

# A Queueing-Theoretic Framework for Vehicle Dispatching in Dynamic Car-Hailing

Peng Cheng  
East China Normal University  
Shanghai, China  
pcheng@sei.ecnu.edu.cn

Jiabao Jin  
East China Normal University  
Shanghai, China  
10175101146@stu.ecnu.edu.cn

Lei Chen  
The Hong Kong University of Science  
and Technology  
Hong Kong, China  
leichen@cse.ust.hk

Xuemin Lin  
The University of New South Wales  
Sydney, Australia  
lxue@cse.unsw.edu.au

Libin Zheng  
Guangdong Key Laboratory of Big  
Data Analysis and Processing, Sun  
Yat-sen University  
Guangzhou, China  
zhenglb6@mail.sysu.edu.cn

## ABSTRACT

With the rapid development of smart mobile devices, the car-hailing platforms (e.g., Uber or Lyft) have attracted much attention from the academia and the industry. In this paper, we consider a dynamic car-hailing problem, namely *maximum revenue vehicle dispatching* (MRVD), in which rider requests dynamically arrive and drivers need to serve riders such that the entire revenue of the platform is maximized. We prove that the MRVD problem is NP-hard and intractable. To handle the MRVD problem, we propose a queueing-based vehicle dispatching framework, which first uses existing machine learning models to predict the future vehicle demand of each region, then estimates the idle time periods of drivers through a double-sided queueing model for each region. With the information of the predicted vehicle demands and estimated idle time periods of drivers, we propose two batch-based vehicle dispatching algorithms to efficiently assign suitable drivers to riders such that the expected overall revenue of the platform is maximized during each batch processing. Through extensive experiments, we demonstrate the efficiency and effectiveness of our proposed approaches over both real and synthetic datasets. In summary, our methods can achieve 3% ~ 10% increase on overall revenue without sacrificing on running speed compared with the state-of-the-art solutions.

## PVLDB Reference Format:

Peng Cheng, Jiabao Jin, Lei Chen, Xuemin Lin, and Libin Zheng. A Queueing-Theoretic Framework for Vehicle Dispatching in Dynamic Car-Hailing. PVLDB, 14(11): 2177 - 2189, 2021.  
doi:10.14778/3476249.3476271

## PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at <https://github.com/inabao/queue-based-order-dispatching>.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.  
Proceedings of the VLDB Endowment, Vol. 14, No. 11 ISSN 2150-8097.  
doi:10.14778/3476249.3476271

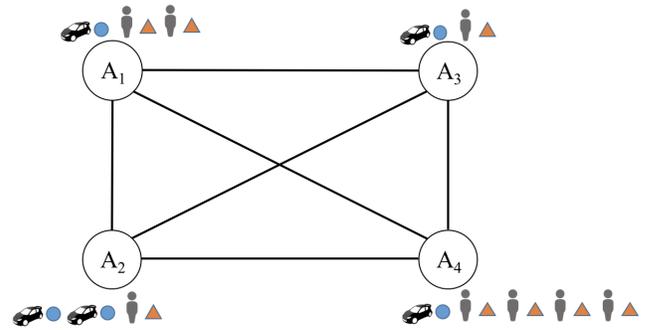


Figure 1: A multi-area vehicle dispatching example.

## 1 INTRODUCTION

Recently, with the popularity of the smart devices and high quality of the wireless networks, people can easily access network and communicate with online services. With the convenient car-hailing platforms (e.g., Uber [5] and DiDi Chuxing [1]), drivers can share their vehicles to riders to obtain monetary benefits and alleviate the pressure of public transportation. One of the crucial issues in the platforms is to efficiently dispatch vehicles to suitable riders. Although the platforms become huge recently, during peak hours (e.g., 8 am) in some high demand areas (e.g., residential areas), riders need to wait for up to several hours before being served. To mitigate the shortage of vehicles in particular time and areas and improve the efficiency of the platforms, we investigate a queueing-theoretic framework in this paper.

We illustrate the general idea of our framework in the following motivation example.

**Example 1.** Consider a scenario of taxi dispatching in Figure 1, where each one of the four connected areas  $A_1 \sim A_4$  maintains a queue of riders and taxis. The numbers of icons of taxis and riders near each area reflect the ratio between them in the corresponding area. For example, in area  $A_1$ , the number of available taxis is only half of the number of waiting riders. Riders only want to wait for a limited time (e.g., 5 minutes), otherwise they will switch to other public transportation systems (e.g., the bus system). Usually, taxis

can easily pickup riders within the same area (e.g., moving several hundreds meters). However, if one taxi is assigned with a rider in a different area, the taxi may need to move several kilometers to pickup the rider. In addition, after serving its current rider, one taxi will usually wander around the destination area to pickup a new rider.

When the number of taxis is less than the demanding riders, the platform needs to smartly select riders to serve, such that the total revenue of the platform can be improved. In the example in Figure 1, the platform should give higher priorities to the riders whose destination is within area  $A_1$  or  $A_4$ , where taxis are scarce. On the contrast, the platform should give lower priorities to the riders whose destination is within area  $A_2$ , where taxis are abundant.

Motivated by the example above, in this paper, we propose a novel vehicle dispatching framework, which aims to take riders' destinations into consideration to alleviate the shortage of taxis in particular areas such that the overall revenue of the platform can be maximized.

Existing works in spatial matching or taxi dispatching only consider the pickup locations of riders and try to minimize the travel distance of taxis to pick riders [23, 25], which causes some taxis need to wait for a long time period before picking up new convenient riders after finishing their last orders, which leads to the low efficiency of the platform (i.e., imbalanced demand-and-supply in some regions). In our previous poster paper [8], to the best of our knowledge, we are the first to introduce the general idea of the queueing theoretic framework to balance the demand-and-supply of taxi dispatching during a relative long time period to maximize the overall revenue of the platform. However, in [8] we just explained the general queueing theoretic framework without detailed algorithms, analyses, and experimental studies, which will be introduced in this paper.

To improve the overall revenue of the platform during a relatively long time period, we propose a *batch-based queueing-theoretic vehicle dispatching* framework in this paper. Specifically, we partition the whole space into regions and maintain a queue of waiting riders and drivers for each region. Once there are available drivers, the most-priority rider in the queue of waiting riders will be served. In addition, riders may *quit/renege* from the platform if she is not served for a long period. Usually, a driver, after delivering her current rider, will continue to serve the next rider around the destination of the current rider. Thus, through serving the riders, the distribution of drivers will be changed. To serve as many riders as possible, intuitively, the platform should *match* the distributions of riders and drivers (i.e., one region with more riders should have more drivers) through associating higher priorities to the riders whose destinations are in regions lacking of drivers, then drivers can serve riders quickly before they quit/renege. We first propose models to estimate the Poisson distributions of riders and drivers. Then, we utilize the queueing theory to analyzing the idle time interval for each driver after finishing his/her assigned rider. Finally, we propose two vehicle/driver dispatching algorithms to maximize the overall revenue of the platform in each batch processing. Note that, we maximize the overall revenue of the platform through improving the efficiency of the entire platform to serve more riders without increasing the charges to riders or decreasing the payment to drivers. In fact, the payment to drivers is usually a portion of the

overall revenue of the platform. Thus, the more the overall revenue of the platform is, the more the payment to drivers is. In conclusion, our solution will benefit riders, drivers and the platform at the same time.

To summarize, we make the following contributions in the paper:

- We propose a batch-based queueing theoretic framework for vehicle dispatching in Section 3.
- We estimate the idle interval time of drivers in Section 4.
- We propose two vehicle dispatching algorithms for each batch processing in Section 5.
- We have conducted extensive experiments on real and synthetic data sets, to show the efficiency and effectiveness of our queueing-theoretic framework in Section 6.

In addition, the remaining sections of the paper are arranged as follows. We review and compare previous studies on queueing theory and vehicle dispatching in Section 7 and conclude the work in Section 8.

## 2 PROBLEM DEFINITION

In this section, we present the formal definition of the vehicle dispatching problem, where a system will assign drivers to riders to deliver them to their destinations.

In this paper, we use a graph  $G = \langle V, E \rangle$  to represent a road network, where  $V$  is a set of vertices and  $E$  is a set of edges. Each edge  $(u, v) \in E$  ( $u, v \in V$ ) is associated with a weight  $cost(u, v)$  indicating the travel cost from vertex  $u$  to vertex  $v$ . Here, the travel cost could be the travel time or the travel distance. When we know the travel speed of vehicles, we can convert one to another. In the rest of this paper, we will not differentiate between them and use *travel cost* consistently.

To better manage the riders and drivers, we assume the entire space is divided into a set of  $n$  regions/grids  $A = \{a_1, a_2, \dots, a_n\}$ .

### 2.1 Riders and Drivers

**Definition 1** (Impatient Rider). Let  $r_i$  be an impatient rider, who submit his/her order  $o_i$  to the platform at timestamp  $t_i$ , and is associated with a source location  $s_i$ , a destination location  $e_i$  and a pickup deadline  $\tau_i$ .

In particular, a rider  $r_i$  comes to the platform to call for one and only one driver to deliver him/her from his/her current location  $s_i$  to his/her destination location  $e_i$ . The request is sent to the platform at timestamp  $t_i$ . As the rider is impatient, if the platform cannot assign an available driver to pick up him/her within  $\tau_i$  time after  $t_i$ , he/she will quit/renege from the platform and may switch to other platforms or use other transportation systems. Usually, rider  $r_i$  will not rejoin the platform immediately after he/she is delivered to his/her destination. Thus, in this paper, we assume that each rider is unique and only lives for the lifetime of his/her ride in the platform. In addition, if a rider  $r_i$  is delivered to his/her destination, the platform will charge him/her for  $\alpha \cdot cost(s_i, e_i)$ , where  $\alpha$  is the travel fee rate of the platform.

