Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing

Chao Zhu†, Giancarlo Pastor†, Yu Xiao†, Yong Li*, Antti Ylä-Jääski†
†Aalto University, Espoo, Finland
* Tsinghua University, Beijing, China
Email: †{chao.1.zhu, giancarlo.pastor, yu.xiao, antti.yla-jaaski}@aalto.fi, *liyong07@tsinghua.edu.cn

***Abstract*—Emerging vehicular applications, such as real-time situational awareness and cooperative lane change, demand for sufficient computing resources at the edge to conduct time-critical and data-intensive tasks. This paper proposes Fog Following Me (Folo), a novel solution for latency and quality balanced task allocation in vehicular fog computing. Folo is designed to support the mobility of vehicles, including ones generating tasks and the others serving as fog nodes. We formulate the process of task allocation across stationary and mobile fog nodes into a joint optimization problem, with constraints on service latency, quality loss, and fog capacity. As it is a NP-hard problem, we linearize it and solve it using Mixed Integer Linear Programming. To evaluate the effectiveness of Folo, we simulate the mobility of fog nodes at different times of day based on real-world taxi traces, and implement two representative tasks, including video streaming and real-time object recognition. Compared with naive and random fog node selection, the latency and quality balanced task allocation provided by Folo achieves higher performance. More specifically, Folo shortens the average service latency by up to 41% while reducing the quality loss by up to 60%.**

I. Introduction

Future vehicles are becoming more intelligent and fully connected. The white paper [1] published by 5G Automotive Association describes emerging vehicular applications, such as real-time situational awareness, high-definition local maps, and see-through for passing. These applications involve data-intensive and latency-sensitive computing tasks, such as pattern recognition [2], [3] and augmented reality (AR) [4], [5].

The cloud model is not applicable to environments where operations are latency-critical. For example, the prevention of collisions and accidents cannot afford the latency caused by the round trip between vehicle and remote cloud. To solve this issue, a new computing paradigm called fog computing [6] has been proposed. Its key idea is to push intelligence (e.g. computing resources, application services) to the edge where the data is being generated and acted upon [7]. In the rest of this paper, *fog nodes* refer to the computing nodes at the edge, while *Vehicular Fog Computing* (VFC) stands for fog computing for vehicular applications.

Due to the mobility of vehicles, the computing and communication workload generated by vehicular applications varies with time and location. Meanwhile, as proposed by Xiao et al. [8] and Satyanarayanan et al. [9], fog nodes in VFC scenarios can be either stationary or mobile ones. For example, commercial fleets with sufficient computing resources and network connectivity can be turned into mobile fog nodes to handle the tasks generated by neighboring vehicles and passengers. The mobility of fog nodes opens up new opportunities like on-demand computing [8]. However, it also adds a layer of complexity to the process of task allocation in vehicular fog computing.

Inspired by [10], we explore a novel dimension, Quality Loss of Results (QLR) to represent the user acquired service with lower or less-than-optimal quality compared with the perfect result. For the vehicular applications designed for improving driving safety and efficiency, they are supposed to satisfy certain constraints on service latency and quality loss. According to [10], service latency could be reduced to certain extent by relaxing the tolerance of quality loss. In this paper, we propose *Folo*, a dynamic task allocation solution that balances service latency and quality loss under constraints. We quantify the tolerance of quality loss with concrete levels of QLR, and formulate the process of task allocation across stationary and mobile fog nodes as a joint optimization problem. The objectives of optimization include minimizing the average service latency and reducing the overall quality loss. As it proves to be a NP-hard problem, we linearize it and solve it using Mixed Integer Linear Programing (MILP).

To evaluate the effectiveness of Folo, we use a set of real-world taxi traces collected in Shanghai city to simulate the mobility of fog nodes at different times of day, and implement two example tasks, including video streaming and real-time object recognition. We measure the resource consumption of the example tasks through real-world experiments, and analyze the performance of vehicle-to-fog communications using an inter-vehicle communication simulator. Compared with the existing solutions, such as naive and random fog node selection, Folo shortens the average service latency, reduces the overall quality loss, and achieves better balance between service latency and quality loss.

The key contributions of this work are summarized below:

- We develop Folo, a novel solution for latency and quality balanced task allocation across stationary and mobile fog nodes in VFC.
- We formulate the process of task allocation as a joint optimization problem, and solve it with MILP.
- We evaluate the effectiveness of Folo through simulation, using real-world application profiles and taxi traces as

This work was partially funded by Nokia Center for Advanced Research.

978-1-5386-4281-8/18/$31.00 ©2018 IEEE

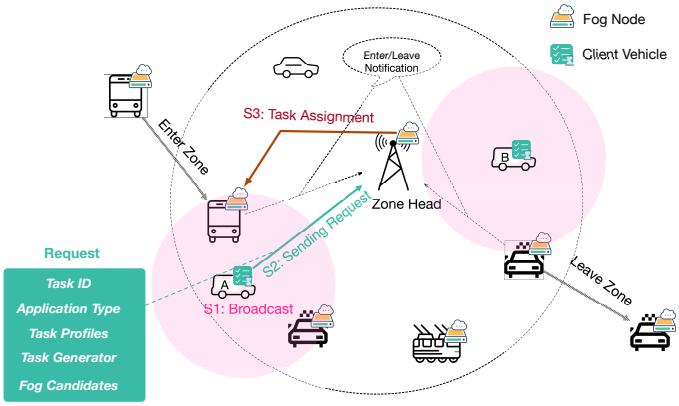


Fig. 1: Overview of Folo. A light magenta circle represents the communication range of the client vehicle in the center. Here the communication is limited to one hop for DSRC.

input. The results show that Folo outperforms the existing solutions in terms of service latency and quality.

The rest of the paper is organized as follows. Section II gives an overview of Folo. Section III describes the system model and constraints. The optimization problem is formulated and solved in Section IV. Section V explains the experiments for collecting the data to be used for evaluation in Section VI. Section VII discusses the related works before we conclude in Section VIII.

II. System Overview

In this section, we define the related terms and give an overview of the process of task allocation in Folo.

