management system called SOUP that aims at minimizing taxi idle time and that monitors the fleet movement status. SOUP includes a passenger request prediction model called ST-GCSL that predicts the number of requests in the near future, and it includes a demand-aware route planning algorithm called DROP that provides idle taxis with search routes to serve potential requests. In addition, SOUP supports visualizing and analyzing historical passenger requests, simulating fleet movement, and computing evaluation metrics. We demonstrate how SOUP accurately predicts passenger demand and significantly reduces taxi idle time.**

I. INTRODUCTION

The near-ubiquitous deployment of smartphones has enabled transportation network companies such as Didi Chuxing [1] and Uber [2] to operate ride-hailing platforms that enable the servicing of transportation requests by means of fleets of drivers. In these platforms, drivers accept requests and move to the origins of requests to complete the requests. Such platforms have reduced the time drivers are idle and spend waiting to service prospective passengers, thus improving the transportation 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 demand prediction, supply and demand allocation, and route planning.

We focus on the passenger demand prediction and competitive taxi supply problem, which includes the prediction of passenger requests and the planning routes for taxis to follow in order to serve requests in a manner that minimizes the average idle time across all taxis. We propose a data-driven solution that assigns a route to a taxi as soon as the taxi becomes idle such that it can serve a request quickly. Overall, we address two sub-problems:

(i) **Dynamic request patterns.** In order to help taxis serve new requests quickly, we need to know the request availability across the road network of a city.
(ii) **Competition among taxis.** If all taxis tend to move towards hot regions with many requests to find new requests, they will compete due to a high supply-demand ratio, which causes the so-called "herding" effect.

Our fleet management system SOUP solves the above problems. We choose carefully a spatial granularity for partitioning a road network and temporal granularity for partitioning time in order to achieve accurate predictions of requests. We then build an end-to-end deep learning model, called spatial-temporal graph convolutional sequential learning, ST-GCSL, to predict future requests. To eliminate the "herding" effect, we develop a demand-aware route planning algorithm, DROP, that assigns taxis to destinations with a supply-demand balance.

The major contributions are summarized as follows:

- We demonstrate SOUP that effectively improves taxi utilization and passenger satisfaction.
- We present the request prediction model ST-GCSL that predicts near-future passenger demand, and we present DROP that provides passenger-search routes for idle taxis.
- We utilize request analysis and visualization functions to analyze historical passenger request datasets and to monitor fleet movement status.

The remainder of this paper is organized as follows. The related work is covered in Section II. We detail the problem addressed in Section III. Section IV presents a system overview of SOUP and introduces the major components in detail. Finally, Section V demonstrates the functionalities of SOUP, including request analysis, request prediction, and simulation.

II. RELATED WORK

Traffic demand prediction is a critical aspect of when aiming to achieve an efficient transportation system. Existing traffic demand prediction models can be divided into Convolutional Neural Network-based (CNN-based) models and Graph Convolutional Network-based (GCN-based) models. CNN-based models, such as ConvLSTM [8], utilize CNNs to extract spatial dependencies, but they only model Euclidean relations among grid regions and ignore the non-Euclidean relations. In contrast, GCNs can extract local features from non-Euclidean structures, resulting in improved performance. For instance,

STGCN [10] combines CNNs and GCNs to capture temporal dependencies and spatial dependencies, respectively. Further, STG2Seq [3] uses a multiple gate graph convolution module to capture spatial-temporal dependencies. These GCN-based models are flexible and progressive, but most of them to some extent miss short-term spatial-temporal dependencies. Unlike the existing models, ST-GCSL captures all dependencies simultaneously and incorporates context features to improve prediction accuracy.

For the taxi reposition problem, existing work can be divided into combination optimization methods and deep reinforcement learning (DRL) methods. For example, one study [9] models the driver repositioning task as a classical Minimum Cost Flow (MCF) problem and then solves it by combination optimization. MCF-FM [6] develops a continuous order dispatch strategy for an effective fleet management. One study [7] uses a Deep Q-Network (DQN) to learn the optimal dispatch policy from a simulated environment. These methods disregard the real-time nature of supply and demand, which hurts their performances. SOUP incorporates a demand-aware route planning algorithm DROP to reposition idle taxis based on real-time supply and demand distributions.

