QoS Guaranteed Edge Cloud Resource Provisioning for Vehicle Fleets

Guoming Tang, *Member, IEEE*, Deke Guo, *Senior Member, IEEE*, Kui Wu, *Senior Member, IEEE*, Fang Liu, Yudong Qin

***Abstract*—Nowadays vehicle fleets are launched to perform business or scientific tasks, with new features supported by the emerging multi-access edge computing (MEC) platform. In the presence of high vehicle mobility, however, it is challenging to precisely provision resources among distributed edge clouds so that i) the QoS of vehicular service is guaranteed and meanwhile ii) the provisioning cost is minimized. We systematically investigate the QoS guaranteed optimal resource provisioning problem for the connected vehicle fleet in the MEC environment. Based on stochastic traffic analysis, we propose an optimization framework to minimize the cost of resource provisioning, while the service blocking probability is guaranteed to be smaller than a predefined threshold. We then present a lightweight two-phase algorithm based on bracketing and binary searching to solve the problem efficiently. To evaluate our method, we use two large real-world datasets collected by an online taxi service platform and validate the QoS with our resource provisioning strategy. The results demonstrate that our method can save the total provision cost up to 40%, compared with the naïve resource provisioning strategy, and meanwhile can provide reliable QoS guarantee, compared with the mobility estimation-based approach.**

I. Introduction

The market of connected vehicles, including the connected autonomous driving (AD) vehicles, is growing rapidly with a five-year compound annual growth rate of 45%, which is 10x faster than the overall car market [1]. According to the statistical report from Statista [2], the revenue in the connected vehicle market accounted for $\$8.2$ billion in 2017 and will grow up to over $\$18$ billion by 2021 in the U.S. alone.

Recently, more and more vehicle manufacturers, retailers and owners are marching into the connected vehicle market. They own and manage a fleet of vehicles, consisting of hundreds or even thousands of cars, taxis or trucks, to undertake specific business or scientific tasks. For example, Google [3], Lyft & Aptiv [4], GM [5] and JD.com [6] have launched their own fleets of connected (AD) vehicles in cities, on highway backbones, or around port wharves, for the purposes of street scene capturing, passenger delivering and goods shipping.

The efficient and safe running of connected vehicle fleet is technically provided by advanced vehicular applications, including i) intra-vehicle assistant services, e.g., self-parking and automated driving [7], and ii) inter-vehicle cooperated services, e.g., traffic monitoring and alerting, optimal route navigation, and real-time videos sharing [8]. While these services introduce new digital vehicular features, they pose significant technical challenges on supporting high computation, low latency, and high bandwidth in the vehicular network.

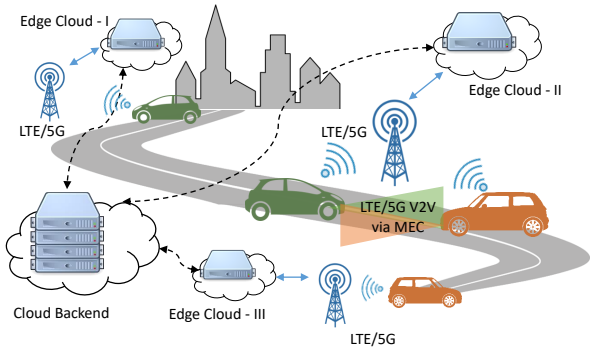


Fig. 1. LTE/5G aided MEC architecture for connected vehicle applications. Note that the edge cloud can be located either at the base stations (or cell towers) upgraded by mobile operators or in the SDN/NFV-enabled (mini) datacenters deployed by telecom equipment vendors (e.g., IBM, Huawei, and Ericsson) and IT platform vendors (e.g., AWS and Azure).

The above challenges can be tackled with the technology of mobile edge computing, with recent evolution to multi-access edge computing (MEC) [9], [10]. Under the MEC paradigm, traditional backend cloud environment is moved to the network edge (named *edge cloud*), and is thus much closer to the end users. Further aided by the upcoming LTE/5G networks, MEC is expected to provide a high quality of service (QoS) for the connected vehicle fleet [11]. Fig. 1 gives a big picture of the MEC architecture and its role played in the connected vehicle fleet. As shown in the figure, the connected vehicles get access to the cellular network via their nearest base stations (or cell towers), and make use of resources provided by distributed edge clouds for particular vehicular applications.

To ensure the QoS for the connected vehicle fleet, we need to carefully provision sufficient resources[1] for the distributed edge clouds. To be specific, the resources of each edge cloud should be sufficient at any time instant, upon the requests of vehicles, especially those running under high levels of autonomous driving [12]. The shortage of edge resources may incur the service delay, and in some cases such a delay might be fatal. Hence, edge resource provisioning cannot be more

G. Tang is with the Peng Cheng Laboratory, Shenzhen, Guangdong, China. D. Guo and Y. Qin are with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan, China. K. Wu is with the Department of Computer Science, University of Victoria, Victoria, BC, Canada. F. Liu is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, Guangdong, China. Corresponding authors: G. Tang (tanggm@pcl.ac.cn), D. Guo (dekeguo@nudt.edu.cn).

[1]The resources at an edge cloud in this paper generally refer to the computation, storage and bandwidth that are deployed for supporting specified vehicular services.

critical for the safe running of the connected vehicle fleet.

Under the condition of high vehicle mobility, how to optimally allocate resources among distributed edge clouds becomes extremely difficult. As demonstrated by a large body of investigations on vehicle mobility, it is hard to precisely compute the vehicle travel time from one location to another, due to the complex traffic conditions and diverse driving habits [13], [14]. Thus in our application context, we can only estimate the *approximate* vehicle travel time, from one mobile network cell[2] to another, particularly with a (travel time) distribution, rather than a precise value. This leads to the uncertainty of real-time vehicle arrivals and departures at each cell, as well as the edge resource requests during a short period. Consequently, precise resource provisioning to guarantee "seamless handover" of vehicular services between cells is almost impossible.

Simple over-provisioning with redundant resources beforehand is not ideal at best and infeasible at worst, due to the high cost and limited amount of resources deployed at the edge. Previous work on edge/cloud resource provisioning utilized the expected resources from either historical observations [15], [16] or mobility estimations [17], [18], and apply the expected values for provisioning. The expected values, however, can be much different from the actual ones from a probabilistic perspective. For instance, with the result of "the expected number of vehicle arrivals is 9.5" from an expectation-based solution, by no means it guarantees that the actual vehicle arrivals will not exceed 10. In other words, those expectation-based solutions cannot guarantee actual resource demands and thus may lead to a high rate of service failures in practice.

While there is a broad spectrum of work on resource provisioning for the mobile cellular network, the problem in the context of MEC for connected vehicle fleets is much different and even more challenging. With the commonly-used virtualization technology, redundant resources at the edge cloud can be temporarily freed and then reused whenever needed. In other words, resource provisioning at the edge cloud should be elastic, which differs from the static provision in the mobile cellular network. In addition, vehicle arrivals and departures in neighboring cells are highly correlated. Although such correlations can be helpful to the design of edge resource provisioning in a general view, to correctly identify and properly exploit them for an optimal resource provisioning solution are still challenging.

The above challenges directly motivate our work in this paper, in which we make the following major contributions:

- We systematically investigate the QoS guaranteed optimal resource provisioning problem for connected vehicles over the typical MEC architecture, where the request uncertainty caused by vehicle mobility has been carefully captured.
- With stochastic traffic analysis, we establish an optimization model to minimize the resource provisioning cost at each edge cloud, with the constraint of service blocking probability smaller than a predefined threshold. Using a *fleet mobility matrix* which will be defined later, our model well captures the correlation in vehicle mobility in neighboring cells. We also develop a two-phase algorithm to solve the problem based on bracketing and binary search.
- We evaluate our method with two large real-world datasets from an online taxi service platform, containing the trajectory information of $58,770$ vehicles in one month time. By extensive experiments, we demonstrate that our method could save the total provisioning cost up to 40% compared with the naïve resource provisioning strategy; meanwhile our method could always provide reliable QoS guarantee compared with the mobility estimation-based method.