A. Related Terms

Fog Nodes In Folo, we consider two types of fog nodes.

* *Stationary Fog Nodes*: The computing nodes co-located with cellular base stations, Wi-Fi access points, road side units, or any other stationary infrastructure.
* *Mobile Fog Nodes*: The computing nodes carried by moving vehicles with on-board DSRC [11] and LTE communication modules.

Tasks The process of an application can be decomposed into a set of tasks. For example, AR-based driving assistance consists of tasks such as video streaming and object recognition. Each task has its own workload profiles, and latency and quality constraints. In Folo, task is considered to be the basic unit for task allocation. In other words, task cannot be divided into sub-tasks from the perspective of task allocation.

Client Vehicles Vehicles that generate tasks are defined as *client vehicles*. Each client vehicle may generate more than one task at the same time. Tasks from one client vehicle can be assigned to different fog nodes.

Service Zones It is safe to assume that urban areas in modern cities are fully covered by cellular networks. Similarly with [12], we divide an urban area into *Service Zones*, and select a stationary fog node within the zone to manage and coordinate all the fog nodes in the same zone. The coordinator is called *Zone Head*. To simplify the system model, we always select a LTE base station to be the zone head, and assume that mobile fog nodes are deployed on commercial fleets, such as buses and taxis. Mobile fog nodes always inform the zone head when they enter or leave the zone, utilizing the existing cellular registration mechanisms. Additionally, they report periodically their locations, moving directions, and available capacities to the zone head. Note that locations (and dynamics) of fog nodes, client vehicles, and base stations are transparent to Folo. A potential step forward would be to integrate stochastic geometry tools [13] to the optimization formulation to relieve simulations.

B. Process of Task Allocation

Figure 1 illustrates the process of task allocation in Folo. The whole process consists of 4 operations.

1) Discovering fog nodes: In the initial stage, a client vehicle needs to figure out which fog nodes are located within its communication range. It broadcasts one-hop probe messages over DSRC, and collects responses from fog nodes. Any fog nodes that respond are included in the list of *fog candidates*. In Figure 1, the fog candidates for the client vehicle A are the fog nodes within A's communication range.

2) Sending requests: After discovering fog candidates, the client vehicle sends a request to the zone head over LTE. The request contains information about the tasks to be offloaded to fog candidates.

* *Task ID*: The unique ID of a task.
* *Application Type*: The type of the application.
* *Task Profiles*: Description of the generated workload and the task-specific constraints, such as data size, tolerable latency and supported video resolutions.
* *Task Generator*: The client vehicle which generates data and sends the request.
* *Fog Candidates*: The fog nodes within the client vehicle's communication range.

3) Assigning tasks to fog candidates: When receiving a request from any client vehicle, the zone head executes the task allocation algorithm to decide where to run the tasks. We will present the details of the algorithm in Section IV.

4) Scheduling task migration: Due to the mobility of client vehicles and mobile fog nodes, the connection between them may not last until the assigned tasks complete. For example, the execution of a task may be interrupted when the corresponding fog node moves out from the current service zone. In that case, the zone head of the current service zone must call another fog node to take over the task.

III. System Modeling and Constraints

A. System Model

In this section, we present the system model of Folo. We define $\mathcal{K}_i$ as the set of tasks generated by the client vehicle i, and $\mathcal{J}_i$ as the set of fog candidates for the client vehicle i. We define a binary variable x_{ik} to indicate whether the task k

Notations	Definitions
$k, \mathcal{K}$	task index, set
$i, \mathcal{I}$	client vehicle index, set
$j, \mathcal{J}$	fog node index, set
$\mathcal{K}_i$	the set of tasks generated by client vehicle i
$\mathcal{J}_i$	the set of fog candidates available for client vehicle i
q_k	the QLR level of task k
$D(q_k)$	data size of task k with the QLR level equal to q_k
x_{jk}	whether task k is assigned to fog node j
x_{ij}	whether fog node j is available for client vehicle i
x_{ik}	whether task k is generated by client vehicle i
$R(q_k)$	the demand from task k with QLR level equal to q_k
$P(q_k)$	the processing delay of task k with QLR level equal to q_k
C_{ij}	data rate between client vehicle i and fog node j.
S_j	capacity of fog node j
τ_k	the maximum tolerable service latency of task k
ℓ	RTT overhead
$\mathcal{Q} = \{q_k\}$	the set of selected QLR levels
$\mathcal{X} = \{x_{jk}\}$	the set of selected fog nodes

TABLE I: Notations and definitions

is generated by i, and another binary variable x_{ij} to indicate whether fog node j is available for i.

$$x_{ik} = \begin{cases} 1, \ k \in \mathcal{K}_i \\ 0, \ Otherwise \end{cases}, \ x_{ij} = \begin{cases} 1, \ j \in \mathcal{J}_i \\ 0, \ Otherwise \end{cases} \tag{1}$$

Further, we define a binary variable x_{jk} to indicate whether task k is assigned to fog node j. The task k will be successfully assigned to fog node j only if fog node j is available for task k's generator. Hence,

$$\forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}, x_{jk} \leq \min\{x_{ik}, x_{ij}\} \tag{2}$$

1) Service Latency: Given limited bandwidth, the transmission delay depends on the size of data to be transmitted. For each task k, we define q_k as the level of QLR, and $D(q_k)$ as the corresponding size of data to be transmitted. We calculate the transmission delay T_k^{Comm} based on the transmission data rate C_{ij} of the link between the client vehicle i and the selected fog node j.

$$T_k^{Comm} = \sum_{i \in \mathcal{I}, j \in \mathcal{J}} \frac{D(q_k) x_{jk}}{C_{ij}} \tag{3}$$

We use $P(q_k)$ to denote the processing delay of task k with QLR equal to q_k, and calculate the processing latency T_k^{Proc} as follows:

$$T_k^{Proc} = P(q_k) \tag{4}$$

The service latency for each task consists of transmission delay, processing latency, and a constant overhead ℓ. ℓ captures the round trip time (RTT) between a client vehicle and a fog node. The service latency of each task k is formulated as follows:

$$T_k = T_k^{Comm} + T_k^{Proc} + \ell \tag{5}$$

B. Constraints

1) Quality Loss Constraint: The tolerance of quality loss is application specific. In Folo, we define 5 levels of QRL. Level 1 represents the strictest quality demand, while Level 5 refers to the highest tolerance for quality loss. The QLR constraint for task k can be modeled as follows.

$$\forall k \in \mathcal{K}, q_k \in \{1, 2, 3, 4, 5\} \tag{6}$$

In practice, q_k can be defined based on video resolution in case of video streaming.

