Ridesharing as a Service: Exploring Crowdsourced Connected Vehicle Information for Intelligent Package Delivery

Fangxin Wang†, Yifei Zhu†, Feng Wang‡, Jiangchuan Liu†*
School of Computing Science, Simon Fraser University, Canada†
Department of Computer and Information Science, the University of Mississippi, USA‡

***Abstract*—Nowadays online shopping has become explosively popular and the vast numbers of generated packages have brought great challenges to the traditional logistics industry, especially the last mile package delivery. Traditional delivery approaches rely on dedicated couriers for package dispatch, while the labor cost is quite expensive and the quality is hard to guarantee due to the diverse delivery addresses and tight deadlines. On the other hand, modern cities are full of available transportation resources such as private car trips. The mobile crowdsourcing through 4G/5G and vehicle-related communications enables the vehicle resources to be connected as an intelligent transportation system. As such, we believe ridesharing will be a core service for connected vehicles, which we refer to as Ridesharing as a Service (RaaS).**

In this paper, we focus on the quality of service (QoS) of RaaS in the last mile package delivery. Mining from real-world car trips, we build up a citywide routing graph and conduct a personalized travel cost prediction considering both the travel time of each driver and the fuel consumption of each vehicle. We then design an online algorithm to assign proper package delivery tasks to the submitted car trips, aiming to maximize the utility of the ridesharing service provider. Our extensive real-world trace-driven evaluations further demonstrate the superiority of our RaaS based package delivery.

I. Introduction

The explosive growth of online shopping brings tremendous opportunities as well as heavy burdens to traditional logistics industry, where a mass of packages are to be delivered every day. Among the entire logistics chain, the cost of *last mile* delivery is the most expensive part, ranging from 13% to even 75% of the entire delivery cost [1]. In the traditional approach, the diverse delivery addresses rely on dedicated couriers where large-scale package consolidation via trains or planes no longer applies, leading to expensive labor cost. The packages generated in China's 2017 online Singles' Day are estimated to achieve 1.5 billion and have to be delivered in a few days [2]. Such a massive package delivery demand obviously surpasses the last mile logistical capability, delaying the eventual package delivery time for days or even weeks [3]. Even though, a worldwide survey from McKinsey reveals that the majority of consumers still prefer the home delivery service, especially the same day delivery [4].

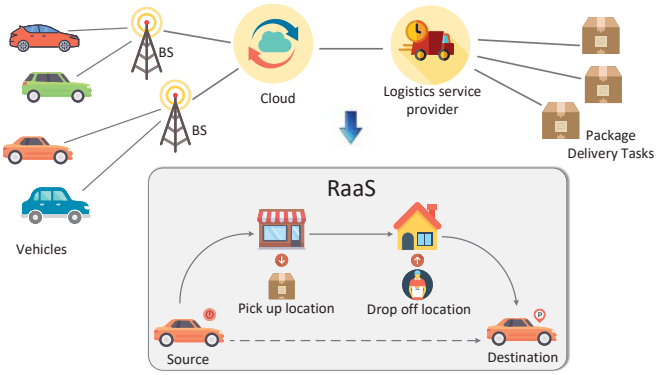

Fig. 1. The architecture of RaaS based last mile package delivery. A driver is planning to travel from the source to the destination through the dashed route. Assume that a package is scheduled to be delivered from a shop to a house, where the shop is close to the source and the house is close to the destination. Then the driver can deliver this package incidentally during the trip through the alternate route shown by solid lines.

The contradiction between the limited delivery capacity and the ever-increasing package delivery demand is becoming increasingly serious, which degrades the quality of service (QoS) in package delivery. It is no doubt costly to improve the logistics capacity to catch the delivery demand. Besides, the sudden burst of demand during large online shopping holidays (e.g., Black Friday in US and Singles' Day in China) further greatly challenges the delivery services, leading to escalating delivery delay. And it is also not cost-efficient to overprovision the delivery capacity to meet such transient demand.

In fact, the hidden delivery capability within an urban environment is enormous – modern cities never lack transportation resources, only if we can utilize them effectively. With the mobile crowdsourcing through 4G/5G [5] and vehicle-related communications [6] (e.g., vehicle-to-vehicle and vehicle-to-infrastructure), the vehicle information can be readily accessed anytime and anywhere, which enables the citywide vehicles to be connected as an intelligent transportation system. This opens an opportunity toward a new generation of last mile delivery that explores the crowd intelligence of connected vehicles. In particular, the citywide car trips are a rich set

This work is supported by a Canada Technology Demonstration Program (TDP) grant, a Canada NSERC Discovery Grant, and an NSERC E.W.R. Steacie Memorial Fellowship. This work is also partly supported by an NSF I/UCRC Grant (1539990).
*Corresponding author: Jiangchuan Liu(jcliu@sfu.ca).

978-1-5386-2542-2/18/$31.00

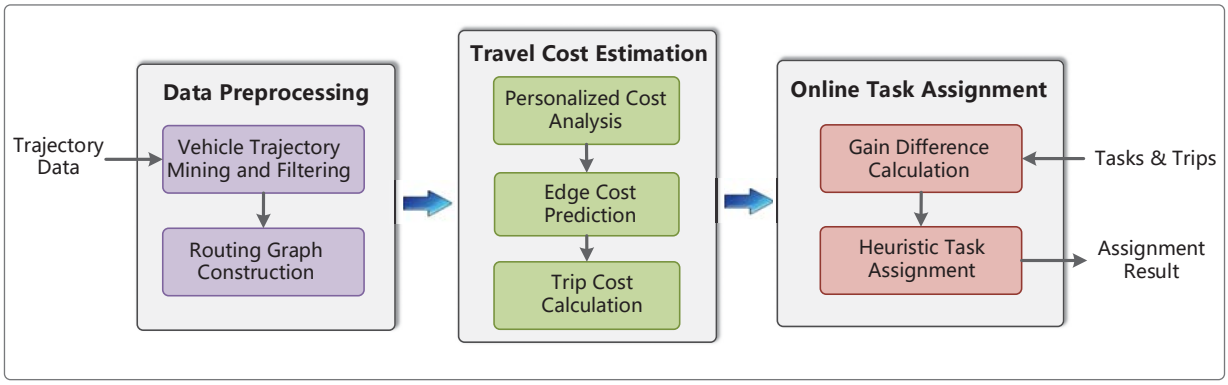

Fig. 2. The framework of our RaaS based package delivery system.

of resources to share, with packages being hitchhiked. A driver can take the delivery task released by the platform (or service provider) and finish the delivery during his/her trip with a reward. The crowdsourced car trip sharing makes the most effective use of transportation resources, filling the gap between the insufficient logistics provision and massive delivery demand. As such, we believe ridesharing will be a core service in the intelligent package transportation, which we refer to as *Ridesharing as a Service* (RaaS), where the architecture is illustrated in Fig. 1.