The rest of the paper is organized as follows. Sec. II reviews the literature relevant to our work. Sec. III gives an overview of the targeted scenario and basic assumptions, as well as the formal problem representations. Sec. IV presents our traffic flow models and the stochastic traffic flow analysis, based on which an optimal resource provisioning problem is built. We solve the optimization problem under QoS guaranteed constraints in Sec. V and validate our approach by performing comparison experiments using real-world datasets in Sec. VI. Sec. VII concludes the paper.

II. Related Work

Both MEC and RAN (radio access network) resource provisioning are related to this work, while we mainly focus on the former in this paper. Research on resource provisioning under the MEC paradigm can be roughly grouped into two categories: (1) historical data based methods, and (2) mobility estimation based methods.

A. Historical Data based Methods

To satisfy the resource needed by the vehicle fleet, the simplest solution is to provision sufficient resources at each cell which can serve the whole fleet simultaneously. However, such a method will cause significant resource waste. To match the allocated resource with future resource demands, the historical data based methods have been proposed, which estimate future resource demands based on historical data. The method in [19] estimated the resource requests in each cell based on historical resource usage information. In [15], [16], vehicle historical trajectories were firstly used to predict the future trajectories, based on the observation that people drive along familiar routes more frequently. Then, the resources of each cell along the future trajectories could be estimated. Authors in [20] addressed the resource deployment problem by solving a mixed integer linear programming problem from a global view. It took the user mobility into consideration by combing the VM replication and migration technologies.

Methods in this category have two major pitfalls: i) There is no guarantee for sufficient resource allocation or optimal resources provisioning at any time, because "history does not happen everyday". ii) Such methods may work well only for situations where the movement pattern of vehicles repeats periodically, but it would be difficult to apply them in other

[2]In the rest of the paper, we use "cell" to represent "mobile network cell" for simplicity.

scenarios involving irregular vehicle movements, such as the online taxi services introduced in our later experiments.

B. Mobility Estimation based Methods

Since a vehicle cannot pass through multiple cells within a short time period, methods in this category normally consider neighboring cells. The method in [17] adopted a proactive resource allocation approach, in which *all* cells one-hop away from the current cell would be provisioned with resources. Obviously, such strategy may lead to resource over-provisioning and inefficient utilization. To reduce the resource waste caused by provisioning resource at all neighboring cells, the authors in [18], [21] proposed to allocate resources to a subset of neighboring cells. The subset was selected based on a weight value that indicates the handover probability with neighboring cells [18], and only the neighboring cells with weights larger than a threshold were selected. Moreover, in [21], the subset was selected by minimizing an objective function related to expected average delay and caching cost. Authors in [22], [23] built a user mobility model by using multiple users' traces. In addition to the user-defined mobility model, some work predicted mobility using Markov-based and compression-based predictors [24]. A continuous time Markov decision process was adopted to decide whether to deploy resources on the cell where the user located [25]. In this scenario, the user would be continuously served by the cloud resources in previous cell even if it reaches a new one, in which the deployment cost was valued more than the QoS.

Our method presented in this work belongs to this category, but it is different from the above ones. Previous mobility estimation based methods usually make resource provision using the expected values of vehicle arrivals/departures. In contrast, our method considers all possible values of vehicle arrivals/departures and guarantees a pre-defined blocking probability upon resource provisioning.

There are also related works on resource provisioning and allocation for MEC using different (and less strict) QoS measures. or example, in the latest work [26], [27], [28], instead of enforcing a rigid QoS threshold, the average service delay was minimized in designing the optimal resource allocation schemes.

III. Problem Representation and Assumptions

A. Overview

Fig. 2 illustrates an example scenario of the MEC-enabled connected vehicle fleet in a target area (i.e., the area where the vehicle fleet is expected to travel around), which consists of four cells (labeled by A, B, C and D). In each cell, the base station (or cell tower) initially constructed for the mobile network is also attached with an edge resource pool (i.e., the edge cloud) supplying resources for computation and data storage of vehicular services. The connected vehicles can travel freely in the target area as shown in Fig. 2.

Note that when a vehicle leaves cell A and drives into B, its original connection to mobile network in A will be cut down, and meanwhile a new connection to mobile network in B will be established. Ideally, the handover process should be seamless such that the vehicle will not be "out of service". For the MEC-enabled vehicular service, once the (radio) handover process is completed, the vehicle will make a migration request to the edge cloud deployed in cell B, and the edge cloud responds to the request by allocating necessary resources for the service migration (i.e., the MEC-based communication is in a request/response way). If cell B runs out of edge resources when the vehicle enters the cell, the vehicular service will be denied by the edge server or wait in a queue. In this case, we say the *service request is blocked*. Specifically, we have the following definition for QoS metric used in this work.

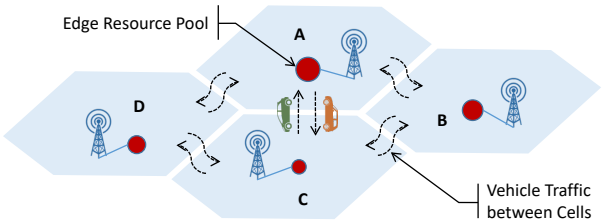


Fig. 2. An example scenario of the system model: the target area consists of 4 cells A, B, C, and D, each installed with a base station for communication and edge resources for computation.

Definition 1. ***Service blocking probability****, denoted by $P(\varnothing)$, is defined as the probability that a vehicle's request cannot be served due to the unavailability of edge resources in the MEC-enabled vehicular service, with the event of no available edge resources represented by $\varnothing$.*

Notice that with the above QoS metric, we can easily infer the *availability* of targeted vehicular service, which is one major concern in the service-level agreements (SLAs) provided to the customers.

Assume that for a particular task assigned to a connected vehicle fleet, the vehicles need to travel around the target area covered by N cells. Each of the cells has an edge resource pool. Since the available resources within each pool should fluctuate as the vehicles arrive and depart throughout the day, we use $C_n(t)$ to denote the required edge resources[3] for cell n at time t. Our goal is to ensure that there are always sufficient resources serving the requests of connected vehicles, while eliminating resource over-provisioning.

We assume a working time period for the connected vehicle fleet, e.g., 8:00 am to 6:00 pm, and then divide the time period into smaller equal time slots, e.g., 3 minutes. Then, we further assume that the edge resources should be provisioned one time slot ahead, because resource provisioning involves tasks such as environment preparation and VM/container allocation, which may be time consuming.

With the above assumptions, our goal is to find out the minimum resources that each edge cloud should be provisioned in each time slot, such that the service blocking probability of the vehicular service is smaller than a user-defined QoS threshold, such as $\epsilon = 0.01$.

[3]To ease analysis, one unit of edge resource here refers to the amount of resources (such as computation, storage and bandwidth) consumed by one vehicle in providing required services for a given time period (e.g., 3 minutes).

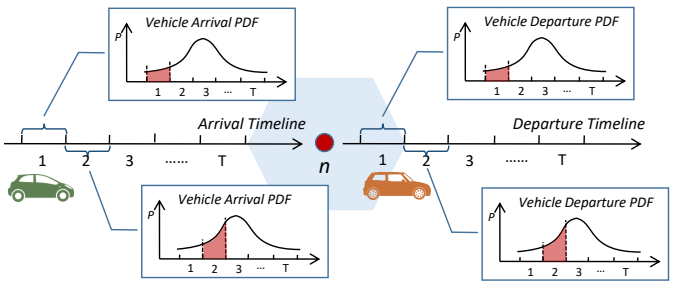


Fig. 3. Arrival and departure time analysis of vehicles at cell n.