2) Assignment Constraint: Task is supposed to be the basic unit for task allocation. Thus, it must be assigned as a whole to one fog node.

$$\forall k \in \mathcal{K}, \sum_{j \in \mathcal{J}} x_{jk} = 1 \tag{7}$$

3) Service Latency Constraint: The maximum tolerable service latency is determined by the type of application. According to measurements in [14], the maximum tolerable service latency of an AR navigation application is 250 ms and that of a video streaming application may reach 1 second. We use τ_k to denote the maximum tolerable service latency of task k. To guarantee that the task would be completed in time, a service constraint is given as follows:

$$\forall k \in \mathcal{K}, T(k) \leq \tau_k \tag{8}$$

4) Capacity Constraint: The demand for capacity (e.g. CPU, GPU, memory) is affected by the expected quality and service latency. The total demand from one fog node cannot exceed its capacity. We define S_j as the capacity of fog node j, and $R(q_k)$ as the demand from task k with QLR q_k. We formulate the capacity constraint as below.

$$\forall j \in \mathcal{J}, \sum_{k \in \mathcal{K}} R(q_k) x_{jk} \leq S_j \tag{9}$$

IV. Problem Formulation and Solution

A. Problem Formulation

We denote the maximum service latency of all tasks by $T = \max_{k \in \mathcal{K}}\{T_k\}$, and the summation of the QLR levels of all tasks by $Q^{sum} = \sum_{j \in \mathcal{J}, k \in \mathcal{K}} \{q_k x_{jk}\}$.

Folo aims at minimizing the maximum service latency T while minimizing the total quality loss Q^{sum}. However, these two objectives are coupled by q_k and cannot be optimized simultaneously. In the following, we investigate the trade-off between the two objectives and define the joint objectives function as $\varphi_t T + \varphi_q Q^{sum}$, where $\varphi_t, \varphi_q \in [0, 1]$ are two scalar weights.

We use $\mathcal{X} = \{x_{jk}\}$ to denote the set of selected fog nodes, and $\mathcal{Q} = \{q_k\}$ to denote the set of selected QLR levels. The optimization problem is formulated as:

$$\xi 1 : \min_{\mathcal{X}, \mathcal{Q}} \ \varphi_t T + \varphi_q Q^{sum} \tag{10}$$

s.t.

$$\forall j,k, x_{jk} \in \{0,1\}, q_k \in \{1,2,3,4,5\} \quad (10a)$$

$$\forall k, \sum_{i\in\mathcal{I},j\in\mathcal{J}} (\frac{D(q_k)}{C_{ij}} + P(q_k) + \ell)x_{jk} \leq \tau_k \quad (10b)$$

$$\forall j, \sum_{k\in\mathcal{K}} R(q_k)x_{jk} \leq S_j \quad (10c)$$

$$\forall k, \sum_{j\in\mathcal{J}} x_{jk} = 1 \quad (10d)$$

$$T = \max_{\forall k\in K}\{\sum_{i\in\mathcal{I},j\in\mathcal{J}} (\frac{D(q_k)}{C_{ij}} + P(q_k) + \ell)x_{jk}\} \quad (10e)$$

$$Q^{sum} = \sum_{j\in\mathcal{J},k\in\mathcal{K}} q_k x_{jk} \quad (10f)$$

$$\forall i,j,k, x_{jk} \leq \min\{x_{ij}, x_{ik}\} \quad (10g)$$

Proposition 1: $\xi 1$ is a NP-hard problem.

Proof 1: Consider a special case where $\varphi_t = 0, \varphi_q = 1$, which means the aim is to minimize the sum of QLR levels. Here we define $\bar{q}_k$ as the quality gain in the result of task k. It stands for the opposite of QLR. Hence, the objective of minimizing the sum of QLR levels can be transformed into maximizing the total quality gain. For simplification, we remove Constraint (10b). Additionally, we relax Constraint (10c) by assuming that the resource requirement of task k is exactly equal to its quality gain $\bar{q}_k$. Furthermore, we let one client vehicle generate only one task, and set one fog node in each service area. We define $\bar{x}_k$ to indicate whether the task k is assigned to the fog node, and S to denote the resource capacity of the fog node. Hence, we get a simplified optimization problem:

$$\xi 2 : \max \sum_{i\in I} \bar{q}_k \bar{x}_k \quad (11)$$

s.t.

$$\sum_{k\in\mathcal{K}} \bar{q}_k \bar{x}_k \leq S \quad (11a)$$

$$\bar{x}_k \in \{0,1\} \quad (11b)$$

Problem $\xi 2$ is a classic Subset Sum Problem, which has been proven to be a NP-complete problem [15]. Thus, we prove that Problem $\xi 1$ is a NP-hard problem.

B. Problem Linearizion

Problem $\xi 1$ is a non-linear optimization problem since Constraint (10b) and Constraint (10c) contain a production of two variables q_k and x_{jk}. To cast the optimization into linear programming (LP), we define $y_{jk} = q_k x_{jk}$ and use $\mathcal{Y} = \{y_{jk}\}$ to denote the set of variable y_{jk}. Furthermore, we relax the discrete variable q_k into continuous one, which is $q_k \in [1,5]$.