There are several main challenges to realize RaaS based package delivery. First, each package should be delivered by its arrival deadline. Yet it is hard to predict the accurate package arrival time considering the uncertain traffic conditions and personalized driving styles. Second, taking a delivery task will impose extra costs (e.g., extra time cost and fuel consumption) to a driver due to the route change. A prerequisite is that the incentive reward of a package must be higher than his/her expectation. Given that many car trips can be suitable to take a task, how to select the best choice and set a reasonable price remain a problem. Third, such a schedule system requires a fast online assignment process given the delivery tasks and car trips are coming in real time.

In this paper, we focus on the QoS of RaaS and explore crowdsourced car trip information for intelligent package delivery. Mining from the real-world car trips, we first build up the citywide routing graph and learn the travel cost (travel time and fuel consumption) features of each road segment. We conduct a personalized analysis to achieve accurate travel cost prediction considering different time period, driving styles and vehicle fuel efficiencies. We then design an online algorithm to assign proper tasks to the submitted car trips, aiming to maximize the utility of the platform. Our real-world trace-driven experiments demonstrate that our personalized travel cost prediction reduces the error rate by up to 60% compared to the approach without such consideration. And our online assignment algorithm can achieve an average of 52% improvement on the utility of the platform compared to the best baseline method.

The rest of the paper is organized as follows. Section II presents the overview and framework design of our RaaS based delivery system. In section III, we describe the triple-dependent analysis on edge cost prediction and the personalized trip cost calculation. In section IV, we introduce our online task assignment algorithm. We show the trace-driven experiment results in section V. We introduce the related work in section VI and conclude this work in section VIII.

II. Overview and Framework

In the RaaS based package delivery, people can submit their trips online to the platform. The platform will determine how to assign package delivery tasks to the best candidate car trips. We first introduce several basic definitions as follows. A *car trip* $\mathcal{T}$ denotes a trip plan from one location to another, which is defined as $< \mathcal{T}.s, \mathcal{T}.e, \mathcal{T}.t, \mathcal{T}.dr, \mathcal{T}.ve >$, where $\mathcal{T}.s$ is the trip start location, $\mathcal{T}.e$ is the trip end location, $\mathcal{T}.t$ is the trip start time, $\mathcal{T}.dr$ is the driver identification and $\mathcal{T}.ve$ is the vehicle type of this trip. The *trip cost* includes two components, i.e., the time cost and the fuel cost. Given a particular trip, the trip cost is represented as $\mathcal{C}(\mathcal{T}) : (\mathcal{C}_T(\mathcal{T}), \mathcal{C}_F(\mathcal{T}))$, where $\mathcal{C}_T$ and $\mathcal{C}_F$ denote the travel time cost and fuel consumption cost, respectively. A *package delivery task* $\mathcal{P}$ includes five properties: a pick up location $\mathcal{P}.p$, a drop off location $\mathcal{P}.d$, a task submission time $\mathcal{P}.t$, a required arrival deadline $\mathcal{P}.ddl$, and a value $\mathcal{P}.v$. The value of a task can be considered as the cost of delivering $\mathcal{P}$ through dedicated couriers.

When deciding whether to assign a set of tasks to a trip, the platform has two main considerations. First, the assigned packages must be delivered before their required arrival deadlines. Yet the ever-changing road conditions and drivers' personalized driving styles make the package delivery time prediction even challenging. Second, since different trips can have different extra costs when taking the same package delivery tasks, the corresponding expected rewards can vary with each trip, leading to different gains. The platform wants to assign tasks to the most suitable car trips to maximize the total utility.

In this paper, we explore crowdsourced car trip information and propose a RaaS based package delivery system to solve

this problem. The framework of our system consists of three components, including *Data Preprocessing*, *Travel Cost Prediction*, and *Online Task Assignment*, as illustrated in Fig. 2.

Data Preprocessing. We first build up the routing graph based on the collected massive trajectory data. We conduct a vehicle trajectory data mining and filter out those trajectories out of the target location coverage. The qualified time-stamped GPS points are then mapped onto the road network. Since the GPS records of each trajectory can be skewed, we first calibrate the GPS locations of each trajectory using Google SnapToRoads API[1] and get the unified GPS samples for each trajectory. When two trajectories intersect, we identify the point of intersection as a landmark, indicating a vertex in our graph. Given the massive trajectories cover all the citywide main roads, we therefore construct a routing graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where each edge $\mathbb{E}$ is a road and each vertex $\mathbb{V}$ is an intersection of at least two roads. With the timestamp and fuel consumption records, we are able to calculate the travel time and fuel cost of each trajectory on each edge in the graph[2].

Travel Cost Prediction. The main objective of this component is to achieve an accurate delivery time and travel cost prediction, which serves for the delivery task assignment. We conduct a personalized prediction based on the massive trajectory data. Different from existing works [7] that only considered time-varying traffic conditions, we comprehensively analyze the impact of different time, different driving styles and different vehicle types on the travel time and travel cost. We predict the expected cost of each edge employing a Gaussian mixture model (GMM) based algorithm. We further construct a dynamic weighted graph to calculate the trip cost for any particular car trip.

Online Task Assignment. We first study the online assignment from its offline scenario and formulate the task assignment as an optimization problem. We then conduct a comprehensive analysis on the reward design mechanism. We propose a heuristic task selection algorithm based on the extra cost of different trips and the threshold gain according to the waiting ratio of each task. For each coming trip, we repeatedly select one task with the maximal gain difference and allocate it to the trip until no suitable tasks can be allocated.

III. Travel Cost Prediction

A. Time Slot Slicing

The traffic conditions are usually changing quickly in different time periods, and can cause a large variance in time and fuel cost. A typical example is that traffics are more crowded in rush hours than during other time, leading to more fuel consumption and longer travel time even on the same road.

Some previous works [7], [8] only divided the time of a day into several time periods and predicted features in different periods. However, this coarse-grained partition is not accurate enough to describe the fast-changing travel cost. We therefore use a fine-grained two-level time slot slicing mechanism to capture the time-dependent features of travel time cost and fuel cost on each edge. We first conduct a *day level* partition that considers the traffic difference from a view of days and weeks. A week is divided into *work days* and *rest days*, where both holidays and weekends are included in the rest days. The work days include three time periods: rush hours W_r, daytime hours W_d and nighttime hours W_n; and the rest days include two time periods: daytime R_d and nighttime R_n. Thus, we have five time period types. The second level is *minute level* partition that divides a day into $L = [\frac{24*60}{\alpha}]$ time slots, where α indicates the time slot duration between two consecutive slots, e.g., 15 minutes. Fig. 3 illustrates the time periods of work days and rest days, respectively.