B. Vehicle Fleet Modelling: States & Partition

Consider a fleet of connected vehicles to complete a given task, such as goods shipping or passenger delivery. Assume that the time slot sequences in a working day are denoted as $\{1, 2, \cdots, T\}$, where T is the total number of time slots. For an arbitrary time slot t, we define a *fleet distribution vector* $S(t)$, with $S_n(t)(n \leq N, t \leq T)$ representing the number of vehicles in cell n *at the beginning of time slot* t:

$$S(t) := [S_1(t), S_2(t), \cdots, S_N(t)], \tag{1}$$

where N is the total number of cells (or edge resource pools) in the target area. Note that the information of fleet distribution vector can be determined with the help of edge cloud service (like vehicle localization), and the total number of working vehicles in time slot t can be computed by $\sum_{n=1}^{N} S_n(t)$.

Then, within an arbitrary time slot t, we define a *fleet mobility matrix* $H(t)$ to include the overall moving directions of all vehicles at the beginning of this time slot:

$$H(t) := \begin{bmatrix} H_{1,1}(t) & H_{1,2}(t) & \cdots & H_{1,N}(t) \\ H_{2,1}(t) & H_{2,2}(t) & \cdots & H_{2,N}(t) \\ \vdots & \vdots & \ddots & \vdots \\ H_{N,1}(t) & H_{N,2}(t) & \cdots & H_{N,N}(t) \end{bmatrix} \tag{2}$$

in which $H_{i,j}(t)$ is the fraction of vehicles in cell i that are travelling towards cell j in time slot t, where $0 \leq H_{i,j}(t) \leq 1$ and $\sum_{j=1}^{N} H_{i,j} = 1$. Note that $H_{i,i}$ denotes the fraction of vehicles that would stay in the same cell i. Such information can also be obtained with the help of the edge cloud service, for example, by collecting the information of navigated routing path and real-time localization of each vehicle.

With the information of fleet distribution vector and fleet mobility matrix, we are able to compute fine-grained vehicle traffic flows: the number of vehicles in cell i that are moving towards cell $j(\neq i)$ in time slot t is $S_i(t)H_{i,j}(t)$. Furthermore, for an arbitrary time slot t, we can also calculate the total number of vehicles that are moving towards cell n by $\sum_{i=1,i\neq n}^{N} S_i(t)H_{i,n}(t)$ and the total number of vehicles that are leaving cell n by $\sum_{j=1,j\neq n}^{N} S_n(t)H_{n,j}(t)$.

C. Vehicle Arrival and Departure Analysis

In addition to knowing how many vehicles are moving towards where, we also need to know approximately *when* they will leave one cell and enter another, i.e., the time instant when a vehicle arrives at the boundary of two cells. As we have mentioned before, precise travel time estimation for vehicles nearly impossible. Instead, we should use an arrival time probability distribution function (arrival time PDF), which discloses the vehicle arrival time from a probabilistic perspective [13]. Similarly, we can also use the departure time PDF of a vehicle, which indicates the distribution of vehicle departure time.

Then, for an arbitrary cell n, as illustrated in Fig. 3, both the arrival time PDFs of vehicles moving towards cell n and departure time PDFs of vehicles leaving cell n in each time slot can be computed, with travel time estimation techniques (e.g., in [13]) and the computing capability provided by the edge cloud. Specifically, for an arbitrary time slot t, we denote the arrival time PDF of vehicle m towards cell n by $P^{(arr)}_{n,m,t}$ and the departure time PDF of vehicle m leaving cell n by $P^{(dep)}_{n,m,t}$, respectively.

Thus, for a specific time slot t, the expected number of vehicles that arrive at cell n (i.e., the vehicle *arrival rate* of cell n), denoted by $\lambda_n(t)$, can be calculated by:

$$\lambda_n(t) = \begin{cases} 0, \text{ if } M = 0, \\ \sum_{k=1}^{M} \int_t^{t+1} P^{(arr)}_{n,m_k,t}(t)dt, \text{ else,} \end{cases} \tag{3}$$

where $M = \sum_{i=1,i\neq n}^{N} S_i(t)H_{i,n}(t)$ is the total number of vehicles that are moving towards to cell n in time slot t.

Similarly, for a specific time slot t, the expected number of vehicles that departure from cell n (i.e., the vehicle *departure rate* of cell n), denoted by $\mu_n(t)$, can be calculated by:

$$\mu_n(t) = \begin{cases} 0, \text{ if } M = 0, \\ \sum_{k=1}^{M} \int_t^{t+1} P^{(dep)}_{n,m_k,t}(t)dt, \text{ else,} \end{cases} \tag{4}$$

where $M = \sum_{j=1,j\neq n}^{N} S_n(t)H_{n,j}(t)$ is the total number of vehicles that are leaving from cell n in time slot t.

Remark 1. *A main difficulty of our problem comes from the uncertainty on the times of vehicle arrivals/departures to/from a cell, which can be indicated by the variation of* $P^{(arr)}_{n,m,t}$ *(or* $P^{(dep)}_{n,m,t}$*). As a real case shown in Fig. 4, we can observe that* $P^{(arr)}_{n,m,t}$ *(or* $P^{(dep)}_{n,m,t}$*) indeed had a large variation. In the daytime particularly, the variation range was as high as* 20 *minutes.*

D. Problem Statement

As vehicles arrive and depart, the volume of available edge resources within each cell changes throughout the working time period. Unexpectedly, the service request of a vehicle towards an edge will be blocked if the resource of the edge has been used up upon arrival of the vehicle.

Mathematically, for an arbitrary time slot $t, 1 \leq t \leq T$ and a cell $n, 1 \leq n \leq N$, given the knowledge of fleet distribution vector $S(t)$, fleet mobility matrix $H(t)$, and vehicle arrival rate $\lambda_n(t)$ and departure rate $\mu_n(t)$, our goal is: to minimize the edge resources provisioned at the cell for this time slot $C_n(t)$, such that the service blocking probability at the cell is no bigger than a (small) user-specified QoS threshold ϵ, i.e., $P(\varnothing) \leq \epsilon$.

Remark 2. *Equations* (3) *and* (4) *cannot be used directly to determine the resources of edge cloud within the given cell.*

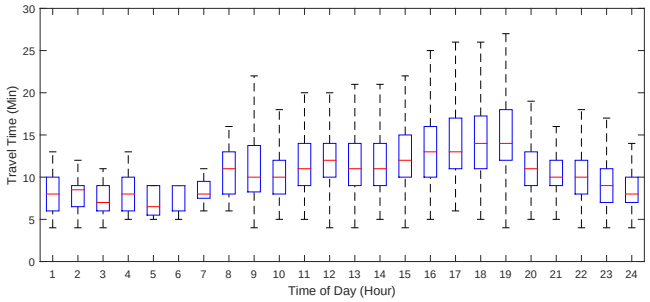


Fig. 4. The variation of vehicle travel time from one cell to another (A to B in Fig. 5), extracted from the dataset collected in a real-world online taxi service platform over one month time.

*These are **expected** values which may be different from **actual** values. For instance, given that the expected number of arrival vehicles during a future time slot is* 9.5*, it is not correct to conclude that provisioning* 10 *units of resources is sufficient during this time slot. Instead, we should apply probabilistic analysis to compute the blocking probability.*

IV. Optimal Resource Provisioning based on Stochastic Traffic Analysis

A. Traffic Flow Definition

To analyze the vehicle mobility among all the N cells, we first investigate the vehicle flows at a particular cell n. Then, making use of the arrival and departure information of vehicles in the current time slot, we calculate the vehicles' arrival and departure in the next time slot at cell n. By doing so for each cell, we can provide the one-time-slot-ahead resource provisioning for each edge cloud.