According to [10], we consider two linear approximate trade-off functions, $P(q_k) = a_t q_k + b_t$, and $R(q_k) = a_r q_k + b_r$. Additionally, we introduce a new variable t with an additional constraint $t \geq \max_{k\in\mathcal{K}} T_k$.

x_{jk}	q_k	$x_{jk}q_k$	**Constraints**	**Implication**
0	$0 \leq q_k \leq 5$	0	$y_{jk} \leq 0$ $y_{jk} \leq q_k$ $y_{jk} \geq q_k - 5$ $y_{jk} \geq 0$	$y_{jk} = 0$
1	$0 \leq q_k \leq 5$	q_k	$y_{jk} \leq 5$ $y_{jk} \leq q_k$ $y_{jk} \geq q_k$ $y_{jk} \geq 0$	$y_{jk} = q_k$

TABLE II: All possible values of y_{jk}

When $D(q_k) = a_d q_k + b_d, P(q_k) = a_t q_k + b_t, R(q_k) = a_r q_k + b_r$, we get a MILP problem that is equal to Problem 10:

$$\xi 3 : \min_{\mathcal{X},\mathcal{Q},\mathcal{Y},t} \ \varphi_t t + \varphi_q \sum_{j\in\mathcal{J},k\in\mathcal{K}} q_k x_{jk} \quad (12)$$

s.t.

$$\forall j,k, x_{jk} \in \{0,1\}, 1 \leq q_k \leq 5 \quad (12a)$$

$$\forall j,k, 0 \leq y_{jk} \leq 5x_{jk} \quad (12b)$$

$$\forall j,k, q_k - 5(1-x_{jk}) \leq y_{jk} \leq q_k \quad (12c)$$

$$\forall k, \sum_{i\in\mathcal{I},j\in\mathcal{J}} (\frac{b_d}{C_{ij}} + b_t + \ell)x_{jk} + (\frac{a_d}{C_{ij}} + a_t)y_{jk} \leq \tau_k \quad (12d)$$

$$\forall j, \sum_{k\in\mathcal{K}} b_r x_{jk} + a_r y_{jk} \leq S_j \quad (12e)$$

$$\forall k, \sum_{j\in\mathcal{J}} x_{jk} = 1 \quad (12f)$$

$$\forall k, \sum_{i\in\mathcal{I},j\in\mathcal{J}} (\frac{b_d}{C_{ij}} + b_t + \ell)x_{jk} + (\frac{a_d}{C_{ij}} + a_t)y_{jk} \leq t \quad (12g)$$

$$\forall i,j,k, x_{jk} \leq x_{ij}, x_{jk} \leq x_{ik} \quad (12h)$$

Proof 2: We prove that the Constraints (12b) and (12c) are equal to the constraint $y_{jk} = x_{jk}q_k$ by listing all the possible products in Table II.

C. Dynamic Task Allocation Algorithm

We propose an **D**ynamic **T**ask **A**llocation (**DTA**) algorithm and describe the detailed algorithm in Algorithm 1.

1) Initialization: In a service zone, the client vehicles which are located in the service zone generate tasks and send service requests to the zone head. The zone head would collect the information and add the tasks to the unassigned tasks set $\mathcal{U}$, as shown in Line 1. In Line 2, DTA generates an empty set $\mathcal{A}$ to load the tasks assigned by the zone head.

2) MILP Based Optimization: To execute the optimized task allocation, we propose a **M**ILP **B**ased **O**ptimization (**MBO**) algorithm and illustrate the workflow of MBO in Figure 2. The input of MBO is the unassigned tasks set, which contains the information about client vehicles set $\mathcal{I}$, fog nodes set $\mathcal{J}$, tasks set $\mathcal{I}$. Next, based on information of tasks and location of vehicles, the MBO formulates the optimization matrix, which contains the available fog nodes information x_{ij}, the host vehicle information x_{ik}, and the transmission data rate information C_{ij} between client vehicles and fog nodes. In the final stage, the MBO would take the end-to-end latency and

Algorithm 1 DTA: **D**ynamic **T**ask **A**llocation

Input: Traces of client vehicles and mobile fog nodes; Location of zone head; Unassigned task set $\mathcal{U}$

Output: Assignment decision set $\mathcal{X}$; QLR set $\mathcal{Q}$; Assigned task set $\mathcal{A}$

1: Initialize unassigned task set $\mathcal{U}$
2: Initialize assigned task set $\mathcal{A} = \oslash$
3: **while** $\mathcal{K} \neq \oslash$ **do**
4: Execute **MBO** for $k \in \mathcal{U}$, calculate $\mathcal{X}, \mathcal{Q}$
5: $t \rightarrow \mathcal{A}$, Remove k from $\mathcal{U}$
6: Transmit and process assigned task $t \in \mathcal{A}$
7: **switch** (**Events**)
8: **case** New task k^{new}**:**
9: $k^{new} \rightarrow \mathcal{U}$
10: **case** Service interrupted task k^{break}**:**
11: Remove k^{break} from $\mathcal{A}$, $k^{break} \rightarrow \mathcal{U}$
12: Execute **MBO** for migrating task k^{break}
13: **end switch**
14: Remove finished task k^{done} from $\mathcal{A}$
15: **end while**

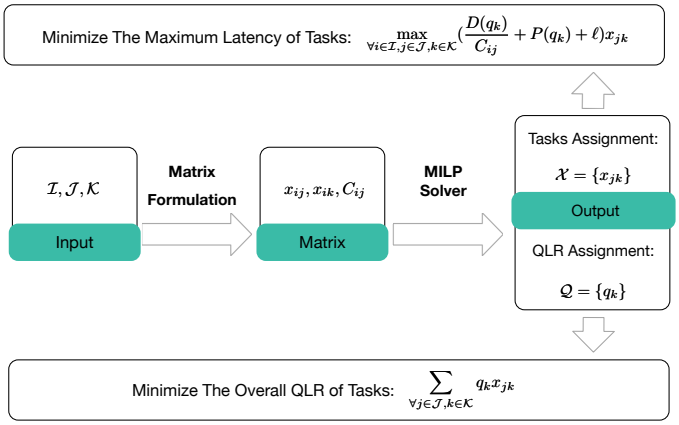


Fig. 2: Workflow of MBO

overall quality loss of tasks into consideration and implements a MILP solver to get the balanced optimization solution.