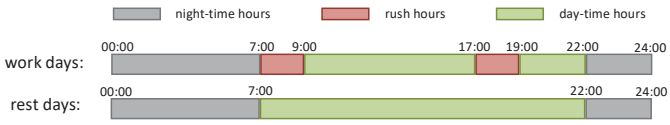

Fig. 3. The distribution of time periods.

For a time slot l and an edge e_k, we filter out the car trips whose trajectory passes through this edge during this particular time slot as $S^l_{e_k}$. Considering the periodic feature of traffic flow, we extend one time slot in a particular day to an extended time slot set of a *time cycle* (e.g., a week or a month) with the same day level type (i.e., work days or rest days). For example, we consider time slot l of work days on edge e_k and the time cycle is a week. Then $S^l_{e_k}$ includes $S^l_{e_k}(Mon) + S^l_{e_k}(Tue) + ... + S^l_{e_k}(Fri)$, not including the rest days. In the rest of this paper, we refer to a time slot as the extended time slot set unless otherwise specified. Based on $S^l_{e_k}$, we can obtain the cost-count set C^l_{T,e_k} and C^l_{F,e_k} as $\{(cost, count)\}$, where *cost* means the travel time or fuel consumption, and *count* represents how many such trips are observed.

Although the time slot slicing to some extent demonstrates the general traffic characteristics in different time periods, it is far from enough to achieve an accurate cost prediction for every trip. We next conduct a personalized edge cost calibration considering the driving style of each driver and the fuel efficiency of each vehicle.

B. Analysis of Different Driving Styles

People with different driving styles can have different time cost for an edge, even departure at the same time. For example, skilled drivers may often change lanes and overtake other cars to avoid slow traffic, and thus passing through a road faster. On the other hand, conservative drivers (especially new drivers) are usually more cautious. They usually follow the speed limit strictly and will not overtake others, leading to a longer travel time on the same road. This difference can cause a large variance in C^l_{T,e_k}. Therefore, the actual benchmark travel cost of each edge is quite dependent on drivers' driving styles and we need to consider it for personalized cost prediction.

[1]https://developers.google.com/maps/documentation/roads/snap

[2]For ease of computing, we assume that a vehicle only starts or ends at a landmark, which only introduces marginal errors.

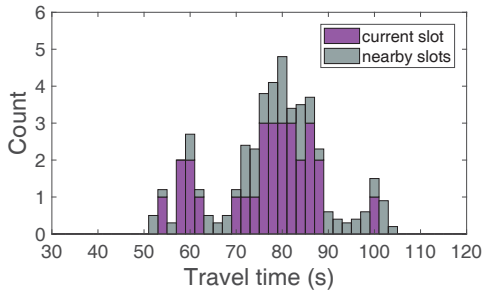

Fig. 4. The cost overlay mechanism for a particular time slot based on the travel time cost of a road.

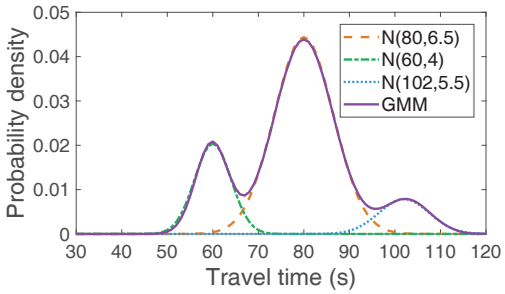

Fig. 5. An example of using three-component GMM to predict the travel time distribution.

To this end, we identify different drivers for each trip and calibrate the costs according to different driving styles. We assume that each car in our dataset is used by the same driver and the driver keeps his/her driving style constant. Then we define the *driving style index* DS_u related to a particular driver u as the mean ratio between his/her travel time through every road and all drivers' average travel time through the same road.

We calculate DS for each driver by the following steps. For a particular driver u, we first extract all the edges he/she has driven in all time slots. Given that each car has a unique ID and a car is usually used by its owner, we can easily extract all the car trips as well as the related edges belonging to a driver. For a particular C^l_{T,e_k}, we denote DS^l_{u,e_k} as the ratio of the travel time cost of driver u to the average travel time cost of all drivers at edge e_k during time slot l. By traversing all our dataset, DS_u can be easily calculated as the average value of all DS^l_{u,e_k}. Repeating this process, we can obtain the driving style index for every driver. At last, we calculate the benchmark travel time cost for an edge at a given time period through dividing each cost in C^l_{T,e_k} by the corresponding DS_u.

C. Analysis of Different Fuel Efficiencies

Similarly, different vehicle types also have different fuel efficiency (represented by fuel consumption of per 100 kilometers in city way), which can cause diverse fuel consumption collection in C^l_{F,e_k}. For example, a van can consume much more fuel than a household sedan. We thus need a unified indicator to evaluate the benchmark fuel cost of an edge.

We first select a baseline vehicle type v_0 and specify the fuel efficiency of this vehicle type f_{v_0} as the baseline fuel efficiency. Note that the fuel efficiency data can be obtained from the official vehicle manuals with the detailed car types from our dataset. Similarly, we define a *fuel efficiency index* FE of vehicle v as $FE_v = \frac{f_v}{f_{v_0}}$, indicating the fuel efficiency ratio of car type v to the baseline car type v_0.

With this index, we then calibrate the benchmark fuel consumption cost for each edge through dividing each cost in C^l_{F,e_k} by the corresponding fuel efficiency index FE_v. These calibrated metrics further enable us to learn the actual characteristics of travel cost, and achieve an accurate personalized prediction.

D. Edge Costs Prediction

We assume that the costs of each time slot follows some random distributions and estimate random variables RV^l_{T,e_k} and RV^l_{F,e_k} for each cost. C^l_{T,e_k} and C^l_{F,e_k} can be simply used for such estimation. Yet such a prediction can be inaccurate and not robust for two reasons. First, as the road traffic usually changes gradually with time, this hard time slot slicing can cut off the continuity of time and cause inaccurate feature extraction. Second, due to the fine-grained time slot granularity, the count of trajectories in each time slot can be limited, or even empty, e.g., it is likely that no car passes through a road in midnight. Thus using such hard time slot slicing may fail to estimate every road in every time slot (because some road may lack record data in some time slots.)