Referring to the mobility of individual vehicles illustrated by Fig. 3, we can model two vehicle flows at cell n:

- **Arrival flow**: it is the traffic flow formed by the vehicles that are arriving at cell n from all the other cells, with a rate of $\lambda_n(t), 1 \leq n \leq N, 1 \leq t \leq T$, given by Equation (3).
- **Departure flow**: it is the traffic flow formed by the vehicles that are leaving cell n towards other cells, with a rate of $\mu_n(t), 1 \leq n \leq N, 1 \leq t \leq T$, given by Equation (4).

During time slot t, to capture the number of vehicle arrivals at cell n before the next time slot, we model the arrival flow and departure flow as Poisson processes, which are widely used in customer-server queue systems and traffic estimation in transportation systems [29]. Thus, the arrival rates of the two Poisson processes are $\lambda_n(t)$ and $\mu_n(t)$, obtained via Equations (3) and (4), respectively.

As the arrival rate of the Poisson process in our context is estimated by the expected value of vehicle arrivals during one time slot, the shorter the time slot is, the more precise the model of Poisson process should be. In later evaluations, our experimental implementations (Sec.VI-A) and results (Sec.VI-C) show that, the time slot with a length of 3 minutes is short enough to yield accurate results.

B. Stochastic Traffic Flow Analysis

1) RV for Vehicle Arrival (Resource Demand): During the time slot t at cell n, we define a random variable (RV):

$$D_n(t) := \text{number of vehicle arrivals at cell } n.$$

Thus, according to our analysis in previous subsection, $D_n(t)$ can be estimated by following a Poisson process with rate $\lambda_n(t)$ and PDF:

$$P_{D_n(t)}(x) = \frac{(\lambda_n(t))^x e^{-\lambda_n(t)}}{x!}, \tag{5}$$

where $0 \leq x \leq \check{B}_n(t)$ with $\check{B}_n(t)$ representing the maximum possible number of vehicles arriving at cell n in time slot t. Thus, $\check{B}_n(t)$ is actually the upper bound of $D_n(t)$, and with the vehicle fleet modelling in Sec. III-B, $\check{B}_n(t)$ is given by:

$$\check{B}_n(t) = \sum_{i=1, i\neq n}^{N} S_i(t) H_{i,n}(t). \tag{6}$$

2) RV for Vehicle Departure (Resource Release): For vehicles leaving cell n in time slot t, we define another RV:

$$U_n(t) := \text{number of vehicle departures from cell } n.$$

Similarly to the vehicle arrival RV, $U_n(t)$ can also be estimated by following a Poisson process with rate $\mu_n(t)$ and PDF:

$$P_{U_n(t)}(x) = \frac{(\mu_n(t))^x e^{-\mu_n(t)}}{x!}, \tag{7}$$

where $0 \leq x \leq \hat{B}_n(t)$, with $\hat{B}_n(t)$ representing the maximum possible number of vehicles leaving cell n in time slot t (i.e., $\hat{B}_n(t)$ is the upper bound of $U_n(t)$). Also, with the vehicle fleet modelling in Sec. III-B, $\hat{B}_n(t)$ can be calculated by:

$$\hat{B}_n(t) = \sum_{j=1, j\neq n}^{N} S_n(t) H_{n,j}(t). \tag{8}$$

3) RV for Difference of Two Flows (Resource Difference): We further define the difference between $U_n(t)$ and $D_n(t)$ as:

$$f_n(t) := U_n(t) - D_n(t). \tag{9}$$

Consider the situation where $f_n(t) \leq 0$, i.e., the resource requested is greater than that of released. Then, according to the difference distribution of two discrete RVs, its PDF is:

$$P_{f_n(t)}(x) = \sum_{k=0}^{\hat{B}_n(t)} P_{U_n(t)D_n(t)}(k, k-x) \tag{10a}$$

$$= \sum_{k=0}^{\hat{B}_n(t)} P_{U_n(t)}(k) P_{D_n(t)}(k-x), \tag{10b}$$

where $P_{U_n(t)D_n(t)}(k, k-x)$ is the joint probability of $U_n(t)$ and $D_n(t)$, and $-\check{B}_n(t) \leq x \leq 0$.

Remark 3. *Notice that we can derive (10b) from (10a), as in a short time slot t (e.g.,* 3 *minutes in our later implementation), the vehicle arrival flow can be considered to be independent of the departure flow from the view point of cell n. For a long time period, however, this does not hold, as the vehicles arriving at a cell eventually will leave the cell.*

C. Optimal Resource Provisioning under QoS Guarantee

At the beginning of time slot t, given that there is an amount of $C_n(t-1)$ resources available in edge cloud within cell n (which is known), the probability that a random vehicle arriving from other cell is blocked at cell n is given by:

$$P(\varnothing|C_n(t-1)) = \begin{cases} 0, \text{ if } C_n(t-1) \geq \check{B}_n(t), \\ \sum_{x=-\check{B}_n(t)}^{-C_n(t-1)-1} P_{f_n(t)}(x), \text{ else.} \end{cases} \quad (11)$$

Note that those vehicles, which are in cell n at the beginning of time slot t and will stay in the same cell during this time slot, would not make any change on the available resources, because their resources have already been allocated at the beginning of time slot t.

Thus, our target is to find the minimum amount of edge resources at the end of the time slot $t-1$, such that:

$$P(\varnothing|C_n(t-1)) \leq \epsilon, \quad (12)$$

where ϵ is the user-defined blocking probability. Then, our optimization problem, for edge resources provisioning at cell n, can be formulated as:

$$\textit{minimize} \qquad C_n(t-1) \quad (13a)$$

$$\textit{s.t.} \qquad (12), \forall t \in \{2, 3, \cdots, T\}. \quad (13b)$$

In the above optimization problem, the decision variable is $C_n(t-1)$, i.e., the volume of resources provisioned at cell $n, 1 \leq n \leq N$ during time slot $t-1$, prepared for the next time slot t. Note that the optimal resource provisioning begins with the second time slot ($t = 2$). Without loss of generality, we assume that the system starts working at the beginning of the first time slot, when all edges have the demanded resources for the first time slot.

The optimization problem in (13) is formulated for the whole target area consisting of N cells. By solving it, we can find the minimum resources at any edge that can satisfy the blocking probability constraint for the next time slot. Thus, we achieve the goal of one-time-slot-ahead optimal resource provisioning under the given QoS guarantee.

V. Solution Methodology

A. A Two-phase Algorithm to the Optimization Problem

To solve the optimization problem and find the optimal volumes of resources for each edge cloud, we apply a two-phase algorithm with techniques of bracketing and binary searching (with pseudo-code in Algorithm 1).

- **Phase-1 (bracketing)**: Feed an initial small value of $C_n(t-1)$ and compute the corresponding blocking probability with Equation (11). If the resulted blocking probability is larger than the predefined QoS threshold ϵ, then double the value of $C_n(t-1)$. Repeat the above process until the desired blocking probability is reached, which will result in a resource volume range, denoted by $[C_L, C_H]$ where $C_L = C_H/2$.
- **Phase-2 (binary searching)**: Apply the binary searching strategy within range of $[C_L, C_H]$ to find the minimum value of C_{min} that can satisfy the QoS requirement, i.e., the blocking probability is smaller than ϵ.

Algorithm 1: A Two-Phase Algorithm to Problem (13)

Input: required QoS threshold ϵ, capacity (maximum resources) of the edge cloud C_{max}
Output: minimum edge cloud resources C_{min} that satisfies the conditions of (13b)

```
C_L, C_H ← 1;
// Phase-1: Bracketing;
while P(∅|C_H) > ε do
    C_L ← C_H ;
    C_H ← 2 × C_H ;
    if C_H > C_max then
        return false;
// Phase-2: Binary Searching;
while true do
    C_min ← (C_L + C_H/2) ;
    if P(∅|C_H) ≤ ε then
        C_H ← C_min;
    else
        C_L ← C_min;
    if C_H − C_L ≤ 1 then
        return C_min;
    else
        return false;
```

Theorem 1. *Algorithm 1 guarantees the convergence to the optimal value of problem (13).*

Proof. We first prove that the blocking probability of $P(\varnothing|C_n(t-1))$ (given by Equation (11)) is a monotonically decreasing positive function, with $C_n(t-1)$ as the variable in the region of $\{1, 2, \cdots, \check{B}_n(t)\}$.