3) Event Handling: In Folo, the zone head detects events occurred in its service zone. In this paper, we consider two types of events:

- New Tasks: The zone head receives a new request from a client vehicle.
- Service Interruption: The connection between a client vehicle and a mobile fog node may break down, when they are moving towards different directions for example. We assume that mobile fog nodes keep monitoring the channel states and would report to the zone head when the disconnection is going to happen. The zone head will then find another fog node for the task to migrate to, as described from Line 11 to Line 12.

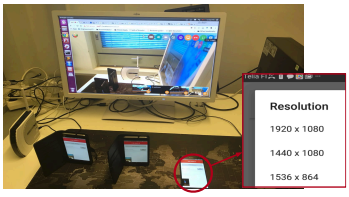
Video Streaming

Type	Server	Client		
Hardware	Desktop	Phone	Phone	Phone
OS	Linux	Android 7.0	Android 7.0	Android 7.0
Model	N.A.	Huawei Mate9	Huawei P10	Huawei P10
CPU	4x3.2GHz	4x2.36GHz	4x2.36GHz	4x2.36GHz
Memory	32GB	6GB	4GB	4GB

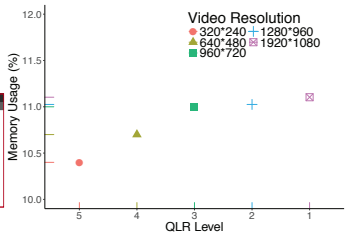
Object Recognition

Type	Server	Client
Hardware	LapTop	Web Camera
Model	HP-zbook G3	Logitech HD
GPU	Quadro M2000M	N.A.
CPU	8x2.7GHz	N.A.

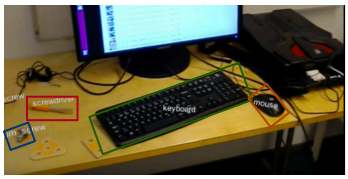
TABLE III: Experiments Hardwares



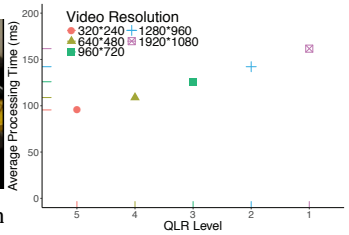
(a) Video Streaming



(b) QLR v.s. Memory consumption



(c) Real-time Object Recognition in AR mode



(d) QLR v.s. Processing Time

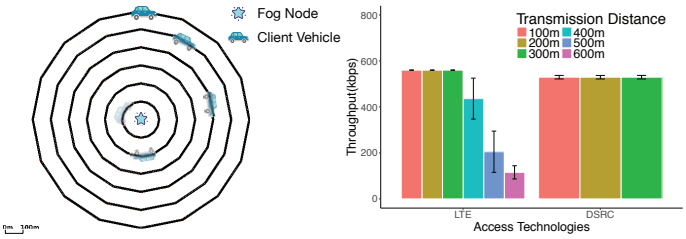
Fig. 3: Application Profiles

V. Experiments

To test Folo, we implemented two example tasks, namely, video streaming and real-time object recognition. We chose these two tasks because they are building blocks of many vehicular applications, such as AR-based driving assistance. In the experiments, we explore the impact of the variation of service quality on the service latency and the amount of resource consumption. This section describes the experiments for creating task profiles and analyzing the performance of vehicle-to-fog communication. The hardware devices used for experiment are listed in Table III. The experimental results are used later for configuring the simulator described in Section VI.

A. Application Profiling

We implemented video streaming based on *Kurento* [16], an open source platform for WebRTC-based real-time communications. As shown in Figure 3a, we ran the Kurento media server on a Linux desktop and a client application on three Android phones. The client application captures video and sends it to the media server. We measured the



(a) Concentric Circles Roads (b) Throughput

Fig. 4: Network performance of LTE vs. DSRC

CPU/GPU/memory usage of the media server while receiving video streams from the phones.

We tested 5 different video resolutions, {1920 * 1080, 1280 * 960, 960 * 720, 640 * 480, 320 * 240}. The frame rate of video streaming was limited to 14 fps, due to the hardware constraint. We define the QLR level of video streaming based on video resolution. The highest resolution corresponds to the lowest QLR level, and vice versa. As shown in Figure 3b, the memory usage increases with video resolution, and decreases nearly linearly with the QLR level. Based on the results, we build a linear model $R(q_k)$ to estimate the memory usage (in MB) based on the QLR level.

$$R(q_k) = -27.5 \times q_k + 247.5 \tag{13}$$

Given a fixed frame rate, the data size depends on the video resolution. We assume that the transmission video is compressed with the Youtube-HD standard. According to [17], we formulate the data size (in KB) of each frame $D(q_k)$ as follows:

$$D(q_k) = -7.7 \times q_k + 41.2 \tag{14}$$

In the second experiment, we implemented an AR-based object recognition application based on Yolo [18]. As show in the Figure 3c, the objects recognized from video streams are labeled in the camera view. Similarly with video streaming, we define the QLR level of this task also based on video resolution. In experiments, we measured the processing time taken to recognize objects from each frame. Figure 3d compares the processing time between 5 QLR levels. As the processing time has a nearly negative linear correlation with the QLR level, we formulate the processing time (in ms) per frame $P(q_k)$ as follows:

$$P(q_k) = -16.56 \times q_k + 176.5 \tag{15}$$

B. Network Performance

Dedicated Short-range Communication (DSRC) and LTE are the most popular vehicular networking technologies. DSRC is designed based on IEEE 802.11p. The data rate of DSRC can reach up to 27Mb/s with hundreds meters coverage [19]. Compared with DSRC, LTE has a much wider coverage and more deterministic quality of service (QoS) guarantees. According to the reports in [20], LTE can support User Equipments with high mobility at speed of 350 km/h.

In this paper, vehicles broadcast beacons over DSRC for mutual handshaking, such as detecting fog nodes and scheduling task migration. Additionally, DSRC channels are responsible for data transmission between client vehicles and mobile fog nodes. On the other hand, LTE is used for communications between client vehicles and the corresponding zone head (base station), such as sending requests and notifications of entering/leaving a service zone. The zone head itself is also a stationary fog node.