Considering the continuity of time periods, we use a *cost overlay* mechanism to address this issue. When estimating the random variable in time slot l, we not only consider the costs in l (i.e., C^l_{T,e_k} and C^l_{F,e_k}), but also include the costs of other nearby time slots. That is, the costs of nearby time slots are multiplied by a weighted factor and then added to the costs in l. Intuitively, time slot that is more further away should have a smaller weight. We thus employ the widely-used exponential decaying function $Decay(t) = e^{-\frac{t}{\lambda}}$, where λ indicates the mean lifetime. The contribution of time slot x regarding the target time slot l is described as $Decay(x,l) = e^{-\frac{1}{\lambda(x,l)} \cdot min\{|x-l|,|x+D-l|,|x-D-l|\}}$, where D is the number of time slots in a day and equals to $[\frac{24*60}{\alpha}]$. We set $\lambda(x,l)$ as different values according to whether x and l have the same time period type, different from [9] using a constant mean lifetime parameter. If they are in the same time slot, we set a larger value τ_{la}; otherwise we set a smaller value τ_{sm}. In this way, the final C^l_{T,e_k} and C^l_{F,e_k} can be calculated as $\{(cost, \sum_x count(x) \cdot Decay(x,l))\}$. Fig. 4 illustrates a simple case of using cost overlay mechanism for a particular time slot based on the travel time cost of an edge. This mechanism can complement the defect of the sparse data and makes the characteristics of cost clearer.

With the benchmark travel time and fuel consumption cost for each time slot on every road, we next consider learning corresponding random variables and their probability functions. Given the possibly diverse data distribution, we need a model that can generally fit all kinds of distributions well.

As such, we consider using Gaussian mixture model (GMM) to estimate these random variables. As a linear combination of multiple Gaussian distributions, GMM is able to approximate any probability distribution [10], which fits well in our context. Fig. 5 shows an example of using three-component GMM to estimate the travel time distribution. We can observe that GMM is able to match the distribution curve precisely.

At last, we estimate the expectation of random variables RV^l_{T,e_k} and RV^l_{F,e_k} as the final benchmark travel cost (including travel time cost $E(RV^l_{T,e_k})$ and fuel consumption cost $E(RV^l_{F,e_k})$) for time slot l on edge e_k. Considering the features of travel cost can change with time (e.g., the travel cost of an edge may greatly increase due to the road maintenance), we need to update the travel cost timely. Recall that we maintain a *time cycle* (a week or a month) when calculating the costs of each time slots. We update the predicted travel cost as well as the cost-count set by incorporating a new day in the cycle and removing the last day from the cycle every time.

E. Personalized Trip Costs Calculation

With the predicted edge cost, we next consider how to select the optimal routing path for a particular trip and achieve an accurate trip cost prediction. Different from the static shortest path problem, the cost of every path is uncertain since the edge cost in the routing graph is changing with different time slots, different drivers and different vehicles. Thus this problem is actually finding the shortest path in a dynamic weighted graph. We next try to find the optimal path to minimize the travel time for a given trip.

The problem actually includes two situations. In the first situation, a travel trip is very short and the entire time period falls in one time slot. As the routing graph is static in this situation, we can simply use the traditional shortest path finding algorithm such as Dijkstra algorithm [11] to find the travel path with the lowest travel time cost. Second, a trip can travel through several time slots, where the weight of each edge for each time slot is different in the routing graph. Then in this situation, the weights of edges may change during this path searching. We assume that all the car trips satisfy the first-in-first-out (FIFO) property[3], which is exhibited in many networks, especially transportation networks [12]. Then we can solve this dynamic shortest path problem in multiple stages, where a stage is corresponding to a time slot.

We define the weight of an edge e_k in a particular time slot l as $w(e_k, l)$. Then an edge has a total of L different weights, where L is the total time slot number. If a trip departs at a particular time within time slot l, then we begin to search the shortest path using the weights $w(e_k, l)$. When the arrival time at any node spans a time slot during the searching process, we then change the cost of every edge as $w(e_k, l+1)$ and continue searching. If we reach the last time slot in a day, we next switch to the first time slot of the next day. We repeat this process until we reach the destination node.

[3]The FIFO property stipulates that vehicles exit from an edge in the same order as they entered, so that delaying one's departure along any path never results in an earlier arrival at an intended destination.

For each particular car trip, we calculate their personalized travel time and fuel consumption. Recall that we have obtained the driving style index for each driver and the fuel efficiency index for each vehicle. Thus, when calculating the travel cost of a trip related to a particular driver u, we calibrate the edge cost of the routing graph by multiplying the original cost by DS_u to obtain the driver specific cost. Besides, if a trip is related to a vehicle type v, the specific fuel consumption cost is the product of corresponding benchmark value and the fuel efficiency index FE_v. For those new drivers without historical records or car trips without specifying vehicle type, we set the two indexes both as 1 so that their predicted cost is the same as the benchmark value. In this way, we finally obtain the package delivery time and the corresponding travel cost for each particular trip.

IV. Online Task Assignment

In this section, we consider how to assign the tasks to proper car trips in the online scenario. We first analyze the problem from the offline problem formulation. Then we conduct a comprehensive analysis on the reward design mechanism and develop a heuristic online algorithm to solve the real-time assignment effectively.

A. Problem Formulation

To better understand the challenges of implementing the online task assignment mechanism, we start from analyzing its offline scenario with all the task and trip information known in advance.

Practically, a driver who submits the trip plan will accept package delivery tasks when offered a reasonable reward. We assume that a driver's expected reward ER is proportional to the extra cost for package delivery with a base reward ER_0. The base reward is necessary since in the situation that a task and a trip share the identical path, the driver still needs an incentive to take the task though almost no extra cost exists. The expected reward of tasks $\mathbb{S} = (\mathcal{P}_1, ..., \mathcal{P}_i)$ assigned to the trip $\mathcal{T}_j$ is represented as

$$ER_j = ER_0 + \eta[\mathcal{C}(\widetilde{\mathcal{T}}_j) - \mathcal{C}(\mathcal{T}_j)] \tag{1}$$

where η is the expected reward coefficient and $\widetilde{\mathcal{T}}_j$ indicates the new trip with the delivery tasks. Thus, the *actual gain* $AG^{\mathbb{S}}_j$ for these tasks $\mathbb{S}$ taken by trip j is:

$$AG^{\mathbb{S}}_j = \sum_{i \in \mathbb{S}} \mathcal{P}_i.v - ER_j \tag{2}$$

where $\mathcal{P}_i.v$ is the value of task i.

We aim to assign delivery tasks to proper candidate trips to maximize the total utility of the platform. Specifically, given a set of package delivery tasks $\mathbb{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n)$ and a set of trips $\mathbb{T} = (\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_m)$, find the task assignment $< (\mathcal{P}_{i_1}, \mathcal{P}_{i_2}, ..., \mathcal{P}_{i_k}) \rightarrow \mathcal{T}_j >$ with the objective of maximizing $\sum U(\widetilde{\mathcal{T}}_j)$ for every $\mathcal{T}_j \in \mathbb{T}$, where U is the total utility to the platform, and $\widetilde{\mathcal{T}}_j = (\mathcal{T}_j, (\mathcal{P}_{i_1}, \mathcal{P}_{i_2}, ..., \mathcal{P}_{i_k}))$ indicating the new trip with the assigned tasks.