**Definition 2** (Driver). Let  $d_j$  be a driver, who is located at position  $l_j(t)$  at timestamp  $t$ . His/her status is either *busy* (i.e., on delivering any riders) or *available* (i.e., free to be assigned to a rider).

When a new driver  $d_j$  joins the platform, he/she is considered to be available to serve riders. Once a rider  $r_i$  is assigned to a driver  $d_j$ , the driver will move to the source location  $s_i$  to pickup rider  $r_i$ , then send rider  $r_i$  to his/her destination location  $e_i$ . During that period, driver  $d_j$  is considered *busy*. After driver  $d_j$  finishes his/her current task,  $d_j$  will become available again. For region  $a_k$  at timestamp  $t$ , we denote the set of available drivers as  $D_k(t)$  and the number of them as  $|D_k(t)|$ .

## 2.2 The Maximum Revenue Vehicle Dispatching Problem

Before presenting the formal definition of the maximum revenue vehicle dispatching problem, we first define the valid rider-and-driver dispatching pair.

**Definition 3** (Valid Rider-and-Driver Dispatching Pair). Let  $(r_i, d_j)$  be a valid rider-and-driver dispatching pair, where driver  $d_j$  can arrive at the pickup location  $s_i$  of rider  $r_i$  before the pickup deadline  $\tau_i$  and driver  $d_j$  is in available status when he/she is picking up rider  $r_i$ .

Now we give the formal definition of the maximum revenue vehicle dispatching problem as follows:

**Definition 4** (Maximum Revenue Vehicle Dispatching Problem, MRVD). For a given time period  $\mathbb{T}$ , a set of impatient riders  $R_{\mathbb{T}}$  and a set of drivers  $D_{\mathbb{T}}$ , the maximum revenue vehicle dispatching problem is to select a set,  $I_{\mathbb{T}}$ , of *valid rider-and-driver dispatching pairs* such that the overall revenue of the platform is maximized, which is:

$$\max \sum_{(r_i, d_j) \in I_{\mathbb{T}}} \alpha \cdot \text{cost}(s_i, e_i), \quad (1)$$

where  $\alpha$  is the travel fee rate of the platform.

Intuitively, to maximize the overall revenue the platform should serve as many long travel distance riders as possible as shown in Equation 1. However, the platform has no control on riders (i.e., the platform cannot schedule the arrivals and enlarge the waiting deadlines of riders), and can only affect the behaviors of drivers (i.e., dispatching drivers to pickup different riders). We will have a reduction in the end of this section to show practical rules to dispatching drivers to maximize the overall revenue of the platform.

## 2.3 Hardness of MRVD

We prove MRVD is NP-hard through a reduction from a variant of traveling salesman problem (TSP), the deadline TSP [7], which is a known NP-hard problem.

**Theorem 2.1.** (*Hardness of MRVD*) *The problem of maximum revenue vehicle dispatching (MRVD) is NP-hard.*

**PROOF.** We prove the theorem by a reduction from the deadline TSP [7]. A deadline TSP problem can be described as follows: given a set of nodes  $V$ , each node  $v_i \in V$  is located at location  $v_i^l$  with a deadline  $v_i^d$ . There is a salesman  $s$  locating at position  $s^l$  at the beginning, who wants to visit the nodes. If  $s$  can visit node  $v_i$  before its deadline  $v_i^d$ ,  $s$  will receive a reward  $v_i^r$ . The problem is to find a path for  $s$  to visit nodes such that the total reward is maximized.

**Table 1: Symbols and Descriptions.**

Symbol	Description
$a_k$	a region/grid
$r_i$	an impatient rider
$o_i$	the order of the impatient rider $r_i$
$t_i$	the timestamp when $r_i$ posts her ride request
$s_i$	the source location of rider $r_i$
$e_i$	the destination location of rider $r_i$
$\tau_i$	the pickup deadline of rider $r_i$
$d_j$	a driver
$l_j(t)$	the position of driver $d_j$ at timestamp $t$
$D_k(t)$	a set of available drivers in region $a_k$ at time $t$

For the deadline TSP instance, we can transform it to an instance of MRVD as follows: we give only one driver  $d_j$  with unlimited lifetime located at position  $s^l$  at the very beginning. In addition, for each node  $v_i \in V$ , we generate a rider  $r_i$ , who is located at  $v_i^l$  with a pickup deadline of  $v_i^d$ . All the riders post their orders at the beginning of time. We set the travel fee rate  $\alpha$  to a large enough value such that the destination  $e_i$  of each rider  $r_i$  is very close to his/her origin location  $v_i^l$  and the travel time of serving  $r_i$  can be ignored. In addition, the travel cost  $\alpha \cdot \text{cost}(v_i^l, e_i)$  is equal to the visiting reward of the corresponding node  $v_i^r$ . Then, for this MRVD instance, we want to arrange a schedule for the given driver such that his/her overall revenue is maximized.

Thus, to maximize the overall revenue satisfying the pickup deadlines of riders is same to maximize the total reward in the deadline TSP problem.

Given this mapping, it shows that the deadline TSP instance can be solved if and only if the corresponding MRVD problem instance can be solved. This way, we reduce the deadline TSP to the MRVD problem. Since the deadline TSP is known to be NP-hard [7], MRVD is also NP-hard, which completes our proof.  $\square$

In real platform, orders are created at different timestamps, which means our MRVD problem is an online problem. To evaluate the effectiveness of algorithms on online problems, competitive ratio is a common used metric, which is the ratio of the result achieved by an online algorithm to the optimal result achieved in the corresponding offline problem. However, in the existing study about online deadline TSP problem [28], the authors prove that there is no algorithm can achieve a constant competitive ratio for online deadline TSP problem even the arriving timestamps of orders are known in advance. Moreover, in MRVD, usually multiple drivers need to be arranged, which means that the MRVD problem is more complex than the online deadline TSP problem. Thus, we turn to use the experimental results to show the effectiveness of our approaches.

## 2.4 Reductions of MRVD

Let  $T_j$  be the lifetime of driver  $d_j$  from the time he/she joins to the time he/she exits the platform. During  $T_j$ , the status of driver  $d_j$  keeps switching between *available* and *busy*. We notice that only when driver  $d_j$  is in *busy* status, he/she contributes to the overall revenue of the platform. Then, we can rewrite the objective function of MRVD as below:

$$\begin{aligned}
& \max \sum_{\langle r_i, d_j \rangle \in I_{\mathbb{T}}} \alpha \cdot \text{cost}(s_i, e_i) \\
\Rightarrow & \max \sum_{d_j \in D_{\mathbb{T}}} \sum_{r_i \in R_j} \alpha \cdot \text{cost}(s_i, e_i) \\
\Rightarrow & \max \alpha \sum_{d_j \in D_{\mathbb{T}}} \sum_{r_i \in R_j} \text{cost}(s_i, e_i) \quad (2)
\end{aligned}$$

where  $D_{\mathbb{T}}$  is the set of drivers on the platform during the given time period  $\mathbb{T}$ , and  $R_j$  is the set of riders that are served by driver  $d_j$  in the selected set,  $I_{\mathbb{T}}$ , of *valid rider-and-driver dispatching pairs*. According to Equation 2, the platform should maximize the length of the total busy time of each driver to maximize its overall revenue. Since the lifetime  $T_j$  of each driver  $d_j$  is fixed, to maximize his/her total *busy* time,  $\sum_{r_i \in R_j} \text{cost}(s_i, e_i)$ , is equivalent to minimize his/her total *idle* time,  $T_j - \sum_{r_i \in R_j} \text{cost}(s_i, e_i)$ . Then, the objective of MRVD can be rewritten as follows:

$$\begin{aligned}
& \max \alpha \sum_{d_j \in D_{\mathbb{T}}} \sum_{r_i \in R_j} \text{cost}(s_i, e_i) \\
\Rightarrow & \min \sum_{d_j \in D_{\mathbb{T}}} (T_j - \sum_{r_i \in R_j} \text{cost}(s_i, e_i)) \\
\Rightarrow & \min \sum_{d_j \in D_{\mathbb{T}}} \sum_{i=0}^{|R_j|} \psi_{ij} \quad (3)
\end{aligned}$$

where  $\psi_{ij}$  is the idle time of driver  $d_j$  after delivering rider  $r_i$ . Here,  $\psi_{0j}$  indicates the idle time of driver  $d_j$  before picking up his/her first rider.

According to Equation (3), to maximize the overall revenue, the platform intuitively should reduce the number of served riders (e.g.,  $|R_j|$ ) and the idle time interval (e.g.,  $\psi_{ij}$ ) between any two consecutive riders for each driver. It may be confused that reducing the number of served riders for each driver seems to contradict the goal of maximizing the overall revenue. To explain this contradiction, we denote the time period of serving a rider and the idle time before serving the next rider as a *service round* for a driver. In fact, the lifetime of a driver  $d_j$  is fixed as  $T_j$ , when driver  $d_j$  serves fewer riders, the average length of service rounds will be longer (i.e.,  $\frac{T_j}{|R_j|}$ ). Then, for a service round that driver  $d_j$  is assigned to serve rider  $r_i$ , minimizing the idle time interval  $\psi_{ij}$  will lead to that the travel cost  $\text{cost}(s_i, e_i)$  increases, which agrees with the intuition from Equation (2). In conclusion, we can have two practical and controllable rules for the platform in its online processes to maximize the overall revenue during a given time period  $\mathbb{T}$ : **a) associating higher priorities to the riders whose travel costs are high; b) reducing the length of the idle time between serving any two consecutive riders for each driver.**

In the rest of this paper, we propose a queueing-theoretic framework, taking into consideration of the travel cost and the idle time length after each ride request, to maximize the overall revenue of the platform during a given time period.