III. PRELIMINARIES

Settings. The setting of SOUP is a fleet of taxis $\mathcal{A} = \{a_i\}$ that serve a set of passenger requests $\mathbf{\Omega} = \{\omega_j\}$ on a road network that is modeled as a weighted and directed graph $G = (V, E, W)$, where V is the node set, E is the edge set, and $W : E \to \mathbb{R}$ assigns a weight to each edge. All taxis are in the system from the beginning, and each locates at a random location in the road network. Taxis are labeled empty and travel along a *search route* provided by our system. Requests are introduced into the system in a streaming fashion, each with a origin and a destination. A taxi needs to arrive at the origin of a request within a fixed time window (for pick-up) and then moves to the destination (for drop-off); otherwise, the request is removed from the system, an outcome called request expiration. When a request enters the system, the system assigns the nearest empty taxi to the request if the taxi can reach the request's origin within the time window.

The goal of SOUP is to plan search routes for taxis to serve new requests when they become empty such that the idle time is minimized. To solve this problem, we consider two sub-problems.

Passenger Demand Prediction. Given a historical set of requests, we aim to build a request data model to predict the numbers of requests at different locations and times.

Competitive Taxi Supply. Given a request data model, a fleet of taxis with original locations, and a stream of requests, we aim to plan search routes for taxis to serve potential requests and avoid the competition such that the average idle time of taxis is reduced.

IV. SYSTEM OVERVIEW

The SOUP framework encompasses three major components: (1) Spatial-temporal partitioning, (2) request prediction, (3) route planning. A detailed description of SOUP can be found in [11].

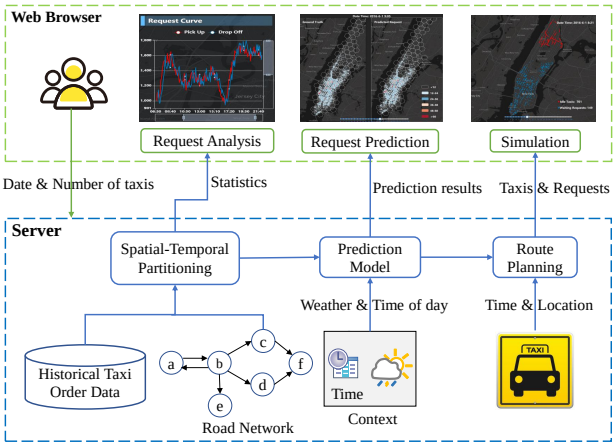


Fig. 1. System Overview

A. SOUP Framework

Fig. 1 shows the SOUP framework, which adopts a browser-server model. The browser side provides three components for visualization and analysis: request analysis, request prediction visualization, and simulation. The server side consists of three modules: spatial-temporal partitioning, request prediction, and route planning. The spatial-temporal partitioning module is responsible for counting the number of requests in each region and time slot. The request prediction module predicts near-future requests. The route planning module provides a search route for each idle taxi based on the predicted requests.

SOUP allows users to select a date and a fleet cardinality. The spatial-temporal partitioning module partitions the road network into hexagon regions and partitions a day into time slots. The number of historical passenger requests of each region and time slot is computed and then sent to the browser side. There, the user can visualize and analyze the data through the request analysis module. Next, the prediction model uses the request data and context features (e.g., weather, events) to predict the number of requests for each region in the next time slot. The prediction is then used by the route planning module. The route planning module plans a search route for each idle taxi based on its current location and the prediction results, so as to guide taxis to locations where requests are anticipated. The simulation module monitors the movement of the fleet of taxis under the assumption that all taxis follow their assigned search routes.

B. Spatial-Temporal Partitioning

We partition the road network into n hexagon regions with the Uber H3 library[1] and denote the set of regions by $R = \{r_1, r_2, \ldots, r_n\}$. We partition a day into m time slots and denote the set of time slots by $T = \{t_1, t_2, \ldots, t_m\}$. Let

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

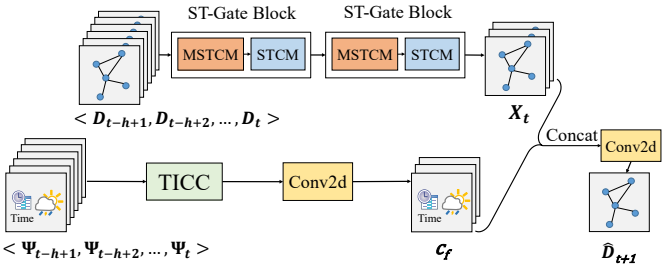
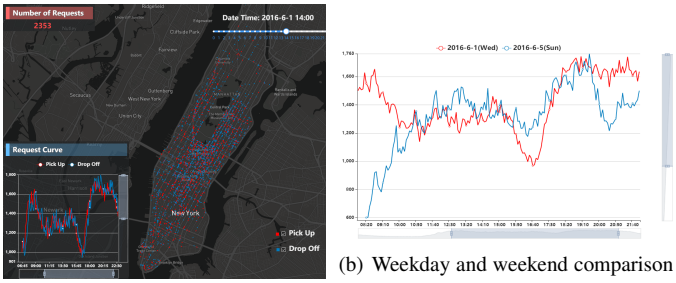