We denote $x_{ij} \in \{0,1\}$ as a variable indicating whether task $\mathcal{P}_i$ is assigned to trip $\mathcal{T}_j$. Let $ArrT(\widetilde{\mathcal{T}}_j, \mathcal{P}_i)$ denote the arrival time of $\mathcal{P}_i$ in trip $\widetilde{\mathcal{T}}_j$. Then the offline task assignment problem can be formulated as follows:

$$Max: \sum_{j=1}^{m}\left[\sum_{i=1}^{n} x_{ij}\mathcal{P}_i.v - ER_j\right] \tag{3}$$

$s.t.$

$$\sum_{j} x_{ij} \leq 1, \forall i \tag{4}$$

$$\sum_{i} x_{ij} \leq N_j, \forall j \tag{5}$$

$$\mathcal{P}_i.ddl \geq ArrT(\widetilde{\mathcal{T}}_j, \mathcal{P}_i), \forall i \tag{6}$$

Constraint 4 specifies that any task is taken by at most one trip, namely package itself is indivisible. Constraint 5 indicates that the upper limit of allowed delivery package amount is N_j for each trip $\mathcal{T}_j$, given the vehicle space limitation. And constraint 6 indicates that the package arrival time should be no later than the required deadline.

From the offline problem formulation, we know that a driver will have a small extra cost when assigned tasks that have similar trip routes with his/her original trip plan. This further leads to a low expected reward and high utilities to the platform. However, in practice, drivers' trips may not be pre-known well ahead by the platform, as drivers usually submit their trip plans when they are ready for departure. The platform has to make an assignment decision and react to the driver instantly. How to make an assignment decision is not easy. On one hand, if the platform assigns tasks to a particular trip, there may appear another more attractive trip with smaller expected reward in the near future. On the other hand, if the platform skips this trip and keeps waiting for other more suitable trips, it may happen that there is no better or even no other suitable trips for this package delivery. In this situation, the platform has to send dedicated couriers for the corresponding delivery tasks, gaining no profit. Thus, it is challenging for the platform to decide how to assign proper tasks to trips submitted by drivers in an online fashion.

B. Online Assignment Algorithm

In the online task assignment scenario, both the tasks and trips arrive over time. Our target is to serve each coming trip instantly and assign the most suitable tasks to each trip to achieve maximal utility. A reasonable online task assignment mechanism should consider both the gain of the current task-trip matching and the possibility of having a better candidate trip in the future. For each delivery task i, we can estimate the necessary delivery time length c_i if dedicated couriers take the task. We define the *remaining waiting time* $WT_i(t)$ as

$$WT_i(t) = \mathcal{P}_i.ddl - t - c_i \tag{7}$$

where t is the current time. This indicates how much remaining time the platform can use to wait for other possible available trips before it is mandatory to send couriers for package delivery. When there is still a long time before the necessary delivery process, the platform may have a strict requirement for the target trip, namely, the expected gain should be relatively large. This actually requires that the tasks and the trip share more similar route paths. With the time passing, the platform should gradually relax the requirement for the target trip and allow a smaller gain. When the deadline further becomes very urgent, any candidate trip that will lead to a positive gain can be selected as the target trip.

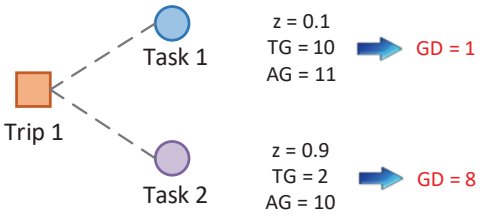

Fig. 6. An example of gain difference based selection. The platform prefers to assign task 2 to trip 1 considering the larger gain difference.

To this end, we seek to use a decaying function to describe the threshold gain of every task currently on the platform. We first introduce the *waiting ratio* $z_i(t)$ for a task i as the ratio of already waited time to the total available waiting time:

$$z_i(t) = \frac{t - \mathcal{P}_i.t}{\mathcal{P}_i.ddl - \mathcal{P}.t} \tag{8}$$

The waiting ratio of a task is actually served as a unified indicator that describes the level of urgency to deliver this task. We next design an exponential decaying function to indicate the *threshold gain* $TG_i(z)$ for assigning the task $\mathcal{P}_i$:

$$TG_i(z) = \left(\frac{\mathcal{L}_i}{\mathcal{U}_i}\right)^z \mathcal{U}_i \tag{9}$$

where $\mathcal{U}_i$ and $\mathcal{L}_i$ indicate the upper bound and the lower bound of the expected gain of the platform for delivering task i. Given the current time and task information, we can calculate the waiting ratio and the threshold gain of each task. Then for each task-trip matching, we can obtain the *gain difference* as

$$GD_j^i = AG_j^i - TG_i(z) \tag{10}$$

which indicates the amount of gain beyond expectation.

It is worth noting that our online assignment mechanism considers the gain difference as the trip selection criterion rather than the actual gain. When there are multiple feasible tasks for a trip, we prefer to select the matching with the maximized gain difference rather than the maximized actual gain. This is because the gain difference actually considers not only the current gain but also the future possible gains. If the current trips cannot have a positive gain difference, we skip these trips and wait for future trips. We use a simple example to explain this as illustrated in Fig. 6. In this situation, although choosing task 2 will get a lower actual gain (10 compared to 11 when choosing task 1), this gain difference is very large,

Algorithm 1: Online Task Assignment Algorithm

Input: Current time t, task set $\mathcal{P}$, the submitted trip $\mathcal{T}_j$, and the task limit N_j for this trip.
Output: The task assignment M.

```
Calculate the gain difference for every task-trip matching;
Select the tasks with GD > 0 as a candidate set W;
Set the allocated task set M = ∅;
while W ≠ ∅ do
    Find task i with largest GD_j^i ∈ M and add it to M;
    if M == N_j then
        break;
    Recalculate the GD for other tasks in W;
    Remove tasks with GD ≤ 0 from W;
return Task assignment result M;
```

indicating a gain far beyond the expectation, especially when the waiting ratio is 0.9.

Based on this principle, we try to assign proper tasks to maximize the sum of the gain difference. However, this allocation problem has a property that the cost sum of taking a set of tasks at once is not equal to the sum of taking these tasks individually. That is to say, the trip cost for taking a set of tasks is an uncertain function of these tasks. In this way, there are a total of 2^N combination, where N is the remaining number of tasks to be assigned. When we consider the realistic car space limit, each trip j is limited to take at most N_j packages. Even this realistic constraint reduces the total combinations, this problem is still not straightforward to solve.