### 3 OVERVIEW OF QUEUEING-BASED VEHICLE DISPATCHING FRAMEWORK

In this section, we introduce an overview of our queueing-based vehicle dispatching framework. In general, the framework includes

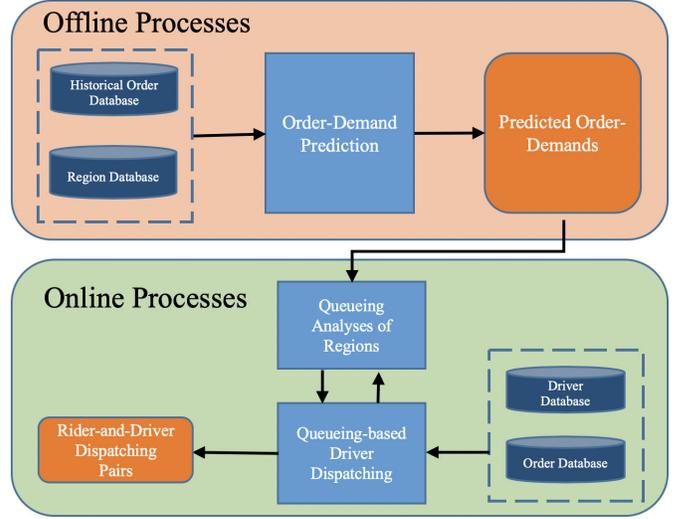


Figure 2: Illustration of the Framework Work Flow.

three parts: vehicle demand-supply prediction, queueing analysis of each region and queueing-based vehicle dispatching algorithms. From the other perspective, our framework contains offline prediction processes and online analysing and dispatching processes as shown in Figure 2. In the offline processes, the platform predicts the number of orders for each region in each time periods based on the historical order records. In the online processes, the platform utilizes the predicted order demands and supplies to assign available drivers to orders with a goal to maximizing the overall revenue of the platform subjected to the deadline constraint of orders. In addition, the driver dispatching process and the queueing analyses process can affect each other, thus we interactively assign orders to drivers and update the results of queueing analyses of regions.

We first briefly introduce the major parts of our queueing-based vehicle dispatching framework, then propose a batch-based vehicle dispatching algorithm to handle the orders from riders.

#### 3.1 Major Parts of Queueing-Based Vehicle Dispatching Framework

Our queueing-based vehicle dispatching framework includes three major parts: offline vehicle demand-supply prediction, region queueing analysis and queueing-based vehicle dispatching algorithm.

**3.1.1 Offline Vehicle Demand-Supply Prediction.** In our framework, we predict the order demand of each region for given time periods. For the rejoined drivers, we can estimate their availability based on their assignments and travel costs. In practice, it is hard to predict the accurate location and timestamp of a particular rider since the uncertain behaviors of a single user. To utilize the distribution of riders, we predict the number of riders for a given region (i.e., a spatial range of area, such as square regions or hexagon regions) in a given time period (i.e., next 30 minutes). Existing work can be applied offline to predict the demand of riders in given regions and time periods, such as demand-supply prediction of traffic [13, 22], and spatial-temporal data prediction [16, 30]. In this paper, we test the representative prediction algorithms, e.g., Historical

Average (HA) method, Linear Regression (LR) method, Gradient Based Regression Tree (GBRT) method [18] and DeepST [29] on the real-world taxi demand-supply dataset and select the most effective one, DeepST [29], for our offline demand-supply prediction process, which can achieve very accurate order demand prediction results (i.e., 2.3 % RMSE) on our testing data set. Specifically, DeepST uses Convolutional Neural Network (CNN) [21] on historical data of order counts and meta data (e.g., time of day, day of week and city weather) to predict the order demand for each region in each time slot (e.g., a period of 30 minutes). Due to space limitation, we put the detailed comparison of the spatial temporal prediction models in Appendix A of our technical report [9].

**3.1.2 Region Queueing Analysis.** The available drivers in a region  $a_x$  in a time period  $\mathbb{T}$  come from the rejoined active drivers and unassigned drivers in the previous batch. With the predicted numbers of orders for the region  $a_x$  in a given time period  $\mathbb{T}$  and the schedules of active drivers, we can know the demand and supply of drivers for the region  $a_x$  in time period  $\mathbb{T}$ . We estimate the waiting times (idle time intervals) for vehicles from finishing last order to receiving next order in Section 4. According to the analyses in Section 2, shorter idle time intervals are better. Thus, the estimated waiting times of vehicles can be used to guide the order dispatching process to achieve a high overall revenue.

**3.1.3 Queueing-Based Vehicle Dispatching.** The platform needs to dispatch drivers to serve most “valuable” riders with high priorities. According to the analyses in Section 2, orders having high travel costs and ending in “hot regions” (i.e., regions with many future orders) can contribute more to the platform, which should be associated with high priorities. In addition, since drivers usually prefer to serve riders close to their locations after they finish the last orders, the platform’s selection on serving orders will affect the vehicle supply in future, which in turn will affect the queueing analyses of the related regions. In Example 1, if the platform dispatch a driver to serve a rider having a destination in region  $A_1$ , the driver supply in region  $A_1$  will increase slightly after finishing the order. We propose efficient and effective algorithms in Section 5 to dispatch available drivers to riders with an optimization goal of maximizing the overall revenue of the platform subjected to the deadline constraint of orders.

## 3.2 The Batch-based Vehicle Dispatching Algorithm

To handle the online processes of vehicle dispatching, we propose a batch-based processing framework to iteratively assign drivers to riders every  $\Delta$  seconds. Note that, in real applications (e.g., DiDi Chuxing [1]),  $\Delta$  is set very small (e.g., several seconds) such that the customers cannot notice the delay of the batch processing. To solve the assignment problem in each batch, we propose two heuristic algorithms to greedily maximize the revenue summation of the platform for the current scheduling time period  $[\bar{t}, \bar{t} + t_c]$ , where  $\bar{t}$  indicates the current timestamp and  $t_c$  is the length of the current scheduling time period.

As shown in Algorithm 1, we iteratively assigns drivers to riders for multiple batches with a time interval  $\Delta$  between every two successive batches. Specifically, for a batch starting at timestamp

---

### Algorithm 1: Batch-based Vehicle Dispatching Algorithm

---

**Input:** The overall time period  $\mathbb{T}$   
**Output:** A set of rider-and-driver dispatching pairs within the time period  $\mathbb{T}$

```

1 while current time  $\bar{t}$  is in  $\mathbb{T}$  do
2   foreach  $a_k \in A$  do
3     retrieve the waiting riders in region  $a_k$  to  $R_k$ 
4     retrieve the available drivers in region  $a_k$  to  $D_k$ 
5     predict the number of upcoming riders in region  $a_k$ 
       during  $[\bar{t}, \bar{t} + t_c]$  as  $|\hat{R}_k|$ 
6     count the number of upcoming rejoined drivers in
       region  $a_k$  during  $[\bar{t}, \bar{t} + t_c]$  as  $|\hat{D}_k|$ 
7   use task-priority greedy or local search approach to obtain a
       set of rider-and-driver pairs  $I_{\bar{t}}$ 
8   foreach  $\langle r_i, d_j \rangle \in I_{\bar{t}}$  do
9     inform driver  $d_j$  to pick rider  $r_i$ 
10  wait till  $\bar{t} + \Delta$ 

```

---

$\bar{t}$ , we first retrieve a set,  $R_k$ , of waiting riders and a set,  $D_k$ , of available drivers for each region  $a_k$  (lines 3-4). Here, waiting riders  $R_k$  include the riders that are not assigned with any drivers during the last batch and the newly coming riders after the last batch in region  $a_k$ . Moreover, available drivers  $D_k$  includes the drivers that are not assigned with any riders in the last batch, and the drivers that have finished the previous assigned tasks then rejoin the platform in region  $a_k$ . To estimate the arrival rates of riders and serving rates drivers for the current scheduling time period  $[\bar{t}, \bar{t} + t_c]$ , we predict the number,  $|\hat{R}_k|$ , of upcoming riders and estimate the number,  $|\hat{D}_k|$ , of rejoin drivers in region  $a_k$  (lines 5-6). Then, we use our proposed heuristic vehicle dispatching algorithms to achieve a set  $I_{\bar{t}}$  of rider-and-driver dispatching pairs to greedily maximize the revenue summation of the platform for the current scheduling time period  $[\bar{t}, \bar{t} + t_c]$  (line 7). For every rider-and-driver dispatching pair  $\langle r_i, d_j \rangle$  in  $I_{\bar{t}}$ , we inform the driver  $d_j$  to pick up rider  $r_i$  (lines 8-9). Finally, we wait until the time comes to the next batch  $\bar{t} + \Delta$  (line 10).

In the following sections, we will first introduce the queueing analyses of regions in Section 4, then propose our queueing based vehicle dispatching algorithms in Section 5.

## 4 QUEUEING ANALYSES OF REGIONS

In this section, we analyze the waiting riders for each single region through a queueing model. In queueing theory, customers join queue in an arrival (or “birth”) rate  $\lambda$ , then the platform will serve the customers in a service (“death”) rate  $\mu$ . We first introduce the queue configuration, then estimate the idle time for a driver after he/she finishes his/her current order.

### 4.1 Queue Configuration of a Single Region

In this paper, the platform can be considered as a server to match available drivers and waiting riders in each region. The riders come to the platform and wait for drivers to pick them. However, the riders are impatient and will leave the platform if they are not served before their deadlines. The available drivers come from the

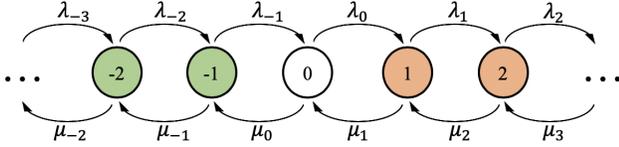


Figure 3: Birth-death chain for the queue of a region.

rejoined active drivers, who are the ones continuing to work on the platform after finishing their assigned orders.

