Zhu, Chao; Tao, Jin; Pastor Figueroa, Giancarlo; Xiao, Yu; Ji, Yusheng; Zhou, Quan; Li, Yong; Ylä-Jääski, Antti

Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing

Published in:
IEEE Internet of Things Journal

DOI:
10.1109/JIOT.2018.2875520

Published: 01/06/2019

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Zhu, C., Tao, J., Pastor Figueroa, G., Xiao, Y., Ji, Y., Zhou, Q., Li, Y., & Ylä-Jääski, A. (2019). Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing. *IEEE Internet of Things Journal*, *6*(3), 4150 - 4161. https://doi.org/10.1109/JIOT.2018.2875520

Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing

Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao*, Yusheng Ji, Quan Zhou, Yong Li and Antti Ylä-Jääski

Abstract—With the emerging vehicular applications such as real-time situational awareness and cooperative lane change, there exist huge demands for sufficient computing resources at the edge to conduct time-critical and data-intensive tasks. This paper proposes Folo, a novel solution for latency and quality optimized task allocation in Vehicular Fog Computing (VFC). Folo is designed to support the mobility of vehicles, including vehicles that generate tasks and the others that serve as fog nodes. Considering constraints on service latency, quality loss, and fog capacity, the process of task allocation across stationary and mobile fog nodes is formulated into a joint optimization problem. This task allocation in VFC is known as a non-deterministic polynomial-time hard (NP-hard) problem. In this paper, we present the task allocation to fog nodes as a bi-objective minimization problem, where a trade-off is maintained between the service latency and quality loss. Specifically, we propose an event-triggered dynamic task allocation (DTA) framework using Linear Programming based Optimization (LBO) and Binary Particle Swarm Optimization (BPSO). To assess the effectiveness of Folo, we simulated the mobility of fog nodes at different times of a day based on real-world taxi traces and implemented two representative tasks, including video streaming and real-time object recognition. Simulation results show that the task allocation provided by Folo can be adjusted according to actual requirements of the service latency and quality, and achieves higher performance compared with naive and random fog node selection. To be more specific, Folo shortens the average service latency by up to 27% while reducing the quality loss by up to 56%.

Index Terms—Computing Offloading, Vehicular Fog Computing (VFC), Dynamic Task Allocation, Linear Programming (LP), Binary Particle Swarm Optimization (BPSO).

1 Introduction

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

Cloud models are not applicable to environments where operations are critical to latency. As an example, the prevention of collisions and accidents cannot afford the latency caused by round trips between the vehicle and the remote cloud. In order to solve this problem, 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 data is being generated and acted upon [7].

- *Chao Zhu, Jin Tao, Yu Xiao, Giancarlo Pastor, Quan Zhou, Antti Ylä-Jääski are with Aalto University, Espoo, Finland.*
  E-mail: {chao.1.zhu, jin.tao, giancarlo.pastor, yu.xiao, quan.zhou, antti.yla-jaaski} @aalto.fi
- *Jin Tao is also with College of Engineering, Peking University, Beijing, China.*
- *Yusheng Ji is with National Institute of Informatics , Tokyo, Japan.*
  E-mail: kei@nii.ac.jp
- *Yong Li is with Tsinghua University, Beijing, China.*
  E-mail: liyong07@tsinghua.edu.cn

*This research work is partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 815191, the Nokia Center for Advanced Research, Technology Industries of Finland Centennial Foundation, Academy of Finland (297892, 315660) and JSPS KAKENHI (JP16H02817). Chao Zhu and Jin Tao contributed equally to this paper and *the corresponding author is Yu Xiao.*

Due to the mobility of vehicles, the computational and communication workloads generated by vehicular applications vary with time and location. At the same time, as Xiao et al. proposed [8] and Satyanarayanan et al. [9], fog nodes in Vehicular Fog Computing (VFC) scenarios can be either stationary or mobile. For instance, commercial fleets with sufficient computing resources and network connectivity can become mobile fog nodes to handle service instances generated by neighboring vehicles and passengers. The mobility of fog nodes opens up new opportunities, such as on-demand computing [8]. However, it also adds a layer of complexity to the task allocation process in VFC.

According to [10, 11], by relaxing the tolerance of quality loss, service latency could be reduced to a certain extent. 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. In this paper, a dynamic task allocation solution, called Folo, is designed that optimizes service latency and QLR under the application-specific requirements, e.g., communicating and computing demands. Furthermore, the process of task allocation across stationary and mobile fog nodes is formulated as a bi-objective optimization problem. The optimization objectives include both minimizing the average service latency and reducing the overall quality loss. As it proves to be an NP-hard problem, an event-triggered task allocation architecture, i.e., Dynamic Task Allocation (DTA) is proposed using Linear Programming based Optimization (LBO) and Binary Particle Swarm Optimization (BPSO) to solve it.

To evaluate the effectiveness of Folo, a set of real-world taxi traces collected in Shanghai city is used to simulate

the mobility of fog nodes at different times of day, and two example tasks, including video streaming and real-time object recognition is implemented [12]. The resource consumption of the example tasks is measured through real-world experiments, and the performance of vehicle-to-fog communications is analyzed using veinsLTE, which is an inter-vehicle communication simulator [13]. Compared with the existing solutions, e.g., naive and random, Folo shortens the average service latency while reducing the overall quality loss, and achieves a better balance between service latency and quality loss.

The key contributions of this work are summarized below:

- Folo is designed, which is a novel solution for latency and quality optimized task allocation across stationary and mobile fog nodes in VFC.
- The process of task allocation is formulated as a joint optimization problem and solved with LBO and BPSO based DTA approach.
- The effectiveness of Folo is evaluated through simulation, using real-world application profiles and taxi traces as 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. An overview of Folo is given in Section 2. Section 3 describes the system model and problem formulation. The dynamic task allocation approach is presented in Section 4. Section 5 explains an application profiling. Section 6 discusses the evaluation configuration and final results before we conclude in Section 7.

2 Related Work

Fog computing shares the same principle of moving computing resources to the edge with mobile edge computing [14]. Different architectures of VFC have been proposed in the literature. For example, Satyanarayanan et al. [9] proposed to turn each vehicle into one fog node and select a coordinator for each zone. Xiao et al. [8] preferred to turn commercial fleets into fog nodes so as to serve neighboring vehicles and passengers, while Hou et al. [15] recommended utilizing additional computing power on slow moving or parked vehicles. In addition, Ni et al. [16] studied the architecture of fog-based vehicular crowdsensing considering security, fairness, and privacy.