To this end, we propose a heuristic algorithm to allocate the suitable tasks to each coming trip, as illustrated in Algorithm 1. The algorithm can be divided into three steps as follows. When a trip plan $\mathcal{T}_j$ is submitted to the platform, we first calculate the gain difference GD_j^i for every available task $\mathcal{P}_i$ and select all the tasks with positive gain differences into a candidate task set. Second, we select the task $\mathcal{P}_{i'}$ that has the largest gain difference from the candidate set and assign it to the trip. Note that the travel routes can have multiple choices when a trip takes multiple tasks. The route selection can be reduced to the traveling salesman problem (TSP) which is NP-hard. Here we simply search for the next nearest feasible stop (a picking up node or a dropping off node) from the trip departure node and repeat this process iteratively in a greedy mechanism until we reach the destination node. In this way, we can uniquely determine the routing path for a particular trip with tasks and the gain difference as well. Third, we recalculate the gain difference between the newly updated trip and all the remaining tasks in the candidate set and remove those tasks with non-positive gain difference from this set. We repeatedly conduct the second and third steps for task assignment until no suitable tasks can be assigned anymore or the assigned task number reaches the space limit N_j.

V. Evaluation

In this section, we conduct extensive real-world trace-driven simulations to show the performance of our system. We first evaluate the impact of different metrics on the accuracy of the edge cost prediction. Then we compare our online task assignment solution with baseline methods to evaluate its performance.

A. Experimental Setup

We first introduce the data of car trips and package delivery tasks we use in our evaluations and the baseline method for comparison as follows:

We use the real-world traces of car trip trajectories for evaluations. We have closely collaborated with Mojio[4], a leading open platform for connected cars, and collected a trajectory dataset of private cars in Vancouver from 09 June 2016 to 30 June 2016. This dataset includes more than 8,634,000 data entries of more than 382 vehicles and 25,978 car trips, recording the timestamp, GPS information, remaining fuel level, car type, odometer and detailed driving information (such as speed, acceleration, deceleration and RPM). We split the entire dataset into the training set (first two weeks) and the testing set (the last week) in the edge cost prediction. And we use the whole data as car trips in the online task assignment.

We empirically generate the data for evaluation and restrict the deadline for each task in a single day between 9 am to 7 pm. Considering that most packages are actually delivered in the afternoon and only a small fraction of them are delivered in the morning, we generate the deadline following Gaussian distribution, i.e., $ddl \sim N(\mu_{ddl}, \sigma_{ddl}^2)$, where the mean value μ_{ddl} is set at 4 pm considering the actual situation. We randomly generate the source and destination for each task. Those tasks with very short delivery distance (e.g., less than 1 km) are removed out since it is not a big overhead using other delivery methods (e.g., people can fetch their packages themselves if the distance is very small). We assume that tasks are released at the beginning of each day.

We implement two baseline methods for comparison, namely, *Largest Waiting Ratio First* (LWRF) and *Largest Actual Gain First* (LAGF). In LWRF, all the tasks are first sorted in a queue by their waiting ratios in descending order. For each coming trip, we try to assign the headmost task to the trip if the trip can satisfy the deadline requirement and the actual gain is positive. We repeat this process until no available trip in this batch can be assigned. In LAGF, we use the similar process in our online algorithm except that we every time select the task-trip matching with the largest actual gain rather than considering maximizing the gain difference. The LAGF mechanism actually only focuses on the current gain, without considering the future potentially more suitable trips.

B. Evaluation on Edge Cost Prediction

We consider the error ratio of travel time *(ERT)* and that of fuel consumption *(ERF)* as the metrics for edge cost

[4]https://www.moj.io/

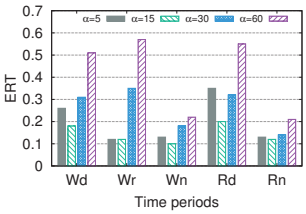

Fig. 7. Error ratio of travel time with various time slot settings.

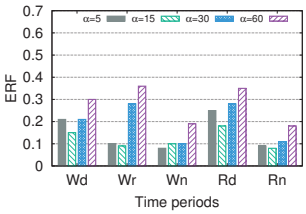

Fig. 8. Error ratio of fuel consumption with various time slot settings.

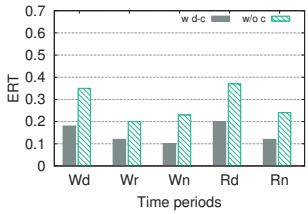

Fig. 9. Error ratio of travel time when considering drivers' driving styles or not.

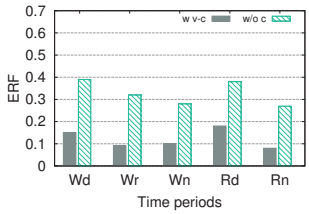

Fig. 10. Error ratio of fuel consumption when considering vehicles' fuel efficiencies or not.

prediction, defined as $ERT = Ave(\frac{|Pred\ time - Real\ time|}{Real\ time})$ and $ERF = Ave(\frac{|Pred\ fuel - Real\ fuel|}{Real\ fuel})$. We next evaluate the impact of the triple-dependent analysis on the prediction accuracy.

We first examine the impact of the time slot granularity α (recall §III-A) on the cost prediction by setting different time slot intervals, such as 5, 15, 30 and 60 minutes.

Fig. 7 illustrates the ERT of edges in the routing graph under the different time slot settings. We can find that when the time slot setting (α) is 15 minutes, the error ratio of travel time is minimal compared to the real data records. When α is 5 minutes, the time slot interval is so small that many roads may not have dense data records. And the travel time cost can be less accurate with a small data size. On the other hand, if we use a very large time slot setting (e.g., 30 minutes and 60 minutes), the road conditions during a time slot can vary a lot due to the coarse-grained time slot setting, leading to a large variance in travel time prediction. ERT also shows great differences in different time periods. Compared to other time periods, W_r, W_n and R_n have relatively smaller ERT, which is probably due to the more consistent road conditions (e.g., most roads are crowded in W_r and are quite clear in nighttimes).

Similarly, we also evaluate the ERF with different time slot settings, as illustrated in Fig. 8. We can find that when we set $\alpha = 15$, fuel consumption prediction achieves the smallest error. In rush hours, the ERF is less than 10% when $\alpha = 15$, while the error ratio increases to more than 30% if $\alpha = 60$. This is because even in rush hours the traffic congestion level varies according to the different time. The overly coarse setting, such as 60 minutes, can include many situations with diverse road conditions and make the prediction inaccurate. In this group of evaluation, we find that the setting of $\alpha = 15$ has the best result among all the other settings, where our prediction can achieve less than 20% error ratio in all time periods for both travel time and fuel consumption.