Fig. 2. The structure of ST-GCSL



(a) Plotting pick-up and drop-off

(b) Weekday and weekend comparison

Fig. 3. Request Analysis Module

D_j^i denote the number of requests in region r_i in time slot t_j. Let $\langle \mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_m \rangle$ be a request sequence, where each $\mathbf{D}_j = \{D_j^1, D_j^2, \ldots, D_j^n\}$ denotes the number of requests of all regions in time slot t_j.

C. Request Prediction

To model the historical request patterns, we observe that three types of dependencies should be considered: spatial, temporal, and short-term spatial-temporal dependencies. Existing models disregard the spatial or the short-term spatial-temporal dependencies. We propose ST-GCSL to capture all three types of dependencies while utilizing the context features for improving the prediction accuracy.

Region Correlation Graph. Since the spatial dependency between regions can be captured accurately by a topology structure rather than by the Euclidean space [3], we transform the request prediction problem into a graph node prediction problem. To model the correlations among regions, we build a region correlation graph $\mathcal{G} = (R, A)$, where the set of nodes R is the region set introduced earlier and A is the edge set of $\mathcal{G}$ in the form of an adjacency matrix. To define A, we consider both the geographical and semantic neighborhoods between regions [11].

ST-GCSL Model. The structure of ST-GCSL is shown in Fig. 2. At the current time t, the input of ST-GCSL has two parts, where the first part is the historical request sequence of the h most recent time steps, i.e., $\langle \mathbf{D}_{t-h+1}, \mathbf{D}_{t-h+2}, \ldots, \mathbf{D}_t \rangle$, and the second part is the corresponding context feature sequence (such as time of day, day of week, weather, holidays, events), i.e., $\langle \mathbf{\Psi}_{t-h+1}, \mathbf{\Psi}_{t-h+2}, \ldots, \mathbf{\Psi}_t \rangle$. We adopt TICC [4] to process the context features. Then, we concatenate the two parts to enhance the node feature. The output is request predictions for the next time step, i.e., $\hat{\mathbf{D}}_{t+1} \in \mathbb{R}^N$. The Spatial-Temporal Convolutional Module (STCM) in a Spatial-Temporal Gate Block (ST-Gate Block) is designed to extract long-term spatial-temporal dependencies, while the Multiple Spatial-Temporal Convolutional Module (MSTCM) stacked by several STCM is used to extract short-term spatial-temporal dependencies. The ST-Gate Block can easily adapt to complicated problems and can efficiently extract spatial-temporal dependencies.

D. Route Planning

The route planning algorithm DROP computes a search route for a taxi when the taxi becomes idle. The idea of DROP is to dispatch taxis to regions based on the predicted request distribution such that the supply and demand is balanced across regions.

Candidate Region Generation. To reduce the search space, we only consider the L-order neighbor regions. Specifically, for a taxi's current region r, let $R_L(r)$ represent the L-order neighbors of r. We add all regions from $R_0(r)$ to $R_L(r)$ to form the candidate region set R^*.

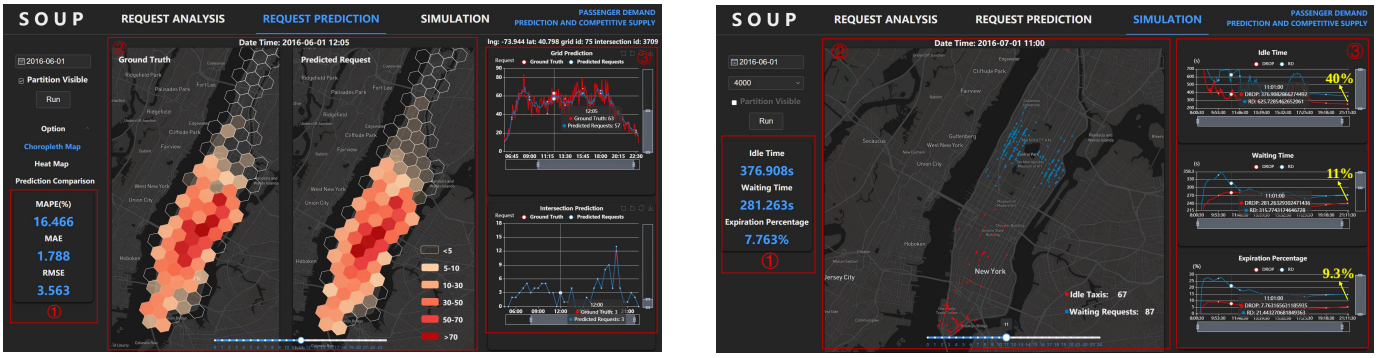
Destination Region Determination. In order to dispatch taxis according to the request distribution, we compute a score $Score(r_i)$ that represents the popularity of region r_i based on the number of request origins and destinations within r_i [11]. The intuition is that the more request origins that locate in a region r_i, the more popular r_i is. In contrast, the more request destinations that locate in a region r_i, the less popular r_i is. Then we use roulette wheel selection to sample a destination region [5]. For a candidate region r_i, the probability of r_i being sampled is $\frac{Score(r_i)}{\sum_{r \in R^*} Score(r)}$.