Relevant to our work, previous research works have investigated task allocation in fog/edge computing [17–24]. Li et al. [11] minimized service response time and energy consumption by jointly optimizing the offloading strategy and the QoR for all edge nodes. Sardellitti et al. [25] jointly optimized radio and computational resources of multiple cells in edge computing. Liu et al. [26] studied the multi-task allocation problem for the edge environment with consideration of resource-intensive and latency-sensitive mobile applications. Dinh et al. [27] presented an offloading framework to jointly minimize the execution latency of tasks and power consumption of devices considering CPU frequency. Deng et al. [28] studied the trade-off between energy consumption and transmission delay in a cloud computing system. However, these existing results cannot be directly applied to VFC, for they did not consider the mobility of vehicles. Feng et al. [29] proposed a job caching framework in VFC based on ant colony optimization algorithm. However, it focused on caching and did not consider other scenarios e.g., local processing. To the best of our knowledge, Folo is the first joint optimization to provide service latency and quality in VFC, with consideration of the mobility of fog nodes.

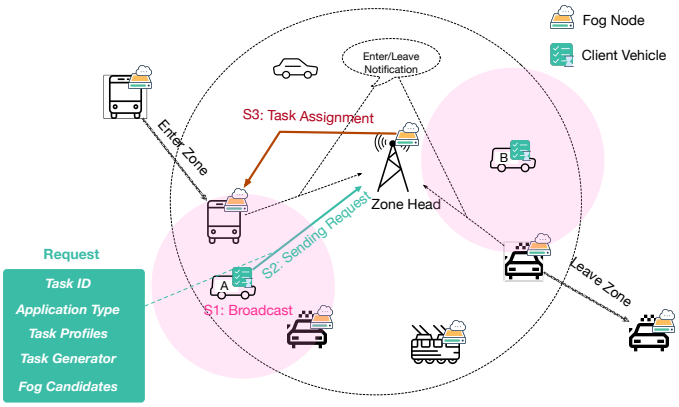


Fig. 1: Overview of Folo. The light magenta circle indicates the communication range of the client vehicle in the center. Here the communication is limited to one hop of DSRC.

3 System Overview

In this section, the related terms are defined and an overview of the process of task allocation in Folo is given.

3.1 Related Terms

Fog Nodes: In Folo, two types of fog nodes is considered as follows:

* *Stationary Fog Nodes*: The computing nodes co-located with cellular base stations, road side units, Wi-Fi access points, or any other stationary infrastructure.
* *Mobile Fog Nodes*: The computing nodes carried by moving vehicles with onboard Dedicated Short-range Communication (DSRC) [30] and Long-Term Evolution (LTE) communication modules.

Tasks: The process of an application can be broken down into a set of services. For example, AR-based driving assistance includes services such as object recognition and video streaming. Services are independently deplorable. And each service has its own latency, quality constraints, and workload profiles. In Folo, a task refers to a service instance. For instance, suppose there is a vehicle demand for recognizing potential obstacles from a video clip taken by its camera and requires to offload this service instance to fog nodes due to its limited computing ability. Then, the task in this service instance consists of video uploading, video preprocessing, feature extraction, pattern matching and results downloading. The task is considered to be the basic unit for task allocation. That is to say, from the perspective of task allocation, a task cannot be divided into sub-tasks.

Client Vehicles: The Vehicle that generates tasks are defined as the client vehicle. Each client vehicle may generate

multiple tasks simultaneously. Tasks generated 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 completely covered by cellular networks. In a like manner, according to [31], an urban area is divided into service zones and a stationary fog node within a zone is selected to manage and coordinate all the fog nodes in the same zone. The coordinator is called zone head. For simplification of the system model, a Long-Term Evolution (LTE) base station is always selected to be the zone head, and assume that mobile fog nodes are deployed on commercial fleets, e.g., taxis and buses. With the existing cellular registration mechanisms, mobile fog nodes always inform the zone head as they enter or leave the zone. In addition, they regularly report their moving directions, locations and available capacities to the zone head. Note that locations and dynamics of fog nodes, client vehicles, as well as base stations are visible to Folo.

3.2 Process of Task Allocation

The task allocation process in Folo is manifested in Figure 1. The whole process consists of 4 steps as shown below.

3.2.1 Mobile fog nodes discovering

In the initial phase, a client vehicle needs to determine which mobile fog nodes are within its communication range. It broadcasts one-hop probe messages through DSRC and collects responses from fog nodes.Any fog nodes that respond are included in the list of fog candidates. As shown in Figure 1, the fog candidates of the client vehicle A are the fog nodes within the communication range of A.

3.2.2 Requests sending

After the fog candidates are discovered, the client vehicle sends a request to the zone head via 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 tolerable latency, supported video resolutions and data size.
* *Task Generator*: The client vehicle that generates data and sends the request.
* *Fog Candidates*: The fog nodes within the communication range of the client vehicle.

3.2.3 Tasks to fog candidates assigning

When receiving a request from any client vehicles, the zone head executes the task allocation algorithm to decide where to run the tasks. Concerning the frequent changes in network topology, the zone head would estimate the service time of the fog node candidates according to their predicted paths and filter out the fog nodes which can only provide services for very short time. Furthermore, with the existing cellular registration mechanisms, the zone head would check the positions of mobile fog nodes periodically to avoid out-of-date information. The details of the algorithm will be presented in Section 5.

3.2.4 Task migration scheduling

The connection between client vehicles and mobile fog nodes may not last until the assigned tasks complete due to their mobility. For instance, when the corresponding fog node moves out from the current service zone, the execution of a task may be interrupted. In this case, the zone head of the current service zone must call another fog node to take over the task.

In Folo, we propose to utilize the light-weight containers (e.g., LXD) for service provisioning [32]. Task migration can be event-triggered and the tasks would be migrated with Iterative Live Migration (i.e., pre-copy) [33]. When the task migration is triggered (e.g., the signal strength decays to a certain threshold), with the Checkpoint & Restore in Userspace (CRIU) technology, the disk files and memory pages of the container would be copied from source fog node to destination fog node while the container keeps running.

4 System Modeling and Problem Formulation

4.1 System Model

In this section, the system model of Folo is presented. $\mathcal{K}_i$ is defined as the set of tasks generated by the client vehicle i, and $\mathcal{J}_i$ is defined as the set of fog candidates for the client vehicle i. A binary variable x_{ik} is defined to indicate whether the task k is generated by i, and another binary variable x_{ij} is defined to indicate whether fog node j is available for i (i.e., the fog node j is in the list of fog candidate). The detailed notations and definitions are listed in Table 1. Then, we have

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

Furthermore, a binary variable x_{jk} is defined to indicate whether task k is assigned to fog node j. The task k will be successfully assigned to the fog node j only if the fog node j is available for the generator of the task k. Therefore,

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

4.1.1 Service Latency

Transmission Delay: Given limited bandwidth, the transmission delay depends on the size of data to be transmitted. In this paper, Quality Loss of Results (QLR) is proposed to quantify the near-optimal of the user acquired service quality (i.e, quality degradation). For each task k, q_k is defined as the level of QLR, and $D(q_k)$ as the corresponding size of data to be transmitted. The transmission delay T_k^{Comm} is calculated according to the transmission data rate C_{ij} of the link between the selected fog node j and the client vehicle i.