We next evaluate the impact of considering drivers' driving styles and vehicles' fuel efficiencies on the edge cost prediction. Fig. 9 illustrates the comparison between the ERT with the consideration of driving styles ($w\ d-c$) and the result without such consideration ($w/o\ c$). We can find that our model achieves much smaller error ratio of travel time in all time periods, where the raw prediction (i.e., without driving style consideration) achieves more than 35% error ratio in W_d and R_d. Given that different drivers have different driving styles, this diversity can lead to a large cost variance, even the road conditions are identical. Through driving style based analysis, we remove this impact factor when considering the benchmark travel time cost, and reconsider this factor when predicting the cost for each individual trip, enabling a personalized trip time prediction.

The fuel consumption prediction even reveals a bigger difference between ERF with the consideration of fuel efficiencies ($w\ v-c$) and the result without this consideration ($w/o\ c$), as illustrated in Fig 10. We can find that without considering the impact of different fuel efficiencies for different vehicle type, the fuel cost prediction can significantly deviate from the actual fuel consumption, achieving an average of 35% ERF among all time periods. In contrast, we consider the different fuel efficiencies of different vehicle types and thus is able to achieve a more accurate prediction result, which is about 12% among all time periods.

C. Evaluation on Task Assignment

We first explain several additional settings before the evaluation on task assignment. We define the *extra cost* of a trip for taking the package delivery tasks as $\beta * extra\ time + \gamma * extra\ fuel$, where β is set as the minimum legal hour rate (e.g., \$12/hour) and γ is set as the typical fuel price (e.g., \$1.4/L). The average package delivery distance $AveDis$ is the distance between the pick up location and drop off location. If we want to generate package sets with a larger $AveDis$, we then remove out the tasks with very small delivery distance and regenerate the same amount of tasks repeatedly until $AveDis$ is achieved, and vice versa. The *ratio of packages to trips (PTR)* indicates the ratio of the total task numbers to the total trips numbers, denoted as $|total\ tasks|/|total\ trips|$. For simplicity, we set the value for a citywide package delivery task as a constant value V (e.g., \$16 as a typical value for same day delivery), and base reward ER_0 as \$3. We set $\mathcal{U}_i$ and $\mathcal{L}_i$ as $V - ER_0$ and 1, respectively. We set the package amount limit for every trip as 3. The default settings of parameter PTR, η, σ_{ddl} and $AveDis$ are 0.4, 1.5, 1.5 and 15km, respectively.

We first evaluate the impact of different PTR on the total utility of the platform. Fig. 11 shows the total utility when we set different PTRs. We can observe that as the package delivery task amount keeps increasing, all the assignment approaches tend to have a higher utility. When the PTR is over 0.6, the

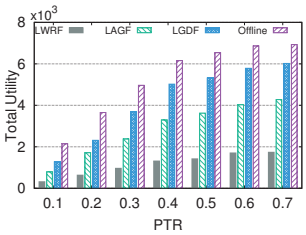

Fig. 11. The total utility when setting different PTRs.

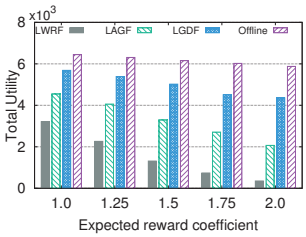

Fig. 12. The total utility when setting different expected reward coefficients.

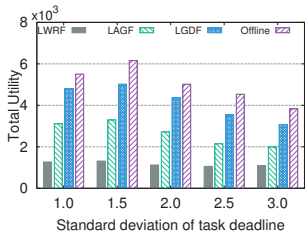

Fig. 13. The total utility when setting different standard deviation of task deadlines.

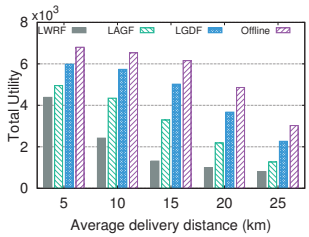

Fig. 14. The total utility of setting different average task delivery distances.

utility becomes relatively stable, rather than rapidly increasing when the PTR is low. Our algorithm (LGDF) can achieve an average of 52% improvement compared to the best baseline method, and has only 15% lower utility than the offline optimal assignment on average when there are enough tasks. These results indicate that our online algorithm is highly effective with different PTRs.

We next consider the impact of different expected reward coefficient η, as illustrated in Fig. 12. We can find that when the coefficient is getting larger, the utility of the platform plummets down when adopting the baseline assignment methods, such as LWRF and LAGF. However, our algorithm is still able to reach a relatively high utility (2x of LAGF and 12x of LWRF) and only has a small difference compared to the offline assignment. This result indicates that our online algorithm is more efficient in assigning proper tasks to the coming trips even when drivers' expected reward coefficient is very high.

Besides the previous two factors, the total utility of platform is also affected by standard deviation of task deadline σ_{ddl}, as illustrated in Fig. 13. We can observe that when the mean deadline is set as 4 pm and the deviation σ_{ddl} is 1.5, the platform has the maximum utility compared to other parameter settings. When the σ_{ddl} keeps increasing, the utility begins to decrease since the deadlines of many tasks are too early. Then dedicated couriers have to be sent more frequently and the utility is reduced. Nevertheless, even when the deadline distribution is very widely spread (e.g., $\sigma_{ddl} = 3$), our online algorithm still remarkably outperforms other two baseline methods by 54% and 281%, respectively. This result indicates that our algorithm is more capable of assigning tasks even when the deadlines are very tight.

We last evaluate the impact of different task delivery distance on the total utility, as described in Fig. 14. As the average delivery distance increases, the performance of all the methods gets worse accordingly. This is probably because the excessively far delivery distance may exceed many people's daily car trip distance, where the arrival deadline cannot be satisfied or the extra cost is too large. Even in this situation, our algorithm can still achieve higher utility than baseline methods. When the average delivery distance is 25km, our LGDF mechanism still has a utility of 2268, achieving a 78% improvement than the best baseline method.

VI. Related Work

A. Crowdsourced Delivery

Crowdsourced delivery has attracted a lot of researches in recent years due to its unique advantages in improving resource utilization, reducing the carbon emission, and satisfying the ever increasing delivery demands. There are a few existing works particularly targeted at the package delivery problem. Chen et al. [7] exploited the relays of taxis with passengers to deliver packages along with the passenger transportation process. They assumed that packages can be temporarily stored at the interchange station in between taxi rides and proposed a two-phase routing plan to minimize the delivery time during the taxi relay deliveries. Yet this relay process actually affected drivers' profit from passenger delivery, which limited its practical usefulness. Sadilek et al. [13] proposed a crowdsourced package delivery mechanism that one user delivers the assigned package to another user nearby, which continues until the package reaches the destination. However, since the deliverer process is hop-by-hop and needs large scales of participants, this delivery mechanism is also limited for practical use and can result in an uncontrollable delivery process. Some related works [14]–[17] also designed models to find optimal carpool routing for urban ridesharing. They focused on finding suitable matching between trips and passengers with personalized travel plans.