When $C_n(t-1) < \check{B}_n(t)$ where $\check{B}_n(t)$ is the upper bound of $D_n(t)$, referring to Equations (10b) and (11), we have:

$$P(\varnothing|C_n(t-1)) \quad (14a)$$

$$= \sum_{x=-\check{B}_n(t)}^{-C_n(t-1)} P_{f_n(t)}(x) \quad (14b)$$

$$= \sum_{x=-\check{B}_n(t)}^{-C_n(t-1)} \sum_{k=0}^{\hat{B}_n(t)} P_{U_n(t)}(k) P_{D_n(t)}(k-x) \quad (14c)$$

$$> \sum_{x=-\check{B}_n(t)}^{-(C_n(t-1)+1)} \sum_{k=0}^{\hat{B}_n(t)} P_{U_n(t)}(k) P_{D_n(t)}(k-x) \quad (14d)$$

$$= P(\varnothing|C_n(t-1)+1) \quad (14e)$$

When $C_n(t-1) = \check{B}_n(t)$, with Equation (11), we have:

$$P(\varnothing|C_n(t-1)) = 0 < P(\varnothing|C_n(t-1)-1) \quad (15)$$

Combining (14) and (15), we conclude that $P(\varnothing|C_n(t-1))$ is a monotonically decreasing function in the region $\{1, 2, \cdots, \check{B}_n(t)\}$. With bracketing and binary searching, it is thus guaranteed to find the optimal $C_n(t-1)$ meeting the blocking probability constraint, as long as the problem has a feasible solution. □

B. Computational Complexity Analysis

Although, theoretically algorithm 1 is guaranteed to find the minimum volume of resources to be provisioned at each edge cloud in each future time slot, it still matters whether the algorithm is efficient enough to obtain the optimum.

We then analyze the computational complexity when solving the problem, especially in computing the value of $P(\varnothing|C_H)$. To be specific, Equations (5) and (7) (i.e., the PDFs of Poisson processes) have factorial terms and therefore can be expensive to compute, especially when the size of the connected vehicle fleet is large. Nevertheless, we have noticed that, the Poisson probability decreases dramatically with the increase of the two RVs $D_n(t)$ and $U_n(t)$. Particularly, for a Poisson process with the mean value of 15, the probability value of arrivals exceeding 40 is almost negligible, i.e., $P(x > 40) \approx 0$. Thus, we can ignore those input values larger than a pre-defined threshold.

Overall, the optimal edge resource provisioning problem has a computational complexity of $\mathcal{O}(t_1 t_2 \log m_0^*)$, where:

- t_1 is the time consumed to compute i) arrival/departure rate ($\lambda_n(t)/\mu_n(t)$) for the vehicle arrival/departure flow by Equations (3) and (4), and ii) the total number of vehicle arrivals/departures ($\hat{B}/\check{B}$) in determining the parameter bounds of two Poisson processes by Equations (6) and (8), which all amount to simple summation operations in short time periods;
- t_2 is the time for computing the probability $P(\varnothing|C_n(t-1))$ in Equation (11), which can be solved efficiently via the given method introduced at the beginning of this subsection;
- $\log m_0^*$ indicates the number of iterations in the two-phase algorithm described above, where m_0^* is the obtained optimal value, which depends on the number of vehicles travelling into cell n.

According to our experiments, the algorithm has a very low computational complexity. As will be shown in Sec. V-B, our method can easily handle three-minute-ahead resource provisioning for over $50,000$ vehicles running across the target area of 100 cells (edge clouds).

C. Algorithm Applicability

During the working period of connected vehicle fleet, e.g., 8:00 am to 6:00 pm, the specified vehicular service is running as a background program at each edge cloud. Under the MEC paradigm and our resource provisioning scheme, the background program mainly conducts three tasks: (i) receive/send necessary information (e.g., vehicle locations and fleet distribution) from/to edge clouds in neighboring cells, (ii) compute the optimal resources provisioning solution with our algorithm, and (iii) complete the resource allocation at the edge cloud, e.g., by applying resource virtualization technology in VMs or containers. In supporting the MEC-enabled vehicular service, we regard the resources consumed by the background program as default/basic configuration at each edge cloud and thus do not take them into consideration during our optimal resource provisioning process.

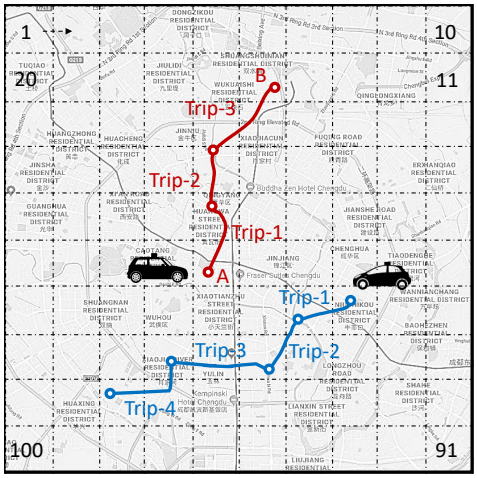


Fig. 5. Target area of our experiment with DATASET-I: the downtown area of a city divided into 10-by-10 blocks. Two example taxi trajectories are illustrated on the map.

VI. Experimental Evaluations

In this section, we first introduce the experimental setup and evaluation methodology. Then, both overall and detailed performance analysis are made, by comparing our method with two typical benchmarks. Furthermore, the investigations on stability and time complexity of our method are also made.

A. Experimental Setup

We use two real-world datasets for evaluations, which were collected by a large online taxi service platform [30]. All the taxis over the platform can be regarded as a special type of connected vehicle fleet for passenger delivery.

Datasets: The two datasets include (1) trajectory records of $40,770$ taxis and the order records of $150,412$ passengers (DATASET-I), and (2) trajectory records of $18,000$ taxis and the order records of $117,569$ passengers (DATASET-II), in two metropolises of China, respectively, both from Nov. 1, 2016 to Nov. 30, 2016. In more detail:

- A taxi trajectory record consists of the taxi ID, the IDs of the orders it takes, and the time stamp with corresponding location, with a sampling rate of 3 seconds.
- A passenger order record consists of the order ID, the departure time and corresponding location, the arrival time and corresponding location (of destination).

Preprocess: We choose a target area with the densest passenger orders in the city for each of the datasets. As an illustration, the area for DATASET-I is shown in Fig. 5. Then, we divide the target areas into 100 blocks, each representing one cell of the mobile cellular network. By observing the working time of most taxi drivers every day from the datasets, we select the time interval between 6:00 am to 0:00 am next day (18 hours) as the working time period. There is a tradeoff in choosing an appropriate length of time slot. On the one hand, a shorter time slot could result in more accurate estimations of vehicle arrivals/departures in one cell. On the other hand, the time slot should be long enough so that our algorithm (for computing the optimal resources to be provisioned) can output the solution one time slot ahead. The

TABLE I
OVERALL RESULTS FROM OUR METHOD AND THE TWO BENCHMARKS

		Benchmark-1	Benchmark-2	Ours ($\epsilon = 0.01$)	Ours ($\epsilon = 0.02$)	Ours ($\epsilon = 0.05$)
DATASET-I ($\#vehicles = 40,770$)	aca	100%	96.8%	99.3%	98.5%	96.1%
	tpc	8.96×10^4	5.34×10^4	5.63×10^4	5.50×10^4	5.26×10^4
	cpv	2.20	1.31	1.38	1.35	1.29
DATASET-II ($\#vehicles = 18,000$)	aca	100%	95.8%	99.2%	98.1%	95.9%
	tpc	3.87×10^4	2.32×10^4	2.43×10^4	2.39×10^4	2.34×10^4
	cpv	2.15	1.29	1.35	1.33	1.30

TABLE II
PROPERTIES OF DATASETS AND PARAMETER SETTINGS

	Properties	*Values*
DATASET-I	Target area	156 km^2
	Time period	2016.11.1 - 2016.11.30
	Number of taxis	40,770
	Number of orders	150,412
	Sample rate	3 seconds
DATASET-II	Target area	63 km^2
	Time period	2016.11.1 - 2016.11.30
	Number of taxis	18,000
	Number of orders	117,569
	Sample rate	3 seconds
Preprocessing	Number of cells	100
	eNBs density	0.64/km^2 (I); 1.59/km^2 (II)
	Working period	6 : 00 am - 00 : 00 am (+1)
	Time slot length	3 minutes

3-minute time slot is set empirically to balance the tradeoff. Thus, we divide the 18 hours into 360 equal-length time slots (each time slot is 3 minutes), and re-sample the trajectory data with a sampling interval of 3 minutes.