$$T_k^{Comm} = \frac{D(q_k)}{C_{ij}} \tag{3}$$

Processing Delay: $P(q_k)$ is used to denote the processing delay of task k with QLR equal to q_k, thus, the processing latency T_k^{Proc} is expressed as follows:

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

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
$D(q_k)$	the 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}	the data rate between client vehicle i and fog node j.
S_j	the capacity of fog node j
τ_k	the maximum tolerable service latency of task k
T_k^{Comm}	the transmission delay
T_k^{Proc}	the processing delay
ℓ	round trip time (RTT) overhead
T_k	the total service latency
T	the maximum service latency of all tasks
Q^{sum}	the total quality loss of all tasks
$\mathcal{Q}$	the set of selected QLR levels
$\mathcal{X}$	the set of selected fog nodes
$\mathcal{U}$	the unassigned task set

TABLE 1: Notations and definitions

The service latency of each task k can be written as follows:

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

where ℓ refers to a constant overhead, which captures RTT between a fog node and a client vehicle.

4.1.2 Constraints

Quality Loss Constraint: The tolerance for quality loss is application-specific. In Folo, 5 levels of QRL is defined. Level 1 refers to the strictest demand for quality, while Level 5 represents the highest tolerance for quality loss. In practice, q_k can be defined based on video resolution e.g., in the case of video streaming.The QLR constraint for the task k is represented as follows:

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

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

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

Service Latency Constraint: According to measurements in [34], the maximum tolerable service latency for an AR navigation application is 250 ms, whereas the maximum tolerable service latency for video streaming application can be as much as 1 second.τ_k is used to denote the maximum tolerable service latency for the task k. In order to ensure that the task can be completed in time, the service constraint is as follows:

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

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

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

4.2 Problem Formulation

The maximum service latency of all tasks is denoted by $T = \max_{k \in \mathcal{K}}\{T_k\}$, and the total quality loss by summation of the QLR levels of all tasks: $Q^{sum} = \sum_{j \in \mathcal{J}, k \in \mathcal{K}} \{q_k x_{jk}\}$.

Folo is designed to minimize the maximum service latency T while minimizing the total quality loss Q^{sum}. Nonetheless, the two objectives are coupled by q_k and cannot be optimized simultaneously. In the following, the trade-off between the two objectives is investigated and the joint objectives function is defined as $\varphi_t T + \varphi_q Q^{sum}$, where $\varphi_t, \varphi_q \in [0, 1]$ are two scalar weights.

$\mathcal{X} = \{x_{jk}\}$ is used 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.

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

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

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

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

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

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

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

Proposition 1: $\xi 1$ is a NP-hard problem.
Proof 1: See Appendix A.

5 Dynamic Task Allocation Approach

In the vehicular scenario, because of the mobility of client vehicles, the geographic locations of tasks are changing. In order to solve the dynamic task allocation problem, an event-triggered DTA algorithm is proposed. In DTA, the task allocation solvers (TAS) will be triggered by the upcoming events (such as new task generation and service interruption) to solve the problem of task allocation. Considering the highly mobile environment (i.e., the topology of the network is changing frequently) of VFC, the system should make decisions of task allocation in real time. In this section, we first propose LBO (Linear Programming based Optimization) to solve Problem $\xi 1$. With the help of the linear programming solver (e.g., glpk), the system can make an efficient task allocation decision with a balanced multi-objective optimization in a short time. To further shortcut the computing time, we then choose a PSO-based algorithm

called BPSO (Binary Particle Swarm Optimization) because of its relatively low computational complexity, since there are no crossover, decoding, and encoding of a genetic algorithm (GA) compared with other bio-inspired algorithms (e.g., ant colony optimization).

5.1 Linear Programming based Optimization

5.1.1 Problem Linearization

Problem $\xi 1$ is a non-linear optimization problem because Constraint (10b) and Constraint (10c) contain a production of two variables q_k and x_{jk}. To convert the optimization into an LP problem, $y_{jk} = q_k x_{jk}$ is defined and $\mathcal{Y} = \{y_{jk}\}$ is used to denote the set of variable y_{jk}. Moreover, the discrete variable q_k is relaxed into a continuous one, that is $q_k \in [1, 5]$.

Learn from [11], two linear approximate trade-off functions, $P(q_k) = a_t q_k + b_t$, and $R(q_k) = a_r q_k + b_r$ are considered. In addition, a new variable t with an additional constraint $t \geq \max_{k \in \mathcal{K}} T_k$ is introduced.

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 $\xi 3$, the LP problem that is equal to Problem $\xi 1$:

$$\xi 3 : \min_{\mathcal{X},\mathcal{Q},\mathcal{Y},t} \ \varphi_t t + \varphi_q \sum_{j \in \mathcal{J}, k \in \mathcal{K}} q_k x_{jk} \tag{11}$$

s.t.

$$\forall j,k, 0 \leq x_{jk} \leq 1, 1 \leq q_k \leq 5 \tag{11a}$$

$$\forall j,k, 0 \leq y_{jk} \leq 5 x_{jk} \tag{11b}$$

$$\forall j,k, q_k - 5(1 - x_{jk}) \leq y_{jk} \leq q_k \tag{11c}$$

$$\forall k, \sum_{i \in \mathcal{I}, j \in \mathcal{J}} (\frac{b_d}{C_{ij}} + b_t + \ell) x_{jk} + (\frac{a_d}{C_{ij}} + a_t) y_{jk} \leq \tau_k \tag{11d}$$

$$\forall j, \sum_{k \in \mathcal{K}} b_r x_{jk} + a_r y_{jk} \leq S_j \tag{11e}$$

$$\forall k, \sum_{j \in \mathcal{J}} x_{jk} = 1 \tag{11f}$$

$$\forall k, \sum_{i \in \mathcal{I}, j \in \mathcal{J}} (\frac{b_d}{C_{ij}} + b_t + \ell) x_{jk} + (\frac{a_d}{C_{ij}} + a_t) y_{jk} \leq t \tag{11g}$$

$$\forall i,j,k, x_{jk} \leq x_{ij}, x_{jk} \leq x_{ik} \tag{11h}$$

Proposition 2: Constraints (11b) and (11c) are equal to the constraint $y_{jk} = x_{jk} q_k$
Proof 2: See Appendix B.

5.1.2 Linear Programming based Optimization

In LBO, the input is the unassigned tasks set $\mathcal{U}$, which contains the information about client vehicles set $\mathcal{I}$, fog nodes set $\mathcal{J}$, tasks set $\mathcal{K}$. Furthermore, based on information of tasks and location of vehicles, the optimization matrix is formulated, 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. Next, the LBO would take the end-to-end latency and overall quality loss of tasks into consideration to implement an LP solver to get the balanced optimization solution with continuous values. After that, the continuous values are reshaped to integral ones. Similar to [27], for each task k, if $\exists m \in \mathcal{J}$, $x_{mk} = \max_{j \in \mathcal{J}}(x_{jk})$, x_{mk} is set to 1 and the rest to 0. Meanwhile, each q_k is rounded up to its nearest integer.