Different from these approaches, we solve the last mile package delivery through crowdsourced car trip sharing. By analyzing the cost of taking multiple tasks for a trip, we develop an effective and practical task assignment mechanism aiming to maximize the utility of the platform.

B. Trajectory Data Mining and Route Planning

The trajectory data mining and route planning are the foundations in RaaS based delivery and also play important roles in our system. Many recent works have mined massive trajectory based data [8], [9], [18]–[20] and proposed intelligent route planning model towards different targets. Ding et al. [21] considered the dynamic road network with the cost varying from time to time and studied the problem of finding the best departure time to minimize the total travel time from the source to the destination. Yuan et al. [8] mined the smart driving directions and learned the user driving behavior from massive trajectories to find the fastest route to a destination

from a given departure time. Baum et al. [22] focused on the key problem of electric vehicle route planning, i.e., finding a routing path that minimizes the energy consumption, and developed a practical algorithm achieving fast computation. Some works [9], [19] further exploited more comprehensive routing targets considering distance, fuel cost, time cost, weather, etc., to achieve personalized routing recommendation based on the individual preference of each driver.

Our work not only identifies fine-grained time-varying traffic conditions but also consider the practical individual driving styles and vehicle fuel efficiencies. In this way, we can accurately predict the travel cost for each individual driver and vehicle, and accordingly schedule the most suitable tasks for package delivery in an online manner.

VII. Discussion and Future Work

We propose a practical RaaS based package delivery system in this paper. Yet in real applications, such RaaS based service can be more complicated and we may need to consider more comprehensive impact factors. Our system can be further extended from the following aspects to make it more practical for use.

First, we can consider more geographical impact factors such as the weather information for the travel cost prediction if we can have such data. The traffic conditions may vary a lot under different weather conditions. For example, in the snowy days, the average vehicle speed can be largely affected due to the slippery road conditions. Yet in sunny days, the good road conditions will enable a faster average speed. Indeed, our system can be easily extended to integrate the weather features. Similar to the processing of the vehicle fuel efficiency, we just need to distinguish this feature in our data analysis and calibrate it for the personalized cost prediction.

Second, we can conduct more in-depth optimization from the aspect of delivery tasks if we have the real package delivery information. For example, different regions may have different delivery demand, e.g., industrial areas usually have very low delivery demand while high-density residential areas can have very high demand. To maximize the platform's utility, an intuitive idea is to prioritize the task assignment in the high demand areas since one trip can possibly take many delivery tasks. We are planning to cooperate with logistics companies to get more detailed delivery demand data for further optimization. We leave this for our future work.

VIII. Conclusion

In this paper, we proposed to explore the crowdsourced trip information of connected vehicles for ridesharing based package delivery. To achieve this, we first mined the characteristics of the travel time and fuel consumption from our real-world vehicle trajectory dataset and built up a routing graph accordingly. We conducted a personalized travel cost prediction considering the time-varying traffic conditions, different drivers' driving styles and different vehicles' fuel efficiencies. We further calculated the best routing paths as well as trip costs for each particular car trip. We then designed an online algorithm to assign proper tasks to the submitted car trip aiming to maximize the utility of the platform. Our extensive real-world trace-driven experiments showed that our algorithm can yield high-quality assignment results.

References

[1] T. Aized and J. S. Srai, "Hierarchical modelling of last mile logistic distribution system," *The International Journal of Advanced Manufacturing Technology*, vol. 70, no. 5-8, pp. 1053–1061, 2014.

[2] "1.5 billion packages to be shipped during chinas singles' day." http://fortune.com/2017/11/11/china-singles-day-alibaba-2/.

[3] "Argos customers have been left furious by long delays to deliveries after black friday." http://www.dailymail.co.uk/news/article-3341898/Argos-customers-fury-delays-Black-Friday-deliveries.html.

[4] M. Joerss, J. Schröder, F. Neuhaus, C. Klink, and F. Mann, "Parcel delivery–the future of the last mile," *McKinsey & Company*, 2016.

[5] J. An, K. Yang, J. Wu, N. Ye, S. Guo, and Z. Liao, "Achieving sustainable ultra-dense heterogeneous networks for 5g," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 84–90, 2017.

[6] L. Zhang, B. Jin, and Y. Cui, "A concurrent transmission enabled cooperative mac protocol for vehicular ad hoc networks," in *Proceeding of 22th International Symposium on Quality of Service (IWQoS)*, pp. 258–267, IEEE, 2014.

[7] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, and E. Sha, "Crowddeliver: planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1478–1496, 2017.

[8] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL GIS*, pp. 99–108, ACM, 2010.

[9] B. Yang, C. Guo, Y. Ma, and C. S. Jensen, "Toward personalized, context-aware routing," *The VLDB Journal*, vol. 24, no. 2, pp. 297–318, 2015.

[10] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[12] B. C. Dean, *Continuous-time dynamics shortest path algorithms*. PhD thesis, Massachusetts Institute of Technology, 1999.

[13] A. Sadilek, J. Krumm, and E. Horvitz, "Crowdphysics: Planned and opportunistic crowdsourcing for physical tasks," in *Proceedings of the 7th International Conference on Weblogs and Social Media (ICWSM)*, ACM, 2013.

[14] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[15] W. He, K. Hwang, and D. Li, "Intelligent carpool routing for urban ridesharing by mining gps trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2286–2296, 2014.

[16] S. Yan, C.-Y. Chen, and Y.-F. Lin, "A model with a heuristic algorithm for solving the long-term many-to-many car pooling problem," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1362–1373, 2011.

[17] E. Kamar and E. Horvitz, "Collaboration and shared plans in the open world: Studies of ridesharing.," in *IJCAI*, vol. 9, p. 187, 2009.

[18] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th SIGKDD*, pp. 316–324, ACM, 2011.

[19] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *Proceeding of the 31st International Conference on Data Engineering (ICDE)*, pp. 543–554, IEEE, 2015.

[20] X. Fan, J. Liu, Z. Wang, Y. Jiang, and X. S. Liu, "Crowdnavi: Demystifying last mile navigation with crowdsourced driving information," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 771–781, 2017.

[21] B. Ding, J. X. Yu, and L. Qin, "Finding time-dependent shortest paths over large graphs," in *Proceedings of the 11th International Conference on Extending Database Technology (EDBT)*, pp. 205–216, ACM, 2008.

[22] M. Baum, J. Dibbelt, T. Pajor, and D. Wagner, "Energy-optimal routes for electric vehicles," in *Proceedings of the 21st ACM SIGSPATIAL GIS*, pp. 54–63, ACM, 2013.