The detailed information about the datasets and parameter settings is summarized in Table II.

B. Evaluation Methodology

To validate the effectiveness of our method, we implemented the following two benchmarks for resource provisioning and compared their performance with ours:

- Benchmark-1 (naïve provisioning): allocate resource beforehand; once the path of a vehicle is determined, the resources are provisioned at all cells along the path.
- Benchmark-2 (mobility estimation based provisioning): provision resources at the most likely cell where a vehicle might appear for the next time slot, based on the vehicle's current status of mobility, e.g., location, (estimated) speed and moving direction. Such a principle was used in [22], [23] in predicting future vehicle locations, and an equivalent idea was also adopted in [28] by minimizing the average latency for optimal service placement.

It studied a user mobility aware service placement problem under the MEC environment, and the concept of "service" in this work took a similar role as the "resource" in ours. To find the optimal service placement solution, the authors constructed an optimization problem to minimize the average service latency. This optimization purpose is actually equivalent to that of Benchmark-2 given in our work, i.e., to determine the most likely cell where a vehicle might appear in such that the average service delay can be minimized.

Note that the complete route information of each vehicle is treated known for both the two benchmarks and our method. What make the three methods different are their ways of applying the route information in provisioning edge resources. Specifically, for Benchmark-1, it allocates (redundant) resources for all edge clouds along the whole path of the route. In Benchmark-2, it estimates the most likely cell that a vehicle may enter in the following time slot and only provisions (one unit of) resource for the edge cloud there. With our method, for each vehicle, we provision resources among the multiple cells that it probably appears based on both the vehicle's arrival/departure time PDFs and the pre-defined QoS threshold.

Our evaluation methodology is as follows. First, we use the taxi trajectory data to populate the fleet model ($S(t)$ and $H(t)$) at each time slot. Referring to the order information that discloses the routing paths, we are able to estimate the travel time distribution of vehicles between cells. We performed statistical tests on the travel times, and found that the travel time distribution can be modelled with a normal distribution, with the mean value representing the most likely travelling distance (for inferring possible destination cells) and the variance indicating the uncertainty of the estimation. Then, in each time slot, we compute the resource demand of each cell for the next time slot, using our method and the benchmark methods. Finally, with our provisioned resources in each cell as well as the vehicle trajectory data, we can compute the actual blocking probability (denoted by ϵ') of the fleet over the whole working time period. To guarantee sufficient resources to be provisioned, we set the capacity of each edge cloud to the number of vehicles in the fleet, i.e., the maximum possible resource requests from the vehicle fleet (when all vehicles appear in the same cell).

We apply the following metrics to evaluate and compare the performance of different methods:

- Actual resources availability (aca):

$$aca := 1 - \epsilon', \tag{16}$$

  where ϵ' is the actual blocking probability during the running of the whole fleet, with a certain resource provisioning method applied beforehand.
- Total provisioning cost (tpc):

$$tpc := \sum_{t=1}^{T} \sum_{n=1}^{N} C_n(t), \tag{17}$$

  where $C_n(t)$ is the provisioned volume of resources at cell n in time slot t. Thus, this metric reflects the total cost of resource provisioning for all the N cells during the whole working period.

- Volume of resources (units) per vehicle (cpv) of the fleet per time slot:

$$cpv := \frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{n=1}^{N} C_n(t)}{\sum_{n=1}^{N} S_n(t)}, \tag{18}$$

where $C_n(t)$ is the provisioned volume of resources at cell n in time slot t and $S_n(t)$ is the number of vehicles localized in cell n in time slot t. This metric indicates the average units of resources that assigned to each vehicle during the whole working time.

Note that the value of cpv can be derived from that of tpc. With the ideal value equal to 1 (one vehicle is at least in need of one unit of resource for QoS guarantee), cpv helps compare the performances of different resource provisioning methods.

C. Performance Results

1) Overall Performance Analysis

For the whole fleet: We set the QoS thresholds to $\epsilon = 0.01, 0.02$ and 0.05, i.e., the probability of resource availability is required to be no less than $99\%, 98\%$ and 95%, respectively[4]. Then we perform the 3-min-ahead resource provisioning at the 100 cells for all vehicles in each day. The overall results from the two datasets are shown in Table I. From the results, we can see that:

- The naïve provisioning strategy (benchmark-1) and our method can achieve the required QoS goal with guarantee, while the mobility estimation based method (benchmark-2) cannot. Compared with benchmark-1, our method can achieve the QoS goal using much smaller amount of edge resources.
- Our method and benchmark-2 consume much less resources than benchmark-1. For example, for the case of $\epsilon = 0.01$, our method and benchmark-2 are with overall cost/resource reductions of i) 37.3% and 40.5% under dataset-I, and ii) 37.2% and 40% under dataset-II, respectively.

From all the above results, we can conclude that our method can much reduce the resource provision cost while still ensures the QoS guarantee.

For subsets of the fleet: To gain more insights on the overall performance, we randomly select 10 subsets of the two fleets (from the two datasets), each subset with $1000, 2000, \cdots, 10,000$ vehicles, respectively. Then, we perform resource provisioning for these vehicle subsets, under the QoS requirement of $\epsilon = 0.01$. The performance results from the two datasets for different vehicle subsets are shown in Fig. 6 and Fig. 7, respectively. From the results, we can further observe that:

- The QoS requirement with our method can be satisfied in all cases, indicating the reliability of our method.
- The benchmark-2 cannot achieve aca above the required value of 99%.
- The maximum reduction of tpc with our method can reach i) 40% for DATASET-I under the $10,000$-vehicle case, and ii) 41% for DATASET-II under the $3,000$-vehicle case, respectively.
- The minimum value of cpv with our method can reach i) 1.35 for DATASET-I under the $7,000$-vehicle case, and ii) 1.31 for DATASET-II under the $3,000$-vehicle case, respectively.

From the viewpoint of total provisioning cost, the advantage of our method and benchmark-2 (the mobility estimation based method) over benchmark-1 (the naïve provisioning strategy) is obvious. From the viewpoint of QoS guarantee, our method and benchmark-1 outperform benchmark-2. Nevertheless, the average aca value for benchmark-2 is above 95% as shown in Table I. This seems to suggest that benchmark-2 would be a good method under the QoS requirement $\epsilon = 0.05$. Unfortunately, this is not true, as disclosed in our next experiment.

2) Detailed Performance Analysis

To further look into the performance of actual resource availability, especially that of benchmark-2, we sample one day's data (a Tuesday from DATASET-I as an example) and perform resource provision for all vehicles. We record the intermediate values of aca for each time slot and illustrate their values from the three methods in Fig. 8. In particular, we investigate the time period from 8:00 am to 6:00 pm, which includes two traffic peaks in a day. As we can see from the figure, our method can always satisfy the QoS requirement ($\epsilon = 0.01$) in each time slot, as it performs resource provision optimization for each next time slot under the condition of blocking probability no less than ϵ.