Similar to the previous assumption in the related work [6], we assume the arrival rate of riders (in number per minute) follows the Poisson distribution with rates  $\lambda$  in a region  $a$  during a short time period with length  $t_c$  (e.g., a half hour). In addition, we also model the arrivals of rejoined active drivers follow a Poisson distribution with a rate of  $\mu$  in a region  $a$  during a short time period with length  $t_c$ . Note that, although the arrival rate of riders and drivers may change during different time periods in a day (e.g., 8 to 9 A.M. and 8 to 9 P.M.), to facilitate the analysis of the queueing situation in a short time period (e.g., a half hour), we model the arrival rates of riders and drivers as stable rates. We verify our assumption that the arrivals of orders and rejoined drivers follow Poisson distributions through chi-square ( $\chi^2$ ) tests [19]. Due to the space limitation, please refer to Appendix B of our technical report [9] for details.

When the riders are more than the drivers in a region, the platform will select a subset of riders with “higher priorities” to serve first. Usually a driver will rejoin the platform in the same region of the destination of her/his last served rider. It leads to that the drivers appear in the regions where the destinations of the selected high-priority riders are, and then the arrival rates of drivers in the corresponding “selected” regions will increase. In our queueing model, the priority of a rider is determined in line with his/her travel cost and the demand-supply situation in his/her destination region. According to the analysis in Section 2.4, to improve the overall revenue, the platform prefers to give higher priorities to the riders who have higher travel costs and are going to “hot” regions.

Figure 3 illustrates the birth-death chain of the queueing model for a region, where each circle indicates a state and the numbers in the circles represent the numbers of waiting riders in the corresponding states. For example, the state of 2 indicates that there are 2 waiting riders in the region. Each link represents the transfer event from the tail state to the head state along its direction, where the value close to the link indicates the transfer rate. For example, the link with value  $\lambda_0$  pointing from state 0 to state 1 indicates the transfer rate from state 0 to state 1 is  $\lambda_0$ . Since drivers may also congest in a region, if the arrival rate of drivers is higher than that of riders (i.e.,  $\mu > \lambda$ ), to uniformly represent the queueing situation of a region, we utilize the state of  $-n$  ( $-n < 0$ ) to indicate that there are  $n$  congested drivers in the region.

Another issue to change the number of waiting riders is that the impatient riders may quit from the platform if they are not served after a time period, which is called *reneging* in queueing theory. As defined in the existing work [24], we can define a state related reneging function  $\pi(n)$  as the reneging rate of riders for the state  $\mathbf{n}$  ( $n > 0$ ) of the birth-death chain in Figure 3. As suggested in [24], a good practice for the reneging function  $\pi(n)$  is to define it as  $e^{\beta n/\mu}$ , where  $\beta$  is a parameter determined based on the historical reneging

records in the corresponding region. Then, the death/service rate  $\mu_n$  of the state  $\mathbf{n}$  can be adjusted as follows:

$$\mu_n = \begin{cases} \mu, & n \leq 0 \\ \mu + \pi(n), & n > 0 \end{cases} \quad (4)$$

For the birth/arrival rate  $\lambda_n$  of state  $\mathbf{n}$ , we define it as  $\lambda_n = \lambda$ , since drivers do not renege in our queueing model.

## 4.2 Expected Idle Time Interval of Drivers

In this section, we analyze the expected idle time interval of a driver  $d_j$ . Let region  $a$  be the destination region of the last rider  $r_i$  of  $d_j$  and  $d_j$  will join the queue of region  $a$  after serving  $r_i$ . Thus, the state of the region  $a$  (i.e., the length of waiting drivers or riders) will directly affect the waiting time (idle time) of  $d_j$  before serving the next rider. For example, if the region  $a$  is in a state of  $\mathbf{n}_a$  and  $n_a > 0$ , which means there are  $n_a$  riders are waiting for drivers, driver  $d_j$  can be immediately assigned with a new rider after finishing the last order. On the contrary, if the region  $a$  is in a state of  $\mathbf{n}_b$  and  $n_b < 0$ , which means there are  $n_b$  available drivers are waiting for riders, driver  $d_j$  will not be assigned with any new riders before  $n_b$  available drivers are assigned with riders first. For region  $a$ , we assume the arrives of the riders and rejoined drivers follow Poisson distributions with rates  $\lambda$  and  $\mu$ , respectively. Then, the region  $a$  can be in any state  $\mathbf{n}$  with the corresponding probability  $p_n$ .

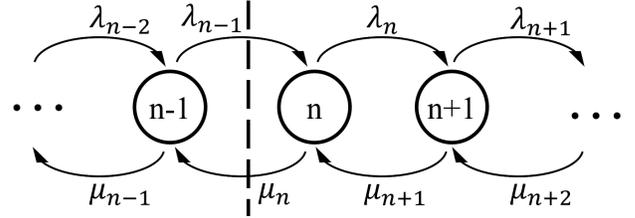


Figure 4: Flow balance between states.

We here briefly introduce the flow balancing rule in analyzing the recursive relation between  $p_n$  and  $p_{n-1}$  [24]. As shown in Figure 4, the mean flows across the dash line must be equal in a steady situation for the queue. In other words, for a relatively long period (e.g., 30 minutes), the rate of transitions ( $\lambda_{n-1}p_{n-1}$ ) from state  $\mathbf{n-1}$  to state  $\mathbf{n}$  must equal the rate of transitions ( $\mu_n p_n$ ) from state  $\mathbf{n}$  to state  $\mathbf{n-1}$ , which is as follows:

$$\mu_n p_n = \lambda_{n-1} p_{n-1} \quad (5)$$

By iteratively applying Equation (5), we can derive:

$$p_n = \begin{cases} p_0 \cdot \left(\frac{\mu}{\lambda}\right)^{-n}, & n < 0 \\ p_0 \cdot \prod_{i=1}^n \frac{\lambda}{\mu + \pi(i)}, & n > 0 \end{cases} \quad (6)$$

Different from the traditional queueing model with only one-sided queue, our queueing model of a single region is a double-sided queue. We need to analyze the particular idle times of drivers with our new queueing model. We estimate the expected idle time in different conditions: 1) more riders arrive; 2) more drivers rejoin; 3) balanced riders and drivers, since they have different properties.

4.2.1 *More Riders Arrive* ( $\lambda > \mu$ ). When  $\lambda > \mu$ , the summation of probabilities of states  $\{\mathbf{n}\}$ ,  $n < 0$ , can be calculated through a summation of geometric sequence, which is:

$$\sum_{i=-1}^{-\infty} p_i = p_0 \sum_{i=1}^{\infty} \left(\frac{\mu}{\lambda}\right)^i = \frac{\mu}{\lambda - \mu} p_0 \quad (7)$$

In addition, we have the fact that the summation of the probabilities  $\{p_n\}$ ,  $n \in [-\infty, \infty]$ , must be 1.

$$\sum_{i=-1}^{-\infty} p_i + p_0 + \sum_{i=1}^{\infty} p_i = 1 \quad (8)$$

Putting Equations (6) and (7) into Equation 8, we have:

$$p_0 \left( \frac{\mu}{\lambda - \mu} + 1 + \sum_{n=1}^{\infty} \prod_{i=1}^n \frac{\lambda}{\mu + \pi(i)} \right) = 1$$

Then, we have

$$p_0 = \left( \frac{\lambda}{\lambda - \mu} + \sum_{n=1}^{\infty} \prod_{i=1}^n \frac{\lambda}{\mu + \pi(i)} \right)^{-1} \quad (9)$$

Drivers are dispatched in a first come, first served order. Let  $T(n)$  be the expected idle time interval of an arrival driver  $d_j$  when the queue is in state  $\mathbf{n}$ . When there are waiting riders ( $n > 0$ ), the idle time interval of  $d_j$  is just the processing time of the platform to arrange a new rider, which can be ignored. When there is no waiting riders ( $n \leq 0$ ), the driver needs to wait until the next ( $|n| + 1$ )th rider appears in the region, which needs  $\frac{|n|+1}{\lambda}$  time on average. Finally, we can estimate the expected idle time,  $ET(\lambda, \mu)$ , of a driver after join a queue of region having the arrival rate  $\lambda$  of riders and the arrival rate  $\mu$  of drivers as follows:

$$\begin{aligned} ET(\lambda, \mu) &= \sum_{i=0}^{-\infty} \frac{|n| + 1}{\lambda} \cdot p_i + \sum_{i=1}^{\infty} 0 \cdot p_i \\ &= \frac{p_0}{\lambda} \sum_{i=0}^{\infty} (i + 1) \left(\frac{\mu}{\lambda}\right)^i \\ &= \frac{\lambda p_0}{(\lambda - \mu)^2} \end{aligned} \quad (10)$$

4.2.2 *More Drivers Rejoin* ( $\lambda < \mu$ ). We notice that when  $\lambda < \mu$ , the queue will congest more and more drivers when time elapses, which will harm the efficiency of the platform much. The platform will avoid that the rate of drivers  $\mu$  become larger than the rate of riders  $\lambda$  for each region. However, when there are indeed more drivers rejoining, we still can estimate the expected idle time.