TABLE 2: BPSO Particles

	Assign_1	...	$\text{Assign}_{\lvert\mathcal{K}\rvert}$	QLR_1	...	$\text{QLR}_{\lvert\mathcal{K}\rvert}$
Particle_1	ass_1	...	$ass_{\lvert\mathcal{K}\rvert}$	q_1	...	$q_{\lvert\mathcal{K}\rvert}$
...	...	...	...	...	...	...
Particle_p	ass_1	...	$ass_{\lvert\mathcal{K}\rvert}$	q_1	...	$q_{\lvert\mathcal{K}\rvert}$
...	...	...	...	...	...	...
$\text{Particle}_{\lvert P\rvert}$	ass_1	...	$ass_{\lvert\mathcal{K}\rvert}$	q_1	...	$q_{\lvert\mathcal{K}\rvert}$

5.1.3 Complexity Analysis

According to [35], the LBO has the complexity in polynomial time $\mathcal{O}(v^{3.5}B^2)$, where v is the number of variables and B is the number of the bits in the input. Given fog nodes set $\mathcal{J}$ and tasks set $\mathcal{K}$, the time complexity of LBO is $\mathcal{O}((|J||K| + 2|K| + 1)^{3.5}B^2)$.

5.2 Binary Particle Swarm based Optimization

5.2.1 Parameterization

Due to the high computation complexity of LBO, a heuristic algorithm BPSO based on Partial Swarm Optimization (PSO) [36] is proposed. However, the original PSO algorithm is designed for solving problems with continuous solutions. To fit for the problem, we first parameterized the particles.

Similar with LBO, BPSO uses the unassigned tasks set $\mathcal{U}$ as input and manages to find the assignment decision x_{jk} and quality loss level q_k for each task k.Firstly, a swarm set P with $|P|$ particles are generated. Each particle has a search space of $2 \times |\mathcal{K}|$ dimension. As shown in TABLE 2, the dimensions $1 \sim |\mathcal{K}|$ demonstrate the decision of the task assignment. Each part of dimension has a discrete set of possible values limited to $\{1 \leq ass_k \leq |\mathcal{J}|\}$. Therefore, the results of dimensions $1 \sim |\mathcal{K}|$ can be transferred to task assignment set $\mathcal{X}$, where $x_{jk} = 1$ when $ass_k = j$. The last part of dimensions from $|\mathcal{K}| + 1$ to $2 \times |\mathcal{K}|$ refer to the QLR selected for each task k, and each dimension has a discrete set of possible values limited to $\{1 \leq q_k \leq 5\}$.

Using such a particle parameterization, the swarm is represented as a $|P| \times (2 * |\mathcal{K}|)$ two-dimensional array consisting of $|P|$ particles. Each particle is represented as a vector of $|\mathcal{K}|$ task assignment decisions and $|\mathcal{K}|$ QLR selections. Thus, each particle flies in a $(2 * |\mathcal{K}|)$-dimensional search for space to search for the best solution for Problem $\xi 1$.

5.2.2 Binary Partial Swarm Optimization

In PSO, the p^{th} particle is denoted by $X_p = (x_{p1}, x_{p2}, ..., x_{pD})$ and the best position it has experienced (with the best fitness value) is recorded as X^p_{best}. The index number of the best position experienced by all particles in the swarm is called G_{best}. The velocity of particle p is represented by $V_p = (v_{p1}, v_{p2}, ..., v_{pD})$. For each generation, its d dimension ($1 \leq d \leq D$) updates according to the following equation:

$$V_{pd} = wV_{pd} + C_1 rand_1[X^p_{best} - X_{pd}] + C_2 rand_2[G_{best} - X_{pd}] \tag{12}$$

$$X_{pd} = X_{pd} + V_{pd+1} \tag{13}$$

Algorithm 1 BPSO: **B**inary **P**aticle **S**warm **O**ptimization

Input: Unassigned task set $\mathcal{U}$
Output: Assignment decision set $\mathcal{X}$; QLR set $\mathcal{Q}$

1: Extract the client vehicles set $\mathcal{I}$, fog nodes set $\mathcal{J}$, tasks set $\mathcal{K}$ from the unassigned task set $\mathcal{U}$
2: Initialize a particle swarm with $|P|$ population, global velocity (C_2), local velocity vector (C_1), inertia weight (w) and velocity vector (V) for each particle in a population, $iterator = 1$
3: **while** $iterator \leq$ Max Iteration Number **do**
4: Calculate the fitness value for each particle using Eq. (10)
5: X^p_{best} = Best position value for each particle
6: G_{best} = Minimum fitness value from the set of service allocation vectors
7: Calculate V_{pd} according to Eq. (12)
8: Calculate X^{try}_{pd} according to Eq. (13)
9: Round X^{try}_{pd} to discrete values
10: **if** X^{try}_{pd} satisfy all constraints in Problem $\xi 1$ **then**
11: update X_{pd} ($X_{pd} = X^{try}_{pd}$)
12: **end if**
13: $iterator = iterator + 1$
14: **end while**
15: **return** G_{best}

where w is the inertia weight, C_1 and C_2 are acceleration constants, and $rand_1$ and $rand_2$ are two random functions with a range [0, 1].

To order to adapt to the problem, we customize the method in [37] and propose the BPSO algorithm, which is illustrated in Algorithm 1. As illustrated in Line 2, we first initialize the related parameters, which are global velocity (C_2), local velocity vector (C_1), inertia weight (w), according to [37] ($C_1 = C_2 = 1, w = 0.9$). Simultaneously, we randomly generate as many potential assignments for the problem as the size of the initial population $|P|$. Here, we used Eq. (10) as the fitness function.

The algorithm keeps an updated version of two special variables through out the course of its execution: global best position G_{best} and local best position X^p_{best}. It does that by conducting two ongoing comparisons: for each particle, compare its fitness value with the best position X^p_{best} it has experienced. If it is better, then it will be set as the current best position X^p_{best}; for each particle, compare its fitness value with the best position G_{best} experienced globally. If it is better, reset G_{best}'s index number. These two positions affect the new velocity of every particle in the population according to Eq. (12). As shown in this equation, two random parameters control the amount of effect the two positions (i.e., G_{best} and X^p_{best}) impose over the new particle velocity. The algorithm uses the new velocity to update the particle's current position with a new position according to Eq. (13). Note that, until now the new positions of particles are consist of continuous values, which are meaningless solutions for the task allocation problem. Thus, as shown in Line 9, we round the values of all dimensions in each particle to their nearest integers and check whether the values fit all the constraints listed in Problem $\xi 1$. Then, the algorithm evaluates the fitness of these particles according to their rounded positions. Once the current positions are better than the previous ones and all the constraints are satisfied, all particles will adjust their positions and constitute the new status of the population. This process continues until the termination condition is satisfied (e.g., maximum iteration or getting the required result).