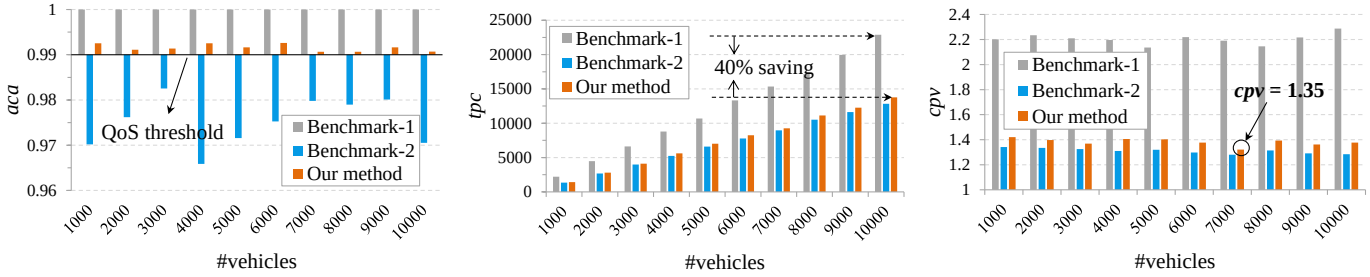
In contrast, the aca resulted from benchmark-2 varies largely along the timeline, with quite low values during traffic peak time in the morning (90.4% around 8:40 am) and the traffic peak time in the afternoon (90.8% around 5:50 pm), and with relatively high values during other times. This phenomenon is caused by inaccurate mobility estimation and incorrect cell determination during the busy traffic times, due to the complex traffic conditions and uncertain driving behaviors. Different from benchmark-2 that only selects one cell for resource provisioning, our method looks into the distribution of possible cells and choose (one or multiple) cells for resource provisioning from a probabilistic perspective.

Notably, with the lowest aca value, benchmark-2 fails to serve nearly 10% of the whole vehicles in the fleet, which is not acceptable in practice. For the same traffic condition, the QoS provided by our method is almost one order higher than that of benchmark-2, as the actual blocking probability with our method is guaranteed to be less than 1%.

3) Stability Analysis

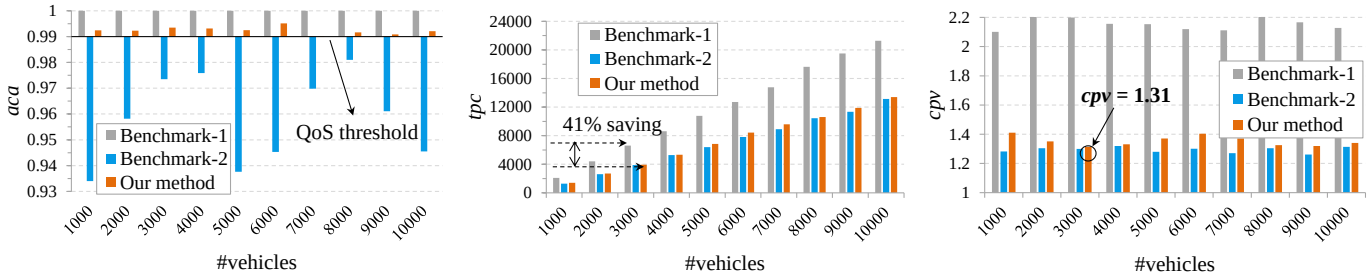
From the results in scenarios where the number of vehicles varies (Figs. 6 and 7) and the results where the traffic conditions change over time (Fig. 8), we can easily observe that our method consistently results in stable performance outcomes w.r.t aca and cpv. In contrast, benchmark-2 has a high variance in aca, as shown in Fig. 8. While the performance of benchmark-1 is also stable, such stability is obtained with the high cost of over provisioning.

[4]The values of QoS thresholds here are mainly for testing and validation purposes. The values should be set higher, e.g., 99.99% for mission critical applications such as autonomous driving.



(a) Actual resource availability (aca). (b) Total provisioning cost (tpc). (c) Resources per vehicle (cpv).

Fig. 6. Performance of vehicle subgroups with different number of vehicles (DATASET-I), with the required QoS $\epsilon = 0.01$.



(a) Actual resource availability (aca). (b) Total provisioning cost (tpc). (c) Resources per vehicle (cpv).

Fig. 7. Performance of vehicle subgroups with different number of vehicles (DATASET-II), with the required QoS $\epsilon = 0.01$.

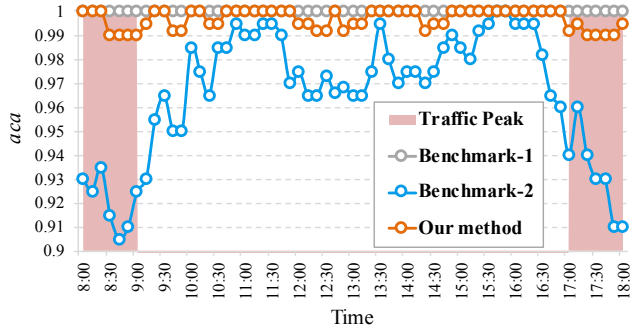


Fig. 8. Detailed results of the actual resource availability of the whole fleet during one day from 8:00 am to 6:00 pm, with the QoS requirement $\epsilon = 0.01$. The traffic peak times (morning peak and afternoon peak) are marked by shaded areas, which are based on the historical traffic data from *TomTom* for Chengdu on Tuesdays [31].

To further test the performance of benchmark-2 (the mobility estimation based provisioning), we update the mobility estimation strategy by: i) replacing the one most likely destination cell of the vehicles with the k most likely cells ($k \geq 1$), and ii) performing a parametric analysis for the value of k, with respective to its impact on the actual resource availability.

From the parametric analysis, we find similar outcome to that in Fig. 8, i.e., the obtained aca is unstable and the performance cannot be guaranteed. To be specific, as the value of k increases, we observe an overall upward shift of the aca curve; however, there are always time instances when the aca values are below the required QoS, until k grows up to a large value with a much increased resource provisioning cost ($cpv = 3.0$ in our test). This further validates the weakness of benchmark-2 in making trade-off between QoS requirement and provisioning cost.

4) Overhead Analysis

In this experiment, we investigate the running time of our method. When we apply the one-time-slot-ahead resource provisioning, we should make sure that the computing task in finding the solution can be finished within one time slot[5]. In other words, if the resource provision solution cannot be found within one time slot, our method would fail in practice even though the resulted solution is theoretically correct.

Fig. 9 shows the running times in solving the problem of (13) with our proposed two-phase algorithm given in Sec. V, performed on a low-end machine (64-bit Windows OS with 3.4-GHz CPU and 8-GB RAM). From the results shown in the figure, we can conclude that: i) our algorithm can effectively solve the problem of resource provisioning for $40,770$ vehicles, the maximum number of vehicles in the system under study, in 101.70 second, ii) if assuming the same trend of running time vs the number of vehicles, our method would be able to handle about $60,000$ vehicles, within a time slot of 3 minutes (i.e., 180 seconds), which should be large enough in practice.

VII. Conclusions

In this paper, we investigated the problem of optimal edge cloud resource provisioning for connected vehicle fleets under given QoS (service availability) requirements. With the commonly used virtualization technology at the distributed edge clouds, our goal is two-fold: i) to minimize the total resource provisioning cost among the edge clouds, and ii) to ensure the

[5]Actually there are more to be finished within the time slot, such as (vehicle trajectory) data collection and (edge cloud) resource re-configuration. These tasks, which should be cooperated with our resource provisioning strategy in a real-time fashion, fall into other research topics and beyond the scope of this paper.