We simulate the scenarios of real-time video streaming using VeinsLTE [21]. VeinsLTE is an extension of Veins [22], which is an open source Inter-vehicle communication (IVC) simulator. VeinsLTE connects a microscope road traffic simulator SUMO [23] with a network simulation engine called OMNET++ through Traffic Control Interface (TraCI). With VeinsLTE, vehicles in simulation can either exchange data with each other through DSRC, or connect to base stations over LTE.

In SUMO, we build 6 near round concentric roads. Each road contains two lanes, and the distance between the neighboring roads is set to 100 meters. As shown in Figure 4a, we place one fog node in the center of the concentric roads, and add a video streaming module to the application layer of a client vehicle in VeinsLTE. The client vehicle moving at speed of 20m/s is continuously sending video data to the fog node in the center. By settling the client vehicle onto different roads, we could measure the throughput of video streaming with varying communication distance.

According to Figure 4b, the data rate in case of LTE remains stable when the communication distance is within 300 meters. When the distance exceeds 300 meters, the data rate decreases with the distance. Unlike LTE, the transmission range of DSRC is shorter than 300 meters, whereas single-hop DSRC provides stable performance. Based on the collected results, for the simulation described in Section VI, we set the default data rate of DSRC to be $500kbps$, and the data rates of LTE to be $\{550kbps, 450kbps, 200kbps, 150kbps\}$, depending on the channel state information. In addition, according to the measurements in [19], we set the RTT overhead to be 20ms for DSRC and 300ms for LTE. The other parameter values are listed in Table IV.

VI. Evaluation

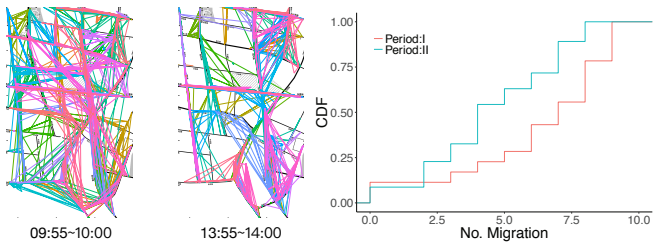
We implemented the task allocation strategies in Folo using Python. In particular, we utilized pyomo, a Python-based open source software package that supports a diverse set of optimization capabilities [24]. To evaluate the effectiveness of Folo, as explained in Section VI-A, we simulate the mobility of fog nodes based on real-world taxi traces, and configure the task profiles according to Table IV. After that, we compare the achieved service latency and quality loss between Folo and previous works in Section VI-B.

A. Vehicle Mobility and Task Generation

We use a set of real-world taxi traces to simulate the mobility of mobile fog nodes. The dataset contains the GPS

TABLE IV: Simulation Configuration

Parameters	Value	
$a_d, b_d(KB)$	-7.7, 41.2	
$a_r, b_r(MB)$	-27.5, 247.5	
$a_t, b_t(ms)$	-16.56, 176.5	
φ_t/φ_q	50, 100, 150, 450, 500	
Resolution/QLR	1920p/1, 1280p/2, 960p/3, 640p/4, 320p/5	
Frame Rate (fps)	14	
Delay Tolerance (s)	1[19]	
Last Time (s)	10	
	Mobile Fog Node	**Zone Head**
Capacity (GB)	0.5, 0.7, 0.9, 1.1, 1.3, 1.5	2.1
	DSRC	**LTE**
Range (m)	300	2000
Data Rate (kbps)	500	550, 450, 200, 150
RTT Overhead (ms)	20	300
No. Client Vehicles	100	



(a) Visualization of taxi traces (b) Mobility-related task migration

Fig. 5: Simulation of Mobile Fog Nodes

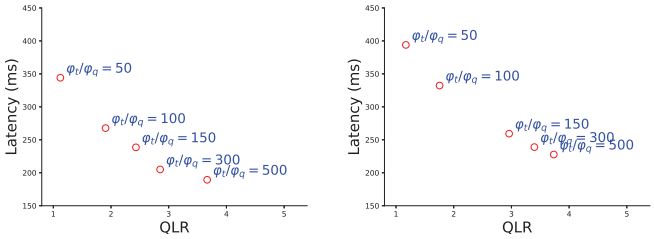
traces collected from April 13 to 30, 2015 in Shanghai city. It was collected by the iData Laboratory of Tongji University [25]. To evaluate the impact of the density of mobile fog nodes, we select an area of 4 km^2 acreage near Shanghai Pudong Airport, and collect the traces within the area during two time periods.

* *Time Period I*: $09:55 \sim 10:00$, April 20, 2015
* *Time Period II*: $13:55 \sim 14:00$, April 20, 2015

Time Period I belongs to rush hour. As visualized in Figure 5a, the density of taxis traces in Time Period I is higher. In details, 172 taxis appeared in the selected area during the first time period, compared with 120 taxis in the latter.

Besides the taxis, we generate another 100 vehicle routes in SUMO, following the method used in [26]. These vehicles act as client vehicles and generate video streams to be processed on fog nodes. Video streams are generated by randomly chosen client vehicles every second. A random number (not greater than 8) of video streams are generated every time, with each stream lasting for 10 seconds. Each run of our simulation lasts for 60 seconds. For example, in the first minute of Time Period I, 33 client vehicles generate 66 video streams, while 35 client vehicles are chosen to generate 70 video streams during the first minute of Time Period II.

When a client vehicle or a mobile fog node on duty is moving out of the current service zone, the ongoing tasks are supposed to be migrated to other fog nodes. Figure 5b shows the number of task migration occurred during Time Period I and II. According to the figure, task migration happens more often in Time Period I when the density of mobile fog node is higher. [1]



(a) Time Period I (b) Time Period II

Fig. 6: Service Performance v.s. Scalar Weight

B. Comparison between Task Allocation Strategies

As mentioned in Section III, the scalar weight φ_t and φ_q represent the optimization tendency toward service latency and quality, respectively. The task allocation strategy is latency sensitive when φ_t/φ_q is higher. Otherwise, it is quality sensitive. We tune the ratio of the scalar weights, φ_t/φ_q, from 50 to 500, and compare the results between the two time periods in Figure 6.