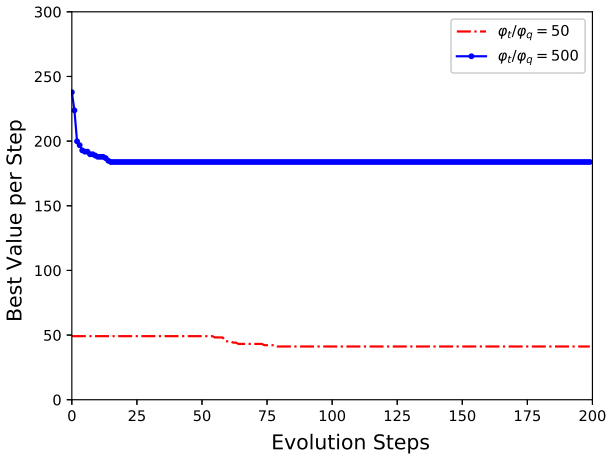


Fig. 2: Evolution of BPSO

5.2.3 *Convergence Analysis*

To explore the convergence of BPSO, we conduct one round of BPSO with an input of 14 tasks for 200 iterations. We record the fitness value of each iteration and illustrate the results of the evolution of BPSO in Figure 2. In Figure 2, we can see there is a dramatic drop in fitness value (from 250 to 180) when $\varphi_t/\varphi_q = 50$. However, when $\varphi_t/\varphi_q = 500$, the BPSO does not have an apparent convergence (from 50 to 48). The most likely reason is that the BPSO falls into a local optimum and can not reach a further better solution. We may introduce chaos searching [38], induce Niching Behavior [39], or use other strategies, such as resampling technique [40], or weight-digression strategy [41] to balance its global search ability and convergence speed in future work.

5.3 Dynamic Task Allocation

In this section, an event-triggered algorithm DTA based on two designed TASs, e.i., LBO and BPSO are proposed, and the detailed algorithm is described in Algorithm 2.

5.3.1 *Initialization*

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

5.3.2 *Event Handling*

In Folo, the zone head detects events in its service zone. Here, two types of events are considered:

Algorithm 2 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 **TAS** 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: Execute **TAS** for new coming task k^{new}
11: **case** Service interrupted task k^{break}**:**
12: Remove k^{break} from $\mathcal{A}$, $k^{break} \rightarrow \mathcal{U}$
13: Execute **TAS** for migrating task k^{break}
14: **end switch**
15: Remove finished task k^{done} from $\mathcal{A}$
16: **end while**

* *New Tasks*: The zone head receives a new request from a client vehicle.

* *Service Interruption*: When a client vehicle and a mobile fog node are moving towards a different direction, the connection between them may break down. Assuming that mobile fog nodes keep monitoring the channel states and 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 12 to Line 13.

6 Application Profiling

To test Folo, two example tasks, i.e., video streaming and real-time object recognition are implemented. The reason for choosing these two tasks is that they are building blocks of many vehicular applications, such as AR-based driving assistance. In the experiments, the impact of the variation of service quality on the service latency and the amount of resource consumption are explored. This section describes the experiments of creating task profiles and analyzing the performance of vehicle-to-fog communication. The hardware devices used for the experiment are listed in Table 3. The experimental results will be used later for configuring the simulator described in Section 7.

Video streaming is implemented based on *Kurento* [42], an open source platform for WebRTC-based real-time communications. As shown in Figure 3a, the Kurento media server is run on a Linux desktop and a client application is run on three Android phones. The client application captures video and sends it to the media server. The CPU/GPU/memory usage of the media server is measured while receiving video streams from phones.

Five different video resolutions, {1920 * 1080, 1280 * 960, 960 * 720, 640 * 480, 320 * 240} are tested. The frame rate of video streaming is limited to 14 fps owing to the hardware constraint. The QLR level of video streaming is defined 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. On the basis of the results, a linear model $R(q_k)$ is built to estimate the memory usage (in MB).

$$R(q_k) = -27.5 \times q_k + 247.5 \tag{14}$$

Given a fixed frame rate, the data size depends on the video resolution. Assuming that the transmission video is compressed with the Youtube-HD standard. According to [43], the data size (in KB) of each frame $D(q_k)$ is formulated as follows:

$$D(q_k) = -7.7 \times q_k + 41.2 \tag{15}$$

In the second experiment, an AR-based object recognition application is implemented based on Yolo [44]. As shown in the Figure 3c, the objects recognized from video streams are labeled in the camera view. Same with video streaming, the QLR level of this task is also defined based on video resolution. In experiments, the processing time is measured considering recognizing objects from each frame.

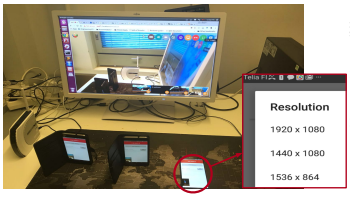
TABLE 3: Experiments Hardwares

Video Streaming

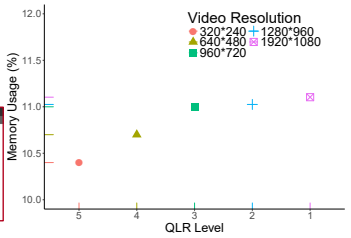
	Server	Client	
Hardware	Desktop	Phone	2 * Phone
OS	Linux	Android 7.0	Android 7.0
Model	N.A.	Huawei Mate9	Huawei P10
CPU	4x3.2GHz	4x2.36GHz	4x2.36GHz
Memory	32GB	6GB	4GB

Object Recognition

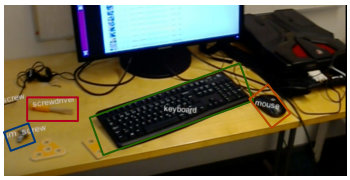
	Server	Client
Hardware	LapTop	Web Camera
Model	HP-zbook G3	Logitech HD
GPU	Quadro M2000M	N.A.
CPU	8x2.7GHz	N.A.



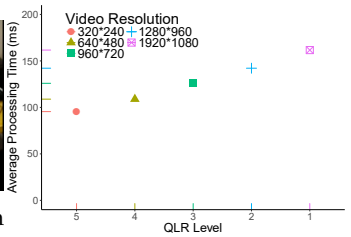
(a) Video Streaming



(b) QLR v.s. Memory consumption



(c) Real-time Object Recognition in AR mode



(d) QLR v.s. Processing Time

Fig. 3: Application Profiles: the highest resolution corresponds to the lowest QLR level, i.e., video resolution 320*240, 640*480, 960*720, 1280*960, 1920*1080 correspond to QLR level of 5, 4, 3, 2, 1 respectively.