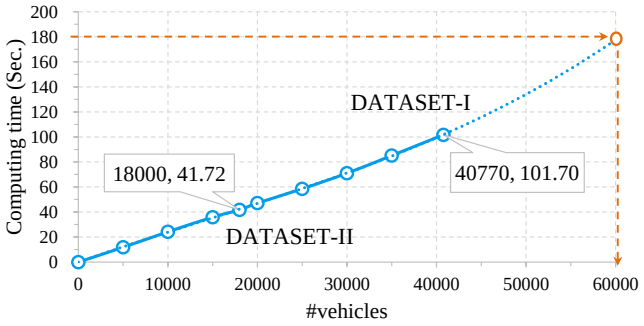


Fig. 9. The running time by applying our method along with the increase of fleet scale. With a 3-min time slot, our method can handle the maximum possible number of vehicles $(40,770)$ in the system under study.

predefined QoS of vehicular service all the time. To address this problem, we defined it formally from a probabilistic perspective. Then, with the stochastic analysis on vehicle traffic flows, we established an optimization model for optimal resource provisioning with the constraint on the service blocking probability that is smaller than a predefined threshold. An efficient algorithm was used to tackle the optimization problem based on bracketing and binary searching. We use two real-world dataset containing $58,770$ vehicles trajectory information in one month to evaluate our solution. With extensive experiments, we demonstrated that our method could save the total provisioning cost up to 40% compared with the naïve resource provisioning strategy; meanwhile our method could always provide reliable QoS guarantee, compared with the mobility estimation based method. Currently in this work, the evaluation was based on two datasets from an online taxi service platform. In the future, the proposed approach may need to be further examined in more general scenarios.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 61802421, No. U19B2024), the Natural Sciences and Engineering Research Council of Canada (No. RGPIN-2018-03896), the China Postdoctoral Science Foundation (No. 2019M663017), the Hunan Provincial Natural Science Foundation for Excellent Young Scholars (No. 2019JJ30029), and the project of "FANet: PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications" (No. LZC0019).

References

[1] 5G Automotive Association, "Toward fully connected vehicles: Edge computing for advanced automotive communications," Tech. Rep., 2017.

[2] Statista, 2018. [Online]. Available: https://www.statista.com/

[3] M. Fitzsimmons, "Waymo: everything you need to know from google until now," 2018. [Online]. Available: https://www.techradar.com/news/waymo

[4] A. J. Hawkins, "Lyft & aptiv extend their self-driving taxi pilot in las vegas," 2018. [Online]. Available: https://www.theverge.com/2018/1/22/16919446/lyft-aptiv-self-driving-taxi-pilot-extend-las-vegas

[5] J. Bhuiyan, "Gm's cruise is launching self-driving pilots in cities because that's where the money is," 2017. [Online]. Available: https://www.recode.net/2017/12/24/16816258/gm-cruise-self-driving-pilot-cities

[6] S. Dai and C. Chen, "Jd.com unveils self-driving truck in move to automate logistics operations," 2018. [Online]. Available: https://www.scmp.com/tech/innovation/article/2148420/jdcom-unveils-self-driving-truck-move-automate-logistics-operations

[7] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "Avr: Augmented vehicular reality," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2018, pp. 81–95.

[8] A. Fox, B. V. Kumar, and F. Bai, "Multi-source variable-rate sampled signal reconstructions in vehicular cps," in *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.

[9] European Telecommunications Standards Institute (ETSI), "Etsi multi-access edge computing starts second phase and renews leadership team," Tech. Rep., 2017.

[10] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537–1562, 2019.

[11] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[12] K. HYATT and C. PAUKERT, "Self-driving cars: A level-by-level explainer of autonomous vehicles," 2018. [Online]. Available: https://www.cnet.com/roadshow/news/self-driving-car-guide-autonomous-explanation/

[13] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2014, pp. 25–34.

[14] Y. Zhang, A. Haghani, and X. Zeng, "Component garch models to account for seasonal patterns and uncertainties in travel-time prediction," *IEEE Transactions on Intelligent Transportation Systems (TITS)*, vol. 16, no. 2, pp. 719–729, 2015.

[15] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das, "Predictive methods for improved vehicular wifi access," in *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2009, pp. 263–276.

[16] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2008, pp. 46–57.

[17] A. Gaddah and T. Kunz, "Extending mobility to publish/subscribe systems using a pro-active caching approach," *Mobile Information Systems*, vol. 6, no. 4, pp. 293–324, 2010.

[18] S. Pack, H. Jung, T. Kwon, and Y. Choi, "Snc:a selective neighbor caching scheme for fast handoff in ieee 802.11 wireless networks," *ACM Sigmobile Mobile Computing & Communications Review*, vol. 9, no. 4, pp. 39–49, 2005.

[19] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *IEEE International Conference on Edge Computing (EDGE)*, 2017, pp. 47–54.

[20] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking (TON)*, vol. PP, no. 99, pp. 1–14, 2017.

[21] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis, "Proactive selective neighbor caching for enhancing mobility support in information-centric networks," in *ACM Conference on Information-Centric Networking (ICN)*. ACM, 2012, pp. 61–66.

[22] M. Kim, D. Kotz, and S. Kim, "Extracting a mobility model from real user traces," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–13.

[23] J. Yoon, B. D. Noble, M. Liu, and M. Kim, "Building realistic mobility models from coarse-grained traces," in *International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2006, pp. 177–190.

[24] L. Song, X. He, X. He, and X. He, "Evaluating location predictors with extensive wi-fi mobility data," *ACM Sigmobile Mobile Computing & Communications Review*, vol. 7, no. 4, pp. 64–65, 2003.

[25] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing (TCC)*, vol. 7, no. 2, pp. 369–382, 2019.

[26] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[27] L. Zhao, J. Wang, J. Liu, and N. Kato, "Optimal edge resource allocation in iot-based smart cities," *IEEE Network*, vol. 33, no. 2, pp. 30–35, 2019.

[28] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 36, no. 10, pp. 2333–2345, 2018.

[29] T. Carpenter, S. Keshav, and J. Wong, "Sizing finite-population vehicle pools," *IEEE Transactions on Intelligent Transportation Systems (TITS)*, vol. 15, no. 3, pp. 1134–1144, 2014.
[30] DiDi Chuxing, "Gaia open dataset: Open collaborative innovative," 2018. [Online]. Available: https://outreach.didichuxing.com/research/opendata/en/
[31] TomTom, "Chengdu traffic - weekly traffic congestion by time of day," 2018. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/chengdu-traffic

Guoming Tang is currently a research fellow at the Peng Cheng Laboratory, Shenzhen, Guangdong, China. He received his Ph.D. degree in Computer Science from the University of Victoria, Canada, in 2017, and the Bachelor's and Master's degrees from the National University of Defense Technology, China, in 2010 and 2012, respectively. He was also a visiting research scholar of the University of Waterloo, Canada, in 2016. His research mainly focuses on computational sustainability, edge computing and intelligent transportation systems.

Deke Guo received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of Systems Engineering, National University of Defense Technology. His research interests include distributed systems, software-defined networking, data center networking, and wireless and mobile systems.

Kui Wu received the BSc and the MSc degrees in computer science from the Wuhan University, China, in 1990 and 1993, respectively, and the PhD degree in computing science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science, University of Victoria, Canada, in 2002, where he is currently a Full Professor. His research interests include smart grid, mobile and wireless networks, and network performance evaluation. He is a senior member of the IEEE.

Fang Liu is an Associate Professor at the School of Data and Computer Science, Sun Yat-Sen University (SYSU), China. She received the B.S. and PhD degrees in computer science from National University of Defense Technology, Changsha, China in 1999 and 2005, respectively. Her main research interests include computer architecture, edge computing and storage system.

Yudong Qin received the B.S. degree and M.S. degree in management science and engineering from National University of Defense Technology, Changsha, China, in 2016 and 2018, respectively. He is currently working toward his PhD degree in the School of Computer at the National University of Defense Technology. His research interests include software defined network and network security.