When the density of mobile fog node is high, as illustrated in Figure 6a, the average service latency is around 350 ms when the QLR level is 1. The service latency decreases with the tolerance of quality loss. For example, when the QLR level increases to 4, the average latency would drop to 200ms. Compared with Figure 6b where less mobile fog nodes are available, the service latency in Figure 6a is on average 50ms shorter.

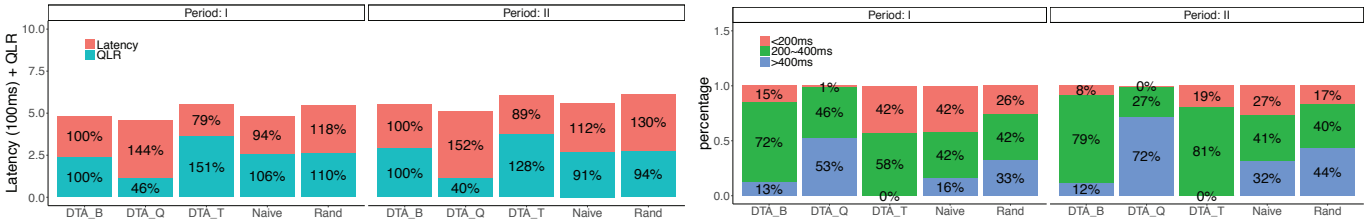
According to Figure 6, we define 3 versions of DTA based on the value of φ_t/φ_q.

* *DTA_Q*: QLR Sensitive DTA with $\varphi_t/\varphi_q = 50$.
* *DTA_T*: Latency Sensitive DTA with $\varphi_t/\varphi_q = 500$.
* *DTA_B*: Balanced DTA with $\varphi_t/\varphi_q = 150$.

For comparison, we also implemented two strategies, namely, *Rand* and *Naive*. These two have been presented in recent publications [27], [28]. *Rand* refers to *F.Rand.Assign* in [27]. It randomly selects one fog node from the available candidates. *Naive* always selects the fog node with the highest available data rate, which is close to *AVE+Naive* in [28]. In our experiments, we assume that the QLR level is randomly selected.

1) Service latency vs QLR: We compare the average service latency and QLR levels among the 5 different strategies in Figure 7a. We use the results of DTA_B as baseline. The lower the percentages are, the higher the performance is. Here performance is measured by service latency and quality. According to the results, DTA_T leads to the shortest service latency, while DTA_Q gains the highest quality. Compared with *Naive* which always chooses the fog node with the highest data transmission rate, DTA_T shortens the overall service latency by 15% by increasing the QLR level. Compared

[1] As the GPS fixes included in the taxi traces are sparse, the actual need for task migration could be less.



(a) Latency and QLR Performance

(b) Service Latency Distribution

Fig. 7: Performance of task allocation Strategies

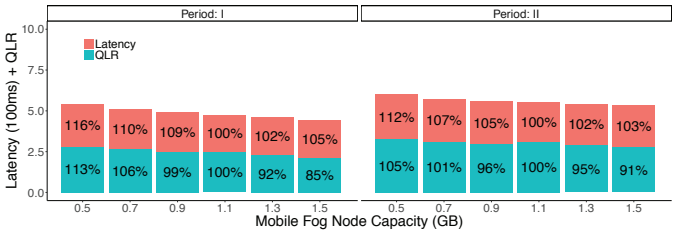


Fig. 8: Performance vs. Memory Capacity

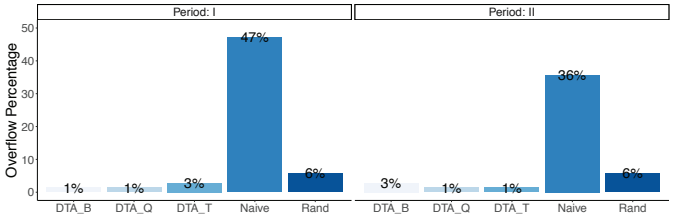


Fig. 9: Server Memory Overflow

with *Rand* and *Naive*, DTA_Q achieves higher quality, while DTA_B achieves better balance between service latency and quality loss. In general, these strategies also perform better when the density of mobile fog node is higher. The impact is less obvious in case of *Rand*, since *Rand* does not consider the impact of the vehicle density on the networking performance.

We also compare the distribution of service latency in Figure 7b. In case of DTA_B, most of tasks complete between 200ms and 400ms. In case of DTA_T, 42% of tasks are completed within 200ms when the density of mobile fog node is high. This is close to the result of *Naive*. Overall, DTA_T still outperforms *Naive*, since all tasks complete within 400ms when DTA_T is applied.

2) Memory Capacity vs. Performance: For the two tasks we tested, memory usage becomes a performance bottleneck. We notice that whenever the memory demand is satisfied, the CPU/GPU demand can be satisfied. Thus, we evaluate the performance with varying memory sizes. As shown in Table IV, the memory size of a mobile fog node is by default set to 1.1GB, and that of a zone head is 2.1GB.

Figure 9 illustrates the memory overflow issues. *Naive* ends up with serious memory overflow (up to 47%), as *Naive* chooses fog nodes based on network conditions alone. Compared with Rand, all the DTA-related strategies better balance the resource consumption, and keep the memory overflow under 3%.

We choose DTA_B as example to evaluate the service latency and QLR with different settings of memory capacity. We tune the memory size of each mobile fog node from 0.5 GB to 1.5 GB. As shown in Figure 8, when the memory capacity increases, both service latency and QLR decrease. In details, the service latency decreases by around 10%. Regarding the QLR, it decreases by up to 28% when the density of mobile fog node is high.

In summary, compared with *Rand* and *Naive*, DTA based strategies shorten the average service latency by up to 41% and QLR by up to 60%. Moreover, the availability of mobile fog nodes has positive impact on the service performance.