Let  $K$  be the number of available drivers during the current scheduling time period with length  $t_c$ . Then, the queue of the region can at most congest with  $K$  drivers. Let  $\theta = \frac{\mu}{\lambda}$ . We can calculate the summation of probabilities of states  $\{\mathbf{n}\}$ ,  $-K \leq n < 0$ , with the equation as follows:

$$\sum_{i=-1}^{-K} p_i = p_0 \sum_{i=1}^K \left(\frac{\mu}{\lambda}\right)^i = \frac{\theta^{K+1} - \theta}{\theta - 1} p_0 \quad (11)$$

Then, we can update Equation 9 when  $\lambda < \mu$  as follows:

---

### Algorithm 2: Idle Ratio Oriented Greedy Algorithm

---

**Input:** A set of Regions  $A$ , current timestamp  $\bar{t}$   
**Output:** A set of rider-and-driver dispatching pairs  $I_{\bar{t}}$

- 1  $I_{\bar{t}} \leftarrow \{\emptyset\}$
- 2  $I_v \leftarrow \{\emptyset\}$
- 3 **foreach**  $a_k \in A$  **do**
- 4     retrieve a set  $I_k$  of valid rider-and-driver dispatching pairs from  $R_k$  and  $D_k$
- 5      $I_v \leftarrow I_v \cup I_k$
- 6     estimate the arrival rate  $\lambda_{(k)}$  of riders and arrival rate  $\mu_{(k)}$  of rejoined drivers in region  $a_k$  during  $[\bar{t}, \bar{t} + t_c]$
- 7 sort dispatching pairs in  $I_v$  based on their idle ratio
- 8 **while**  $I_v$  is not empty **do**
- 9     select the rider-and-driver pair  $\langle r_i, d_j \rangle$  having the smallest idle ratio from  $I_v$
- 10    add  $\langle r_i, d_j \rangle$  to  $I_{\bar{t}}$
- 11    update  $\mu_{(k)}$  of the destination region  $a_k$  of  $r_i$
- 12    remove  $\langle r_i, \cdot \rangle$  and  $\langle \cdot, d_j \rangle$  from  $I_v$
- 13 **return**  $I_{\bar{t}}$

---

$$p_0 = \left( \frac{\theta^{K+1} - 1}{\theta - 1} + \sum_{n=1}^{\infty} \prod_{i=1}^n \frac{\lambda}{\mu + \pi(i)} \right)^{-1} \quad (12)$$

In addition, when the expected number of rejoined drivers during the current scheduling time period with length  $t_c$  is  $K$  and  $\lambda < \mu$ , we can estimate the expected idle time  $ET(\lambda, \mu)$  as follows:

$$ET(\lambda, \mu) = \frac{p_0 (K + 1) \theta^{K+2} - (K + 2) \theta^{K+1} + 1}{\lambda (\theta - 1)^2}. \quad (13)$$

4.2.3 *Balanced Riders and Drivers* ( $\lambda = \mu$ ). When  $\lambda = \mu$ , we can update Equation 11 as follows:

$$\sum_{i=-1}^{-K} p_i = p_0 \sum_{i=1}^K \left(\frac{\mu}{\lambda}\right)^i = K p_0 \quad (14)$$

Then, we have

$$p_0 = \left( K + 1 + \sum_{n=1}^{\infty} \prod_{i=1}^n \frac{\lambda}{\mu + \pi(i)} \right)^{-1} \quad (15)$$

Next, we can estimate the expected idle time  $ET(\lambda, \mu)$  when  $\lambda = \mu$  as follows:

$$ET(\lambda, \mu) = p_0 \frac{(K + 1)(K + 2)}{2\lambda} \quad (16)$$

## 5 QUEUEING-BASED VEHICLE DISPATCHING ALGORITHMS

### 5.1 The Idle Ratio Oriented Greedy Approach

We first propose an *idle ratio oriented greedy* approach to solve each batch process in line 7 of Algorithm 1 with a goal to maximize the revenue summation of the platform during the current scheduling time period  $[\bar{t}, \bar{t} + t_c]$ , where  $\bar{t}$  is the current timestamp and  $t_c$  is the length of the current scheduling time window. We first define the idle ratio of driver  $d_j$  to server rider  $r_i$ , whose destination  $e_i$  is in region  $a_k$ , as follows:

$$IR(r_i, d_j) = \frac{ET(\lambda_{(k)}, \mu_{(k)})}{cost(s_i, e_i) + ET(\lambda_{(k)}, \mu_{(k)})}, \quad (17)$$

where  $ET(\lambda_{(k)}, \mu_{(k)})$  is the expected idle time of driver  $d_j$  when he/she rejoins the platform at region  $a_k$ , and  $cost(s_i, e_i)$  is the travel cost (travel time) on serving rider  $r_i$ . Recall that, in Section 2.4, we have two guiding rules for the platform to maximize its overall revenue after analyzing the MRVD problem: **a) associating higher priorities to the riders whose travel costs are higher; b) reducing the length of the idle time between serving any two consecutive riders for each driver.** We notice that when the travel cost  $cost(s_i, e_i)$  increases,  $IR(r_i, d_j)$  will decrease; when the expected idle time  $ET(\lambda_{(k)}, \mu_{(k)})$  decreases,  $IR(r_i, d_j)$  will also decrease. As a result, we only need to greedily select the rider-and-driver dispatching pairs with low idle ratios (as defined in Equation 17), then we can follow the above mentioned two guiding rules to maximize the overall revenue of the platform. Based on the observation, we propose an idle ratio oriented greedy approach as shown in Algorithm 2, which greedily selects the rider-and-driver dispatching pair having the smallest idle ratio value in each iteration.

Specifically, we first initialize the selected rider-and-driver pairs  $I_{\bar{t}}$  and the valid rider-and-driver pairs  $I_v$  with empty sets (lines 1-2). Then, for each region  $a_k$ , we put the valid rider-and-driver pairs  $I_k$  between the waiting riders  $R_k$  and available drivers  $D_k$  in the region into  $I_v$  (lines 4-5) and estimate the arrival rates,  $\lambda_{(k)}$  and  $\mu_{(k)}$ , of new riders and rejoined drivers during the current scheduling period  $[\bar{t}, \bar{t} + t_c]$  as follows:

$$\lambda_{(k)} = \begin{cases} \frac{|\hat{R}_k|}{t_c}, & |R_k| \leq |D_k| \\ \frac{|\hat{R}_k| + |R_k| - |D_k|}{t_c}, & |R_k| > |D_k| \end{cases} \quad (18)$$

$$\mu_{(k)} = \begin{cases} \frac{|\hat{D}_k| + |D_k| - |R_k|}{t_c}, & |R_k| \leq |D_k| \\ \frac{|\hat{D}_k|}{t_c}, & |R_k| > |D_k| \end{cases} \quad (19)$$

where  $|\hat{R}_k|$  and  $|\hat{D}_k|$  are the numbers of predicted riders and future rejoined drivers in region  $a_k$  during  $[\bar{t}, \bar{t} + t_c]$ . Next, after retrieving all the valid pairs, we sort them based on their idle ratios calculated with Equation 17 (line 7). Note that, the expected idle time  $ET(\lambda_{(k)}, \mu_{(k)})$  is determined by the arrival rates,  $\lambda_{(k)}$  and  $\mu_{(k)}$ , of new riders and rejoined drivers in the destination region  $a_k$ , thus we only need to estimate that for each region but not for each rider-and-driver pair individually. In each iteration of the while-loop (lines 8-12), we select the rider-and-driver pair  $\langle r_i, d_j \rangle$  having the smallest idle ratio and remove its related pairs,  $\langle r_i, \cdot \rangle$  and  $\langle \cdot, d_j \rangle$ , of rider  $r_i$  and driver  $d_j$  from  $I_v$  (since each driver only can serve one rider at one time). The selected pair  $\langle r_i, d_j \rangle$  is added in  $I_{\bar{t}}$  (line 10) and all the selected pairs  $I_{\bar{t}}$  will be finally returned (line 13).

**Complexity Analysis.** Let the number of total waiting riders be  $m$ , the number of total available drivers be  $n$  and the number of total regions be  $x$ . Assume riders and drivers be evenly distributed in  $x$  regions and  $x$  is much smaller than  $m$  and  $n$ . In lines 3-6 of algorithm 2, retrieving all the valid rider-and-driver pairs needs  $O(\frac{mn}{x})$ . To sort the valid pairs in  $I_v$  needs  $O(\frac{mn}{x} \log_2(\frac{mn}{x}))$  (line 7). In each iteration of the while-loop (lines 8 - 12 of Algorithm 2), selecting the pair having the smallest idle ratio from sorted  $I_v$  needs  $O(1)$  (lines 9-10); updating  $\mu_{(k)}$  and the idle ratio of average

---

### Algorithm 3: Local Search Algorithm

---

**Input:** A set of Regions  $A$ , current timestamp  $\bar{t}$   
**Output:** A set of rider-and-driver dispatching pairs  $I_{\bar{t}}$

- 1 Obtain a set,  $I_{\bar{t}}$ , of pairs with Algorithm 2
- 2 **do**
- 3      $FLAG \leftarrow False$
- 4     **foreach**  $\langle r_i, d_j \rangle \in I_{\bar{t}}$  **do**
- 5         **foreach**  $r'_i \in R_j$  **do**
- 6             **if**  $IR(r'_i, d_j) < IR(r_i, d_j)$  **then**
- 7                 update  $\langle r_i, d_j \rangle$  to  $\langle r'_i, d_j \rangle$
- 8                  $FLAG \leftarrow True$
- 9     **while**  $FLAG$  is  $True$
- 10 **return**  $I_{\bar{t}}$

---

$\frac{mn}{x^2}$  related pairs needs  $O(\frac{mn}{x^2})$  (line 11); removing the related valid pairs  $\langle r_i, \cdot \rangle$  and  $\langle \cdot, d_j \rangle$  from  $I_v$  needs  $O(\max(\frac{n}{x}, \frac{m}{x}))$  (line 12). Since in each iteration, at least one rider and one driver will be matched, thus there will be at most  $\min(m, n)$  iterations. Then the complexity of the while-loop is  $O(\frac{\min(m, n)mn}{x^2})$ . Thus, the complexity of Algorithm 2 is  $O(\max(\frac{mn}{x} \log_2(\frac{mn}{x}), \frac{\min(m, n)mn}{x^2}))$ . If we consider  $x$  as a constant number, and  $m$  is linearly related to  $n$ , the complexity can be considered as  $O(n^3)$ .