TABLE 4: 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[45]	
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	

As shown in Figure 3d, the processing time is compared between 5 QLR levels. For the processing time has a nearly negative linear correlation with the QLR level, the processing time (in ms) per frame $P(q_k)$ is formulated as follows:

$$P(q_k) = -16.56 \times q_k + 176.5 \tag{16}$$

7 Evaluation

7.1 Evaluation Configuration

7.1.1 Vehicle Mobility and Task Generation

A set of real-world taxi traces is used to simulate the mobility of mobile fog nodes. The dataset contains the GPS traces collected from April 13 to 30, 2015 in Shanghai city, which was collected by the iData Laboratory of Tongji University [12]. To evaluate the impact of the density of mobile fog nodes, an area of 4 km^2 acreage near Shanghai Pudong Airport is selected, and the traces within the area during two time periods are collected.

* *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 4a, 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, 100 vehicle routes in SUMO are generated, following the method used in [46]. These vehicles act as client vehicles and generate video streams to be processed on fog nodes. Each time a random number (no more than 8) of video streams is generated, each stream lasts 10 seconds. Our simulation runs last for 60 seconds. For instance, 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 moves out of the current service zone, ongoing tasks should be migrated to other fog nodes. Figure 4b shows the number of task migration that occurred during Time Period I and II. According to the figure, task migration happens more frequently in Time Period I when the density of mobile fog node is higher (Since the GPS fixes included in the taxi traces are sparse, the actual need for task migration may be less).

7.1.2 Network Configuration

DSRC and LTE are the most popular in-vehicle networking technologies. DSRC is designed based on IEEE 802.11p. The data rate of DSRC can reach up to 27Mb/s with around 300 meters of coverage [45]. Compared with DSRC, LTE has a much wider coverage and more deterministic quality of service (QoS) guarantees. According to [47], LTE can support User Equipment (UE) with high mobility at the speed of 350 km/h.

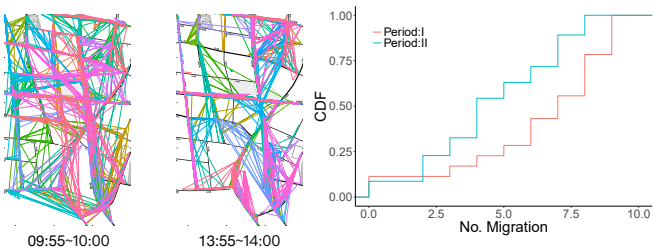
In this paper, vehicles broadcast beacons through DSRC to handshake with each other, such as detecting fog nodes and scheduling task migration. In addition, the DSRC channels are responsible for data transmission between client vehicles and mobile fog nodes. Alternatively, LTE is used for communications between client vehicles and the corresponding zone head (base station), e.g., sending requests and notifications of entering/leaving a service zone. The zone head itself is also a stationary fog node.

According to Appendix C, the default data rate of DSRC is set to be $500kbps$, and the data rates of LTE is set to be $\{550kbps, 450kbps, 200kbps, 150kbps\}$, depending on the channel state information. Additionally, according to the measurements in [45], the RTT overhead is set to be 20ms for DSRC and 300ms for LTE. And the other parameter values are listed in TABLE 4.

7.1.3 Task Allocation Strategies

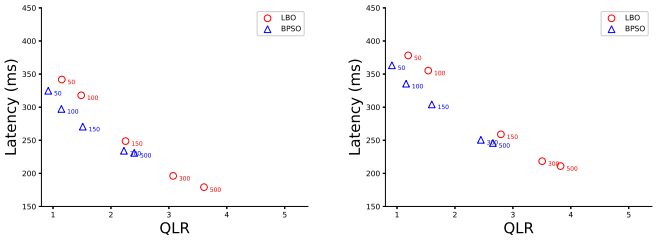
As mentioned in Section 4, the scalar weight φ_t and φ_q refers to the optimization tendency toward service latency and quality, respectively. When φ_t/φ_q is higher, the task allocation strategy is latency sensitive; otherwise, it is quality sensitive. For both of LBO and BPSO, we tune the ratio of the scalar weights, φ_t/φ_q, from 50 to 500, and compare the results between the two time periods in Figure 5.

When the density of mobile fog nodes is high, as illustrated in Figure 5a, the average service latency in case of LBO is around 350 ms and that of BPSO is around 330 ms when the QLR level is 1. The service latency in both cases decreases with the tolerance of quality loss. However, LBO outperforms BPSO when the algorithms are acting toward the latency optimization. For example, when the QLR level increases to 4, the average latency in case of LBO would drop up to 200 ms but that of BPSO still remain at around 220 ms. The reason may be that the BPSO has been dragged into local optimum as illustrated in Figure 2. Compared with Figure 5b where less mobile fog nodes are available,



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

Fig. 4: Simulation of Mobile Fog Nodes



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

Fig. 5: Service Performance v.s. Scalar Weight

the service latency in Figure 5a is on average 50 ms shorter when the QLR level is less than 2.

According to Figure 5, we define 4 variants of DTA based on the value of φ_t/φ_q.

* *LBO_Q*: QLR Sensitive LBO with $\varphi_t/\varphi_q = 50$.
* *LBO_T*: Latency Sensitive LBO with $\varphi_t/\varphi_q = 500$.
* *BPSO_Q*: QLR Sensitive BPSO with $\varphi_t/\varphi_q = 50$.
* *BPSO_T*: Latency Sensitive BPSO with $\varphi_t/\varphi_q = 500$.

For comparison, two other strategies, i.e., Rand and Naive, are implemented. These two strategies have been investigated in recent publications [27, 29]. Rand refers to F.Rand.Assign in [27]. It randomly selects one fog node from among the available candidates. Naive always selects the fog node with the highest available data rate, which is close to AVE+Naive in [29]. In our experiments, we assume that the QLR level is randomly chosen.

7.2 Evaluation Results

7.2.1 *Service latency vs QLR*

As shown in Figure 6a, the average service latency and QLR levels among 6 different strategies are compared. The results of Naive is used as the baseline. Overall speaking, the lower the percentage is, the better the performance is. Here, service latency and quality are used to measure the performance.

According to the results, LBO_T leads to the shortest service latency, while BPSO_Q gains the highest quality. In the period I, compared with Naive which always chooses the fog node with the highest data transmission rate, LBO_T shortens the overall service latency by up to 19%. However, the BPSO_T does not have a good performance for shortening service latency due to the inherent flaws (easy to fall into local optimum) of PSO. From the service quality aspect, LBO_Q and BPSO_Q decrease the QLR by up to 42% and 56%, respectively. Notably, the BPSO_Q achieves a more balanced optimization (service latency increased by 45%) compared with LBO_Q (service latency increased by 54%). It can also be observed that when the density of mobile fog nodes is lower, these strategies perform better. In the case of Rand, the impact is less obvious, because Rand does not consider the vehicle density impact on networking performance.