VII. Related Work

Fog computing shares with mobile edge computing [29] the same principle of moving computing resources to the edge. Different architectures of vehicular fog computing have been proposed in the literature. For instance, Satyanarayanan et al. [9] preferred to turn every vehicle into a fog node and to select a coordinator for each zone. Xiao et al. [8] proposed to turn commercial fleets into fog nodes to serve neighboring vehicles and passengers, while Hou et al. [30] suggested utilizing the extra computing power on slow moving or parked vehicles. In addition, Ni et al. [31] examined the architecture of fog-based vehicular crowdsensing with consideration of security, privacy, and fairness.

Relevant to our work, several previous works studied task allocation in fog/edge computing. Sardellitti et al. [32] jointly optimized radio and computational resources for multicell in edge computing. Liu et al. [33] considered the multi-task allocation problem for the edge environment with resource-intensive and latency-sensitive mobile applications. Dinh et al. [27] proposed an offloading framework to jointly minimize the tasks' execution latency and devices' energy consumption with consideration of CPU frequency. Li et al. [10] minimized service response time and energy consumption by jointly optimizing the QoR of all edge nodes and the offloading strategy. Deng et al. [34] investigated the tradeoff between power consumption and transmission delay in the fog-cloud computing system. However, these works cannot be directly applied to vehicular fog computing, as they did not consider

the mobility of vehicles. Feng et al. [28] proposed a framework for job caching in vehicular fog computing based on ant colony optimization algorithm. However, it focused on caching and did not consider other scenarios such as local processing. To the best of our knowledge, Folo is the first one that provides joint optimization of service latency and quality in vehicular fog computing, taking into account the mobility of fog nodes.

VIII. Conclusion

In this paper, we propose Folo, a dynamic task allocation solution for vehicular fog computing. It aims at minimizing average service latency while reducing the overall quality loss. We formulate the process of task allocation as a joint optimization problem, taking into account the constraints on service latency, quality loss, and fog node capacity. As it is a NP-hard problem, we simplify the problem through linearization, and solve it using MILP. We evaluate our solution with simulation. The simulation is configured based on real-world application profiles and mobility dataset. Compared with previous works, our solution reduces service latency by up to 41% and QLR by up to 60%.

References

[1] "Toward fully connected vehicles: Edge computing for advanced automotive communications 5g Automotive Association."

[2] S. Kumar, L. Shi, N. Ahmed, S. Gil, D. Katabi, and D. Rus, "CarSpeak: A Content-centric Network for Autonomous Driving," *SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 259–270, Aug. 2012.

[3] C. Tran, K. Bark, and V. Ng-Thow-Hing, "A Left-turn Driving Aid Using Projected Oncoming Vehicle Paths with Augmented Reality," in *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ser. AutomotiveUI '13. New York, NY, USA: ACM, 2013, pp. 300–307.

[4] P. Gomes, C. Olaverri-Monreal, and M. Ferreira, "Making Vehicles Transparent Through V2v Video Streaming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 930–938, Jun. 2012.

[5] H. Qiu, F. Ahmad, R. Govindan, M. Gruteser, F. Bai, and G. Kar, "Augmented Vehicular Reality: Enabling Extended Vision for Future Vehicles," in *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '17. New York, NY, USA: ACM, 2017, pp. 67–72.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16.

[7] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[8] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Mar. 2017, pp. 6–9.

[9] M. Satyanarayanan, "Edge computing for situational awareness," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Jun. 2017, pp. 1–6.

[10] Y. Li, Y. Chen, T. Lan, and G. Venkataramani, "MobiQoR: Pushing the Envelope of Mobile Edge Computing Via Quality-of-Result Optimization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 1261–1270.

[11] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[12] W. Hu, Z. Feng, Z. Chen, J. Harkes, P. Pillai, and M. Satyanarayanan, "Live Synthesis of Vehicle-Sourced Data Over 4g LTE," in *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2017.

[13] G. Pastor, I. Mora-Jiménez, A. J. Caamano, and R. Jäntti, "Medium access probability in uniform networks with general propagation models," in *Wireless Communication Systems (ISWCS 2013), Proceedings of the Tenth International Symposium on*. VDE, 2013, pp. 1–5.

[14] Y. Xiao, M. Noreikis, and A. Yla-Jaaski, "QoS-oriented capacity planning for edge computing," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[15] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, no. 1-3, pp. 181–199, Aug. 1994.

[16] L. Lpez, M. Pars, S. Carot, B. Garca, M. Gallego, F. Gortzar, R. Bentez, J. A. Santos, D. Fernndez, R. T. Vlad, I. Gracia, and F. J. Lpez, "Kurento: The WebRTC Modular Media Server," in *Proceedings of the 2016 ACM on Multimedia Conference*, ser. MM '16. New York, NY, USA: ACM, 2016, pp. 1187–1191.

[17] P. Forret, "Video filesize calculator | toolstudio."

[18] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv:1612.08242 [cs]*, Dec. 2016, arXiv: 1612.08242.

[19] Z. Xu, X. Li, X. Zhao, M. H. Zhang, and Z. Wang, "DSRC versus 4g-LTE for Connected Vehicle Applications: A Study on Field Experiments of Vehicular Communication Performance," *Journal of Advanced Transportation*, 2017, dOI: 10.1155/2017/2750452.

[20] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "LTE for vehicular networking: a survey," *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148–157, May 2013.

[21] F. Hagenauer, F. Dressler, and C. Sommer, "Poster: A simulator for heterogeneous vehicular networks," in *2014 IEEE Vehicular Networking Conference (VNC)*, Dec. 2014, pp. 185–186.

[22] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[23] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012.

[24] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in Python," *Mathematical Programming Computation*, vol. 3, no. 3, p. 219, Sep. 2011.

[25] "iData Laboratory."

[26] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research," in *2015 IEEE Vehicular Networking Conference (VNC)*, Dec. 2015, pp. 1–8.

[27] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[28] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous Vehicular Edge Computing Framework with ACO-Based Scheduling," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[29] P. Corcoran and S. K. Datta, "Mobile-Edge Computing and the Internet of Things for Consumers: Extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73–74, Oct. 2016.

[30] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, June 2016.

[31] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, Privacy, and Fairness in Fog-Based Vehicular Crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[32] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015, arXiv: 1412.8416.

[33] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2398–2410, Oct. 2016.

[34] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.