## 5.2 The Local Search Algorithm

In the idle ratio oriented greedy approach, we greedily select the pair having the ‘‘current’’ smallest idle ratio. However, the arrival rate  $\mu_k$  of rejoined drivers in region  $a_k$  will change after selecting some riders whose destinations are in  $a_k$ . Thus, the idle ratios of early selected rider-and-driver pairs may slightly increase in later iterations. To overcome this shortcoming in the idle ratio oriented greedy approach, we will propose a local search algorithm to improve the results, which keeps searching for rider-and-driver pairs  $\langle r'_i, d_j \rangle$  with a smaller idle ratio for driver  $d_j$  and update the assigned rider of  $d_j$  to  $r'_i$  until no such pairs can be found.

Specifically, in Algorithm 3, we first obtain a set,  $I_{\bar{t}}$ , of rider-and-driver pairs for current timestamp  $\bar{t}$  through Algorithm 2 (note that, we can also obtain  $I_{\bar{t}}$  through any other algorithms). Then, in each iteration, we check whether the rider of a pair  $\langle r_i, d_j \rangle \in I_{\bar{t}}$  can be replaced by any other valid rider  $r'_i \in R_j$  for  $d_j$ , where  $R_j$  is the valid riders to  $d_j$ . If no replacement happens, we will return the updated set,  $I_{\bar{t}}$ , of the selected rider-and-driver pairs.

We prove our local search algorithm can converge. We prove it in the below lemma.

**Lemma 5.1.** *The local search algorithm (Algorithm 3) can converge.*

**PROOF.** Assume Algorithm 3 cannot converge. Then, there is at least one driver  $d$  who keeps switching between two riders  $r_u$  and  $r_v$ . When  $d$  selects  $r_u$ , we have  $IR(r_u, d) < IR(r_v, d)$ ; otherwise, we have  $IR(r_u, d) \geq IR(r_v, d)$ . We denote the regions where  $r_u$  and  $r_v$  will end as  $a_u$  and  $a_v$ , respectively.

Since the travel costs of  $r_u$  and  $r_v$  do not change, different  $ET(\lambda_u, \mu_u)$  and  $ET(\lambda_v, \mu_v)$  lead to different  $IR(r_u, d)$  and  $IR(r_v, d)$ . Specifically, according to the definition of  $IR(r, d)$  in Equation

17,  $IR(r_u, d)$  is positively correlated with  $ET(\lambda_u, \mu_u)$  (e.g., when  $ET(\lambda_u, \mu_u)$  increases,  $IR(r_u, d)$  will also increase). When more drivers rejoin in region  $a_u$ , the expected waiting time  $ET(\lambda_u, \mu_u)$  will increase.

Let driver  $d$  select rider  $r_u$  at the beginning. If driver  $d$  switches from  $r_u$  to  $r_v$  in some iteration  $\zeta$ , there must be some more rejoined drivers switch to region  $a_u$ , which leads to  $IR(r_u, d) > IR(r_v, d)$ . Thus, there must be at least one other driver  $d'$  who switches from his/her valid rider  $r'_v$  to a new rider  $r'_u$  whose destination is also in region  $a_u$  (i.e.,  $IR(r'_u, d') < IR(r'_v, d')$ ). After  $d$  switches to  $r_v$ , the number of rejoined drivers in region  $a_u$  will decrease, and  $IR(r'_u, d')$  will also decrease. As a result,  $d'$  will not switch back to  $r'_v$ . Since no drivers will switch out from region  $a_u$ ,  $IR(r_u, d)$  will at least not decrease. As a result,  $d$  will not switch back to  $r_u$ , which is contradicted with the assumption that  $d$  keeps switching between  $r_u$  and  $r_v$ . Thus Algorithm 3 can converge.  $\square$

**Complexity Analysis.** Let the number of total waiting riders be  $m$ , the number of total available drivers be  $n$  and the number of total regions be  $x$ . Assume that riders and drivers are evenly distributed in  $x$  regions and  $x$  is much smaller than  $m$  and  $n$ . The number of total valid rider-and-driver pairs will be  $O(\frac{mn}{x})$ . The number,  $|R_j|$ , of valid riders for driver  $d_j$  will be  $O(\frac{n}{x})$ . Then each iteration of the while-loop needs  $O(\frac{mn^2}{x^2})$ . Let  $L_{max}$  be the maximum iteration numbers, then the complexity of Algorithm 3 will be  $O(L_{max} \frac{mn^2}{x^2})$ . If we consider  $x$  and  $L_{max}$  as constant numbers, and  $m$  is linearly related to  $n$ , the complexity can be considered as  $O(n^3)$ .

## 6 EXPERIMENTAL STUDY

In this section, we show the efficiency and effectiveness of our queueing-theoretic framework with different vehicle dispatching algorithms embedded through experimental studies on both synthetic and real datasets.

### 6.1 Data Sets

We use both real and synthetic data to test our framework. Specifically, for the real data set, we use the taxi trip data sets in NYC [2]. **New York Taxi Trip Dataset.** New York Taxi and Limousine Commission (TLC) Taxi Trip Data [2] is a dataset recording the information of taxi trips in New York, USA. The records are collected and provided to the NYC Taxi and Limousine Commission technology under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP [4]). Trip records can be categories as three types: yellow taxi, green taxi and FHV (For Hire Vehicle). However, due to the privacy issues, only the locations of yellow taxi can be access in the dataset long time ago. In addition, the number of FHV and green taxi records is much smaller than that of yellow taxi. Thus, we only use the taxi trip records of yellow taxis in our experiments. Each trip record includes its pick-up and drop-off taxi-zones, GPS locations and timestamps, the number of passengers and the total travel cost. In our experiment, we use taxi trip data records from January 1st, 2013 to May 20th, 2013 as training data set and May 28th, 2013 as the test data set. In the taxi records of May 28th, 2013, there is 282,255 orders. Figure 5 shows the pick-up locations of orders from 8:00 A.M. to 8:45 A.M. in New York.

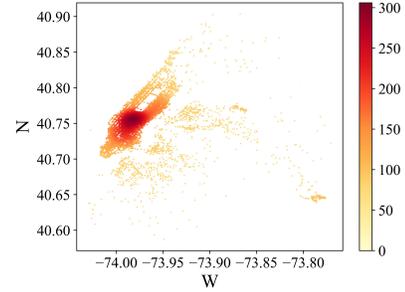


Figure 5: Distribution of Orders.

### 6.2 Experimental Configurations

For the experiments on the real data set, we use the pickup location and timestamp of a taxi trip record to initialize the source location  $s_i$  and the posting timestamp  $t_i$  of a ride order  $r_i$ . Then the dropoff location of the taxi trip record is used to set the destination  $e_i$  of the ride order. Thus, there are 282,255 riders in our experiments. For the pickup deadline  $\tau_i$  of rider  $r_i$ , we configure it by adding a uniform random noise  $\tau' \in [1, 10]$  and a base pickup waiting time  $\tau$  (configured with the setting in Table 2) to the posting timestamp  $t_i$  (e.g.,  $\tau_i = t_i + \tau' + \tau$ ). To initialize the origin locations of drivers at the beginning timestamp 0, we first randomly select a set of order records and use their pick-up locations as the origin locations of drivers. The number of drivers are configured as the parameter  $n$  in Table 2 from 1K to 5K. The whole space of New York City area (i.e.,  $-73.77^\circ \sim -74.03^\circ$ ,  $40.58^\circ \sim 40.92^\circ$ ) is evenly divide into  $16 \times 16$  grids.

In our experiment, we run the batch process every time period  $\Delta$ . To estimate the arrival rate of new riders and rejoined drivers, we look up a time window of length  $t_c$  with the “current” timestamp  $\bar{t}$  as the beginning time of the time window.

### 6.3 Approaches and Measurements

We conduct experiments to evaluate the effectiveness and efficiency of our queueing-theoretic vehicle dispatching framework with two batch processing vehicle dispatching algorithms, namely *idle ratio oriented greedy* (IRG) and local search (LS), in terms of the total revenue and the average batch running time. Note that, we set the parameter  $\alpha$  as 1, such that the total revenue is equal to the total serving time (e.g., the total travel cost of served ride orders).

Specifically, for IRG (or LS) we can further have two different combinations: IRG-P and IRG-R (or LS-P and LS-R), which use the predicted taxi demand and the real taxi demand, respectively. In addition, we also compare our approaches with three baseline methods: (1) *long trip greedy* (LTG), which greedily assigns orders with the highest revenue to available taxis; (2) *nearest trip greedy* (NEAR), which greedily assigns the nearest order to each available taxi; (3) *random* (RAND), which randomly assigns orders to available taxis. We also compare our methods with the state-of-the-art solution, POLAR [26], on car-hailing problem, which utilizes the predicted number of orders and drivers to conduct an offline bipartite matching first, then uses the offline result as a blueprint to guide the online task matching. In addition, we report the upper bound (UPPER) by summing up the revenue of the most expensive orders that can be served by idle drivers ignoring their pick-up distances in each batch. Our framework can also handle the target of maximizing the number of total served orders through modifying IRG to

**Table 2: Experimental Settings.**

Parameters	Values
the number, $n$ , of drivers	1K, 2K, 3K, <b>4K</b> , 5K
base pickup waiting time, $\tau$ (seconds)	60, <b>120</b> , 180, 240, 300
the length of batch interval, $\Delta$ (seconds)	<b>3</b> , 5, 10, 20, 30
the length of time window, $t_c$ (minutes)	<b>5</b> , 10, 15, 20, 40, 60, 80, 100

**Table 3: Results of the Estimated Idle Time**

#Drivers	MAE (s)	RMSE (%)	Real RMSE (s)
1K	2.12	5.02	8.73
2K	1.89	4.76	6.89
3K	1.78	4.53	4.43
4K	2.04	5.11	7.04
5K	2.22	5.47	11.24
6K	2.54	5.93	13.81
7K	3.20	6.45	26.39
8K	4.34	7.43	44.43

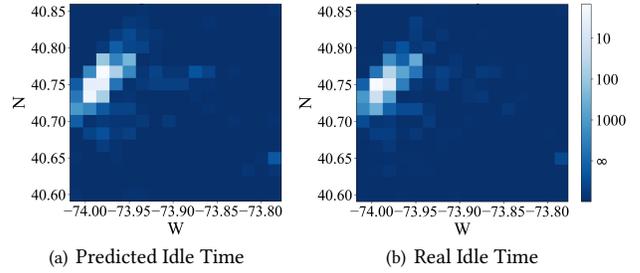
select the order with the smallest summation of its travel cost and expected idle time in each iteration. Due to space limitation, please refer to Appendix C of our technical report [9] for more details of maximizing the number of total served orders.