We also compare the distribution of service latency in Figure 6b. In the case of BPSO_T and LBO_T, most tasks are completed between 200 ms and 400 ms. Especially in the case of LBO_T, 55% of tasks are completed within 200 ms when the density of mobile fog node is high, which outperforms the result of Naive and rand. Speaking of BPSO_Q and LBO_Q, these strategies have a longer service latency compared with Naive. This is because they improve the service quality by sacrificing service latency. Overall, if the application has a strict requirement of service latency, LBO_T performs better since all tasks complete within 400 ms when it is applied.

7.2.2 *Memory Capacity vs. Performance*

For the two tasks we have tested, memory usage becomes a performance bottleneck. We have noticed that CPU/GPU requirements can be satisfied as long as the memory requirements are met. Therefore, the performance is evaluated with varying memory sizes. As shown in Table 4, the memory size of a mobile fog node is set to 1.1 GB by default, and that of a zone head is set to be 2.1 GB.

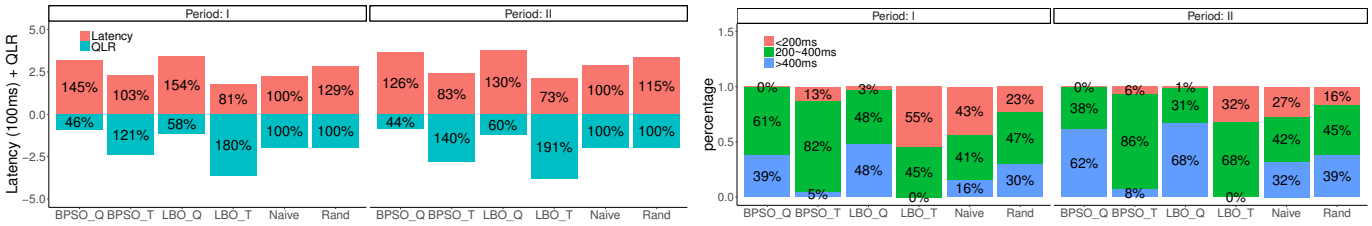
Both BPSO and LBO with $\varphi_t/\varphi_q = 150$ (see Figure 5) are chosen as examples to evaluate the service latency and QLR with different memory capacities settings. The memory size of each mobile fog node is tuned from 0.5 GB to 1.5 GB. As shown in Figure 7, when the memory capacity of BPSO increases, the service latency, and QLR do not show significant change. When the memory capacity of LBO increases, the service latency decrease. In details, the service latency decreases by around 12%.

Figure 8 illustrates the memory overflow issues. Overall speaking, these strategies perform better in Period II when less mobile fog nodes are available. The quality of sensitive DTA algorithms (i.e., BPSO_Q and LBO_Q) have higher memory overflow compared with the latency sensitive ones (i.e., BPSO_T and LBO_T).

In summary, DTA based strategies shorten the average service latency by up to 27% and QLR by up to 56% comparing with Rand and Naive. Furthermore, the density of traffic has the negative impact on the service performance.

8 Conclusion

In this paper, we highlighted Folo, a dynamic task allocation solution for VFC. It is designed to minimize average service latency while reducing the overall quality loss. We considered the constraints on service latency, quality loss, and fog node capacity and formulated the task allocation process as a bi-objective optimization problem, where a trade-off is maintained between the service latency and quality loss. As it is proved to be an NP-hard problem, we proposed an event-triggered dynamic task allocation framework based on LBO and BPSO to solve the optimization problem. To be specific, using LBO, the problem first should be linearized, while BPSO solves the nonlinear discontinuity model directly. We evaluate our solution with simulation, which is configured based on real-world application profiles and mobility data set. Compared with previous works, the task allocation provided by Folo can be adjusted to service latency sensitive and quality sensitive separately according to actual requirements. Specifically, our solution reduces service latency by up to 27% and increases QLR by up to 56%.



(a) Latency and QLR Performance

(b) Service Latency Distribution

Fig. 6: Performance Comparison between task allocation Strategies

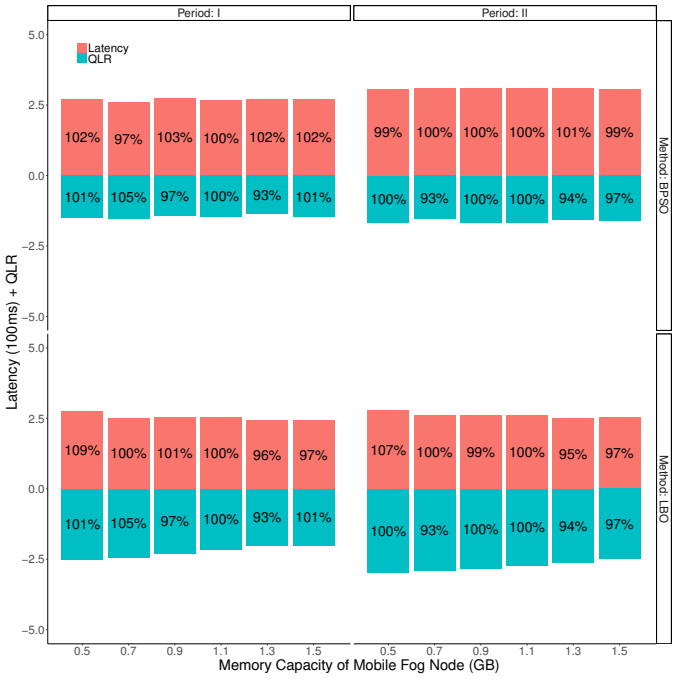


Fig. 7: Performance vs. Memory Capacity

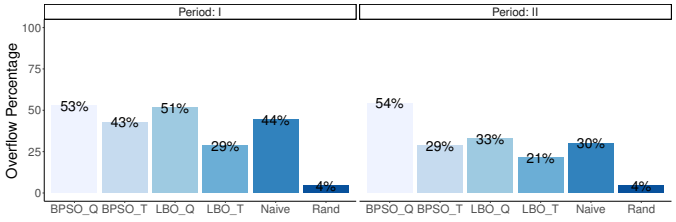


Fig. 8: Server Memory Overflow

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 Comput. Commun. Rev.*, 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] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2018, pp. 1–9.

[11] 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.

[12] "iData Laboratory."

[13] 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.