Planning Search Route. To determine the destination, we first select road intersections with the largest predicted number of requests from the destination region as potential destinations. Then we again apply the roulette wheel selection to sample an intersection as the destination. Finally, a search route directed to this intersection along the shortest travel-time path is planned for the taxi.

V. DEMONSTRATION

We use the New York TLC Trip Record YELLOW Data[2] to demonstrate the functionalities of SOUP. This dataset contains records with both pick-up and drop-off information for yellow taxis in New York City. We use the data from 8:00 to 22:00 on June 1st, 2016 for demonstration. The underlying road network is taken from OpenStreetMap[3] and contains 4,360 nodes and 9,542 edges. To evaluate ST-GCSL, we use three well-adopted metrics: Mean Average Percentage Error (MAPE), Mean Absolute Error (MAE), and Rooted Mean Square Error (RMSE). To evaluate DROP, we use three

[2]https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[3]https://www.openstreetmap.org/



(a) Request Prediction

(b) Simulation

Fig. 4. Demonstration of SOUP

metrics: the average taxi idle time, the average request waiting time, and the request expiration percentage.

A. Request Analysis

The passenger request distribution information is shown in Fig. 3. The red and blue dots in Fig. 3(a) represent pick-up and drop-off points, respectively. Requests are located in midtown and lower Manhattan, and very few requests exist in uptown. Further, SOUP provides a request curve to show the changing trend of pick-ups and drop-offs during a day: there are two peak periods of requests, one in the morning and one in the evening. SOUP provides a time line that users can move in order to quickly view the changing trend of requests during the day. The line chart in Fig. 3(b) compares the request distributions on weekdays and weekends.

B. Request Prediction

SOUP supports the visualization of prediction results. The user interface of the request prediction module is shown in Fig. 4(a). The two choropleth maps in Fig. 4(a)-② show the distribution of the ground truth data on June 1st, 2016 and the distribution predicted by ST-GCSL. Fig. 4(a)-① computes the values of the three metrics—these enable the user to evaluate the performance of the prediction model. The line charts in Fig. 4(a)-③ enable comparison between the ground truth and predicted values. The user can adjust the time period by dragging the slider below, thus viewing the accuracy of the prediction results at different levels of detail.

C. Simulation

The simulation module visualizes the real-time locations of idle taxis and waiting requests. Fig. 4(b) gives a snapshot of the simulation process. Users can select the date and the number of taxis, then click the "Run" button to start the simulation. The real-time locations of idle taxis and waiting requests are updated dynamically on the map. Red dots represent idle taxis, and blue dots represent waiting requests. It is easy to see that idle taxis should be directed to midtown, where most waiting requests are located. We also show the values of the three evaluation metrics in Fig. 4(b)-①. The line charts in Fig. 4(b)-③ compare DROP and the baseline RD [5] according to the three metrics. DROP can reduce the average taxi idle time by 40% and the average request waiting time by 11%. The expiration percentage is reduced to 5.8%, an improvement of 9.3% over RD.

VI. CONCLUSION

We demonstrate the fleet management system SOUP that supports passenger demand prediction and competitive taxi supply. SOUP encompasses a prediction model called ST-GCSL that predicts the number of passenger requests in the near future, and it encompasses a demand-aware route planning algorithm called DROP that provides search routes for idle taxis based on predicted passenger requests. We demonstrate SOUP on a real dataset, showing that it is capable of good performance.

REFERENCES

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

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

[3] 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.

[4] 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, 2018.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, pages 3634–3640, 2018.

[11] B. Zheng, Q. Hu, L. Ming, J. Hu, L. Chen, K. Zheng, and C. S. Jensen. Spatial-temporal demand forecasting and competitive supply via graph convolutional networks. *CoRR*, abs/2009.12157, 2020.