Table 2 shows the settings of our experiments, where the default values of parameters are in bold font. In each set of experiments, we vary one of the parameters and keep other parameters in their default values. For each experiment, we run the tested approaches on 10 different generated problem instances and report their average total revenues and average batch processing times for a whole day (from 00:00:00 to 23:59:59). All our experiments are conducted on an Intel Xeon X5675 CPU @3.07 GHZ with 32 GB RAM in Java. The code of our queueing-theoretic vehicle dispatching framework and prediction methods can be accessed in our github project [3].

#### 6.4 Results of the Estimated Idle Time

In this section, we evaluate the accuracy of our queueing theoretic model on estimating the idle time of the drivers after finishing their assigned tasks. To show the results, we vary the number of drivers from 1K to 8K and keep the other parameters in their default values as shown in Table 2. We report the mean average error (MAE), relative root mean square error (RMSE) and real root mean square error (Real RMSE) of our estimated waiting time periods of drivers compared with their real waiting time periods in Table 3.

From the results, we find that our queueing theoretic model can achieve good estimated idle time periods of the drivers after finishing their assigned tasks. When the number of drivers increases from 1K to 8K, the MAE, RMSE and real RMSE first decrease then increase. The reason is that our default batch interval is 3 seconds, when the number of drivers is 1K, the drivers can almost immediately receive new task after they finish their assigned tasks. However, due to the batch process, they need to wait until next batch process, which in fact leads to the major difference between the estimated waiting time periods and the real ones. When the number of drivers increases from 1K to 4K, more and more drivers needs to wait for a while to receive a new task after they finish their last tasks. Then the estimation errors caused by the batch processes become tiny. When the number of drivers continues increasing from 4K to 8K, the idle time of drivers also increases obviously. The

**Figure 6: Comparison of Predicted and Real Idle Time.**

MAE and real RMSE of the results of our queueing theoretic model also increases obviously, however the relative RMSE only increases 2.32%, which shows that our estimation model is accurate. Figure 6(a) shows the predicted idle times for each regions achieved by our queueing theoretic model, which is very close to the real idle times of drivers (shown in Figure 6(b)) during the running of our vehicle dispatching framework.

#### 6.5 Effects of the Prediction Methods

In this section, we evaluate the effects of prediction methods for three prediction related approaches, POLAR, IRG and LS. Table 4 shows the achieved total revenue of three prediction related approaches with default parameters (in Table 2) by using different prediction methods (introduced in Section 3.1.1). From the results, we can find that: a) the more accurate the prediction method is, the higher total revenue that each approach can achieve; b) LS is the best approach on utilizing the prediction information to improve the total revenue.

**Table 4: Results of Effects of Prediction Methods ( $10^8$ )**

	HA	LR	GBRT	DeepST	Real
IRG	2.2460	2.3203	2.3446	2.3756	2.3899
LS	2.2921	2.3725	2.4267	2.4625	2.4727
POLAR	2.0460	2.2293	2.2767	2.2953	2.3285

#### 6.6 Experimental Results of Vehicle Dispatching Approaches

In this section, we show the effects of the number,  $n$ , of drivers, the base pickup waiting time  $\tau$ , the length,  $\Delta$ , of batch interval, and the length,  $t_c$ , of time window to estimate the arrival rates of riders and rejoined drivers.

**Effect of the Number,  $n$ , of Drivers.** Figure 7 illustrates the experimental results on varying the number of drivers from 1K to 5K, where other parameters are in their default values. In Figure 7(a), when the number of drivers increases from 1K to 5K, all the tested approaches can achieve results with increasing total revenue. The reason is that when more drivers are available, more riders can be served before their pickup deadlines. When the number of drivers is 1K, our IRG and LS approaches can achieve higher total revenue than RAND, LTG, NEAR and POLAR. The difference between the results of our IRG and LS are small. When the number of drivers increases, the advantage of our IRG and LS in terms of the total revenue become narrow. We also notice that when the number of

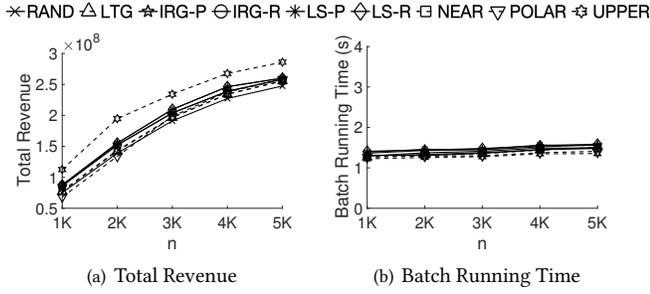


Figure 7: Effects of Number of Drivers  $n$ .

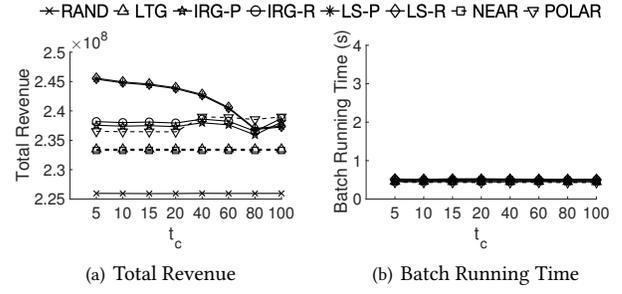


Figure 9: Effects of Time Window  $t_c$ .

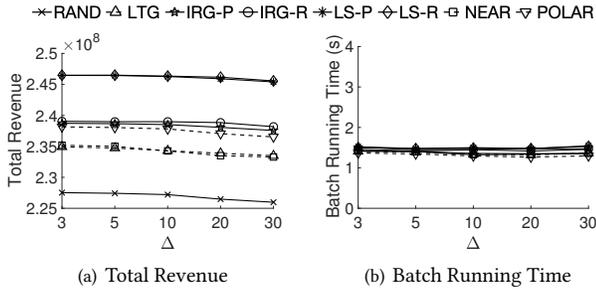


Figure 8: Effects of Batch Length  $\Delta$ .

drivers reaches 5K, all the tested approaches can achieve results with total revenue close to the upper bound. The reason is that when there are 5K drivers, almost all the riders can be served as long as he/she joins the platform. Our LS can achieve from 78.1% to 92.0% of the upper bound revenue when the number of drivers increases from 1K to 5K. To clearly show the differences between the total revenues of our tested approaches, we will not plot out the results of UPPER as they are always same with the results in Figure 7(a). In Figure 7(b), when the number of drivers increases, the batch running time of all the tested approaches also increases slightly, which is because in each batch there are more drivers requiring more time to process. We can see that all the tested approaches can finish each batch processing within 2 seconds, which is unnoticeable to the users and acceptable for the batch processes with 3-second intervals.

**Effect of the Length,  $\Delta$ , of Batch Interval.** Figure 8 shows the experimental results on varying the length,  $\Delta$ , of the batch interval from 3 to 30 seconds, while other parameters are set to their default values. As shown in Figure 8(a), when the length,  $\Delta$ , of batch interval increases from 3 to 30 seconds, the total revenues of the results achieved by the tested approaches decrease slightly. The reason is that when the length of the batch interval increases, more riders may be missed before their pickup deadlines within two consecutive batches. In other words, when  $\Delta$  increases, the probability of a rider becomes time out will increase during the batch intervals, when the platform does not respond to any riders or drivers. Another reason is that when drivers become available, they also need to wait for the next batch to be assigned with new riders, which also leads to the bad effect on the total revenue. Thus, in real applications,  $\Delta$  should not be too large. In addition, we notice that our IRG-P and LS-P can achieve higher total revenues than RAND, LTG, NEAR and POLAR.

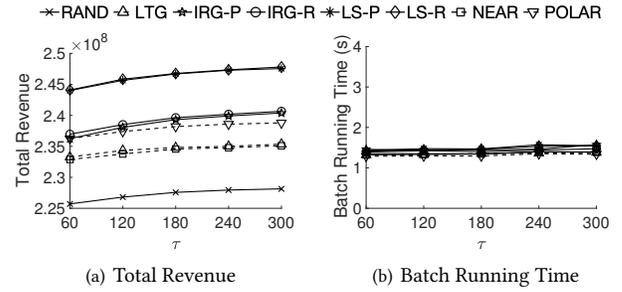


Figure 10: Effects of Base Waiting Time  $\tau$ .

We find that when we use the ground truth of the taxi demand for our IRG-R and LS-R algorithms, they can achieve higher total revenues than IRG-P and LS-P, which shows the importance of the accuracy of the taxi demand methods. In other words, for the real applications, a more accurate prediction model can bring increases on the total revenue. In Figure 8(b), the batch running time of the tested approaches slightly increases, since the number of riders and drivers for each batch will increase when  $\Delta$  increases.

**Effect of the Length,  $t_c$ , of Time Window.** Figure 9 presents the experimental results on varying the length  $t_c$  of time window on estimating the arrival rate of new riders and rejoined drivers. In Figure 9(a), the total revenue achieved by IRG and LS will decrease when  $t_c$  becomes larger than 20 minutes. The reason is that most taxi trips in NYC taxi trip dataset have a travel time of less than 20 minutes [12]. The effect of future rejoined drivers in more than 20 minutes later is almost neglectable for our IRG and LS algorithms. However, when  $t_c$  becomes 40 minutes, POLAR can perform better than itself in the experiment of  $t_c$  smaller than 20 minutes. Thus, in real platform,  $t_c$  should not be too large. Since RAND and LTG do not consider the demand and supply of the taxis in future, the length,  $t_c$ , of time window has no effect on them. In Figure 9(b),  $t_c$  has no clear effect on the running time of our tested approaches.