[14] 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.
[15] 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, Jun. 2016.
[16] 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.
[17] R. Yu, G. Xue, and X. Zhang, "Application Provisioning in Fog Computing-enabled Internet-of-Things: A Network Perspective," *IEEE Infocom*, p. 9, 2018.
[18] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource Provisioning for IoT Services in the Fog," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2016, pp. 32–39.
[19] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-Aware Resource Allocation for Edge Computing," in *2017 IEEE International Conference on Edge Computing (EDGE)*, Jun. 2017, pp. 47–54.
[20] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, "Providing Task Allocation and Secure Deduplication for Mobile Crowdsensing via Fog Computing," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2018.
[21] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint Radio and Computational Resource Allocation in IoT Fog Computing," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2018.
[22] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems With Min-Max Fairness Guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.
[23] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1216–1228, Oct. 2017.
[24] L. Wang, D. Zhang, D. Yang, A. Pathak, C. Chen, X. Han, H. Xiong, and Y. Wang, "SPACE-TA: Cost-Effective Task Allocation Exploiting Intradata and Interdata Correlations in Sparse Crowdsensing," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 2, pp. 20:1–20:28, Oct. 2017.
[25] 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.
[26] 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.
[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] 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.
[29] 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.
[30] 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.
[31] W. Hu, Z. Feng, Z. Chen, J. Harkes, P. Pillai, and M. Satyanarayanan, "Live Synthesis of Vehicle-Sourced Data Over 4g LTE," 2017.
[32] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the Cloud: Distributed Computing for the 99%," in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC '17. New York, NY, USA: ACM, 2017, pp. 445–451.
[33] "How to integrate criu pre-dump feature with docker for Iterative Migration? · Issue #456 · checkpoint-restore/criu."
[34] Y. Xiao, M. Noreikis, and A. Ylä-Jaäiski, "QoS-oriented capacity planning for edge computing," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.
[35] N. Karmarkar, *A New Polynomial-Time Algorithm for Linear Programming*, 1984.
[36] J. Kennedy, "Particle Swarm Optimization," in *Encyclopedia of Machine Learning*. Springer, Boston, MA, 2011, pp. 760–766.
[37] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, Nov. 2002.
[38] J. Tao, Q. Sun, P. Tan, Z. Chen, and Y. He, "Active disturbance rejection control (ADRC)-based autonomous homing control of powered parafoils," *Nonlinear Dynamics*, vol. 86, no. 3, pp. 1461–1476, Nov. 2016.
[39] S. Biswas, S. Kundu, and S. Das, "Inducing Niching Behavior in Differential Evolution Through Local Information Sharing," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 246–263, Apr. 2015.
[40] X. Wang, H. Zhang, S. Fan, and H. Gu, "Coverage control of sensor networks in IoT based on RPSO," *IEEE Internet of Things Journal*, pp. 1–1, 2018.
[41] L. Tong, X. Li, J. Hu, and L. Ren, "A PSO Optimization Scale-Transformation Stochastic-Resonance Algorithm With Stability Mutation Operator," *IEEE Access*, vol. 6, pp. 1167–1176, 2018.
[42] L. López, M. París, S. Carot, B. García, M. Gallego, F. Gortázar, R. Benítez, J. A. Santos, D. Fernández, R. T. Vlad, I. Gracia, and F. J. López, "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.
[43] P. Forret, "Video filesize calculator | toolstudio."
[44] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv:1612.08242 [cs]*, Dec. 2016, arXiv: 1612.08242.
[45] 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," 2017.
[46] 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.
[47] 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.
[48] 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.
[49] 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.
[50] 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.

APPENDIX A
NP-HARD PROOF

A special case, where $\varphi_t = 0, \varphi_q = 1$, is considered, which means that the goal is to minimize the sum of QLR levels. Here $\bar{q}_k$ is defined as the quality gain in the result of task k. It represents the opposite of QLR. Therefore, the goal of minimizing the sum of QLR levels can be transformed into maximizing the sum of quality gains. For the sake of simplicity, Constraint (10b) is removed. In addition, Constraint (10c) is relaxed by assuming that the resource requirement of task k is exactly equal to its quality gain $\bar{q}_k$. Additionally, we assume one client vehicle generate only one task and 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 represent the resource capacity of the fog node. Therefore, we get a simplified optimization problem:

$$\xi 2 : \max \sum_{i \in I} \bar{q}_k \bar{x}_k \tag{17}$$

s.t.

$$\sum_{k \in \mathcal{K}} \bar{q}_k \bar{x}_k \leq S \tag{17a}$$

$$\bar{x}_k \in \{0, 1\} \tag{17b}$$

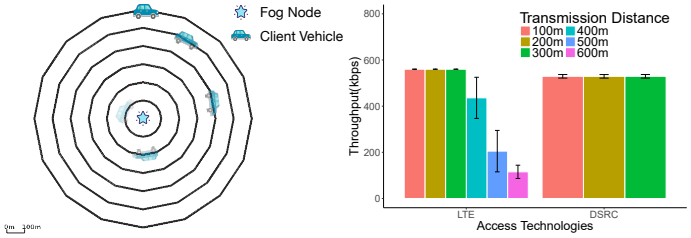
Problem $\xi 2$ is a classic Subset Sum Problem that has been proven to be an NP-complete problem [48]. Hence, the Problem $\xi 1$ is an NP-hard problem is proved.

APPENDIX B
LINEARIZION PROOF

By listing all the possible products in Table 5, we prove that the Constraints (11b) and (11c) are equal to the constraint $y_{jk} = x_{jk} q_k$.

TABLE 5: All possible values of y_{jk}

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	$y_{jk} \leq 5$ $y_{jk} \leq q_k$ $y_{jk} \geq q_k$ $y_{jk} \geq 0$	$y_{jk} = q_k$



(a) Concentric Circles Roads (b) Throughput

Fig. 9: Network performance of LTE vs. DSRC

APPENDIX C
ACCESS TECHNOLOGIES PERFORMANCE

The scenarios of real-time video streaming are simulated using VeinsLTE [13]. VeinsLTE is an extension of Veins [49], which is an open source Inter-vehicle communication (IVC) simulator. VeinsLTE connects a microscope road traffic simulator SUMO [50] with a network simulation engine called OMNET++ through Traffic Control Interface (TraCI). With VeinsLTE, vehicles in the simulation can either exchange data with each other via DSRC, or connect to base stations over LTE.

In SUMO, 6 near round concentric roads are built. Each road consists of two lanes, and the distance between the neighboring roads is set to be 100 meters. As shown in Figure 9a, one fog node is placed in the center of the concentric roads, and a video streaming module is added to the application layer of a client vehicle in VeinsLTE. The client vehicle moving at speed of 20m/s continuously sends video data to the center fog node. By placing the client vehicle on different roads, the throughput of video streaming with varying communication distance could be measured.

From Figure 9b, the data rate in the case of LTE remains stable when the communication distance is within 300 meters. The data rate decreases with the distance when the distance exceeds 300 meters. Unlike LTE, the transmission range of DSRC is shorter than 300 meters, as single-hop DSRC provides stable performance.