**Effect of the Base Waiting Time  $\tau$ .** Figure 10 illustrates the effect of the waiting time  $\tau$  of riders by varying  $\tau$  from 60 to 300 seconds while keeping other parameters in their default values. In Figure 10(a), when the waiting time of riders  $\tau$  increases, the total revenue of the results achieved by our tested approaches also increases. The reason is that when riders can wait for longer time, the probability that they can be served by some drivers will increase, which is consistent with human intuition. With the help of ground truth of the taxi demand (more accurate than our predicted demand), LS-R

can achieve slightly higher total revenue than LS-P. IRG, LS and their variants can all surpass RAND, LTG, NEAR and POLAR. In Figure 10(b), the batch running time of tested approaches increases slightly when the waiting time of riders increases. The reason is that when riders can wait for longer time and the number of drivers does not change, the number of riders in each batch will also increase, which leads to the processing time of each batch becomes longer.

In summary, LS and IRG can perform better than RAND, LTG, NEAR and POLAR in terms of total revenue. Our proposed algorithms are more effective when the number of drivers is smaller (e.g., 1K drivers our in experiments). The accuracy of taxi demand prediction method can affect the final results on the total revenue. Thus, taxi demand prediction models with higher accuracy are more valuable for the platform. Our framework is efficient. In all the experiments, the running time of each batch for all the tested approaches is less than 2 seconds, which is affordable for the platform to perform a batch process with 3 seconds for each batch interval.

## 7 RELATED WORK

Recently, online car-hailing platforms develops rapidly, which has drawn attention from academia and industry.

Our MRVD problem is related to task assignment in spatial crowdsourcing [10, 11, 20, 26], which assign a set of workers to the locations of tasks to conduct subject to various constraints and optimization goals. However, in our MRVD problem, each order has a pickup location and a destination, while each task in spatial crowdsourcing usually has only one required location. In [20], based on the publishing models, the authors classified the spatial crowdsourcing in two modes: worker selected task (WST) mode [17] and server assigned tasks (SAT) mode [10, 11, 26]. In WST mode, workers select tasks by themselves. In SAT mode, the server/system has the control on assign tasks to workers base on its objectives. In SAT mode, there are two processing styles: online task assignment mode [26] and batch-based task assignment mode [10, 11, 20]. Recently, researchers start to utilize the prediction models to predict the future distributions of workers and tasks to improve the overall performance in a relatively long time period (e.g., 1 day). For instance, researchers build an offline blueprint based on the predicted distributions of workers and tasks, then use it to guide the online task assignment to maximize the total number of assigned tasks [26]. Our MRVD targets on maximizing the total revenue of the platform, which cannot apply existing solutions directly. Thus, we develop our queueing theoretic framework, which uses queueing theory to estimate the idle time of drivers based on the predicted number of orders and drivers in each region.

Our MRVD problem is also related to dial-a-ride problem (DARP), which assume a fleet of vehicles located at a common depot, and schedules should be made to accommodate  $m$  rider requests based on their pick-up and drop-off time constraint. Existing works on DARP have mainly focused on static offline DARP, where the constraints are known beforehand. The general DARP is NP-hard and intractable, unless its scale is not big (e.g., hundreds of vehicles and riders) [14]. [15] uses a heuristic method called tabu search to find the neighbourhood solution from current solution, to avoid finding cycle result and local optimum, they forbid the recent visited answers and use some diversification mechanism.

With the emergence of ridesharing business, many riders prefer choosing the ridesharing service, as it is cheaper than non-share car request with limited time delay. The authors [12] designs a algorithm to dispatch the similar rider to the same car with a goal of maximizing the total utility, which includes the rider related utility, vehicle-related utility and trajectory-related utility. In [31], the authors propose a packing-based approach, which first packs the riders together then assigns groups of riders to vehicles. To solve the scheduling problem for a vehicle with a set of assigned riders, authors in [27] propose a linear time complex method. However, ridesharing mainly focuses on scheduling and solving conflicts of route-sharable riders to vehicles, which is different from MRVD.

In addition, traffic prediction is also a critical technology in urban city transportation scenario. With accurate prediction, we can foresee the future and make plan to fulfill the long time revenue. There are many models which focusing on predicting the number of orders in the next time slot by integrating temporal and spatial information. [29] proposes a Deep ST model which combines the geographical and historical traffic data together, to decrease the difference between estimated traffic flow number and actual count. With the powerful deep convolutional neural network and rich daily meta data (e.g., holiday and weather), they get the state of art prediction results.

## 8 CONCLUSION

In this paper, we study the problem of maximum revenue vehicle dispatching problem (MRVD), in which rider requests dynamically arrive and drivers need to serve as many riders as possible such that the entire revenue of the platform is maximized. We prove that the MRVD problem is NP-hard and intractable. Through analyses, we find to maximize the total revenue, we need to give higher priorities to ride orders with long travel cost and less idle time. We propose a queueing-theoretic framework, which predicts the taxi demand (rider orders) offline and schedule the drivers to regions where the idle time of them will be small. Our framework dispatching drivers to riders in a batch-based processing for every  $\Delta$  seconds. To handle the batch vehicle dispatching problem, we propose two heuristic approaches, namely idle ratio oriented greedy (IRG) and local search (LS). Through experiments on the real and synthetic data sets, we show the effectiveness and efficiency of our queueing-theoretic vehicle dispatching framework.

## ACKNOWLEDGMENTS

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants. Peng Cheng's work is sponsored by Shanghai Pujiang Program 19PJ1403300. Xuemin Lin's work is supported by ARC DP200101338. Libin Zheng's work is supported by the Fundamental Research Funds for the Central Universities, Sun Yat-sen University.

## REFERENCES

- [1] 2021. [Online] DiDi Chuxing. <https://www.didichuxing.com>.
- [2] 2021. [Online] NYC Taxi & Limousine Commission Trip Record Data. [http://www.nyc.gov/html/tlc/html/about/trip\\_record\\_data.shtml](http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml).
- [3] 2021. [Online] Source Code of Queueing-Theoretic Vehicle Dispatching Framework. <https://github.com/inabao/queue-based-order-dispatching>.
- [4] 2021. [Online] Taxicab Passenger Enhancements Project. [http://www.nyc.gov/html/tlc/html/industry/taxicab\\_serv\\_enh.shtml](http://www.nyc.gov/html/tlc/html/industry/taxicab_serv_enh.shtml).
- [5] 2021. [Online] Uber. <https://www.uber.com>.
- [6] Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. 2016. Dynamic pricing in ridesharing platforms. *ACM SIGecom Exchanges* 15, 1 (2016), 65–70.
- [7] Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. 2004. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. 166–174.
- [8] Peng Cheng, Chao Feng, Lei Chen, and Zheng Wang. 2019. A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1622–1625.
- [9] Peng Cheng, Jiabao Jin, Lei Chen, Xuemin Lin, and Libin Zheng. 2021. A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing [technical report]. *arXiv preprint arXiv:2107.08662* (2021).
- [10] Peng Cheng, Xiang Lian, Lei Chen, Jinsong Han, and Jizhong Zhao. 2016. Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2201–2215.
- [11] Peng Cheng, Xiang Lian, Zhao Chen, Rui Fu, Lei Chen, Jinsong Han, and Jizhong Zhao. 2015. Reliable diversity-based spatial crowdsourcing by moving workers. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1022–1033.
- [12] Peng Cheng, Hao Xin, and Lei Chen. 2017. Utility-aware ridesharing on road networks. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1197–1210.
- [13] Jing Chu, Kun Qian, Xu Wang, Lina Yao, Fu Xiao, Jianbo Li, Xin Miao, and Zheng Yang. 2018. Passenger Demand Prediction with Cellular Footprints. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.
- [14] Jean-François Cordeau. 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54, 3 (2006), 573–586.
- [15] Jean-François Cordeau and Gilbert Laporte. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37, 6 (2003), 579–594.
- [16] Noel Cressie and Christopher K Wikle. 2015. *Statistics for spatio-temporal data*. John Wiley & Sons.
- [17] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. 2013. Maximizing the number of worker’s self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems*. 324–333.
- [18] Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.
- [19] Priscilla E Greenwood and Michael S Nikulin. 1996. *A guide to chi-squared testing*. Vol. 280. John Wiley & Sons.
- [20] Leyla Kazemi and Cyrus Shahabi. 2012. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th international conference on advances in geographic information systems*. ACM, 189–198.
- [21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
- [22] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 33.
- [23] Kiam Tian Seow, Nam Hai Dang, and Der-Hong Lee. 2010. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation Science and Engineering* 7, 3 (2010), 607–616.
- [24] John F Shortle, James M Thompson, Donald Gross, and Carl M Harris. 2018. *Fundamentals of queueing theory*. Vol. 399. John Wiley & Sons.
- [25] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. 2016. Online minimum matching in real-time spatial data: experiments and analysis. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1053–1064.
- [26] Yongxin Tong, Libin Wang, Zhou Zimu, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. 2017. Flexible online task assignment in real-time spatial data. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1334–1345.
- [27] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1633–1646.
- [28] Xingang Wen, Yinfeng Xu, and Huili Zhang. 2012. Online Traveling Salesman Problem with Deadline and Advanced Information. *Comput. Ind. Eng.* 63, 4 (Dec. 2012), 1048–1053.
- [29] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. (2017), 1655–1661.
- [30] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 92.
- [31] Libin Zheng, Lei Chen, and Jieping Ye. 2018. Order dispatch in price-aware ridesharing. *Proceedings of the VLDB Endowment* 11, 8 (2018), 853–865.