2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771889

Enabling Low-latency Applications in Vehicular Networks Based on Mixed Fog/Cloud Computing Systems

Bintao Hu[‡], Jianbo Du[†], Xiaoli Chu[‡]
[‡] *Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, UK*
[†] *Shaanxi Key Laboratory of Information Communication Network and Security*
School of Communications and Information Engineering
Xi'an University of Posts and Telecommunications, Xi'an, 710121, China
Bhu9@sheffield.ac.uk, x.chu@sheffield.ac.uk, dujianboo@163.com

***Abstract*—In order to support delay-sensitive applications of vehicle equipment (V-UE) in the Internet-of-Vehicles (IoV) systems, it is necessary to allow V-UEs to offload their computationally intensive applications to a cloud or fog computing server. Existing works mainly focused on minimising the transmission and processing delays while ignoring the mobility of V-UEs and/or the queueing delays at the cloud or fog servers. In this paper, we consider a vehicular network supported by a mixed fog and cloud computing system, where the queues at the fog node (FN) and the cloud centre are modelled following the M/M/1 and M/M/C queueing models, respectively. To minimise the maximum service delay (which includes the transmission delay, queueing delay and processing delay) among the V-UEs, we propose to jointly optimise the offloading decisions of all V-UEs and the computation resource allocation at the FN while considering the V-UEs' mobility and queueing delays at the FN and cloud centre. This is achieved by devising a fireworks algorithm-based offloading decision optimisation algorithm in conjunction with a bisection method-based FN computation resource allocation scheme. Simulation results demonstrate that our proposed algorithm achieves a much lower maximum service delay than the benchmarks.**

***Index Terms*—Vehicular networks, computation offloading, fireworks algorithm, cloud/fog computing, queueing delay**

I. Introduction

Intelligent transportation systems and the Internet of Vehicles (IoV) have drawn a growing interest from both the academia and the industry in the last few years. As vehicle user equipment (V-UE) has limited capacity to process its applications, while cloud computing may lead to a large offloading delay, fog computing has been considered to provide storage and computation services to nearby V-UEs with a much reduced offloading delay [1].

To support delay-sensitive applications of V-UEs, computation offloading has been investigated to allow V-UEs to offload their computationally intensive applications to a cloud or fog computing server. Zhu *et al.* [2] designed a vehicular fog computing system to shorten the average offloading latency of V-UEs while satisfying the application-specific requirements. Lyu *et al.* [3] proposed to maximise a system utility metric defined based on the task completion time and energy consumption of individual UE by jointly optimizing the offloading decisions and communication and computation resources of multiple UEs in a proximate cloud scenario. Liu *et al.* [4] proposed an Mobile Edge Computing (MEC) based computation offloading scheme for a vehicular network to improve the quality of experience (QoE) of vehicle users by jointly optimising the vehicular terminal's computation time, computation resources and energy consumption. Feng *et al.* [5] proposed a distributed iterative algorithm to minimise task completion delay and energy consumption at each UE side, while satisfying the constraints of task offloading deadline in a MEC system. Wang *et al.* [6] proposed a branch-and-bound algorithm to minimise the average response time for V-UEs in a fog-enabled real-time traffic management system. Du *et al.* [7] designed a mixed fog/cloud system for mobile applications offloading and proposed to minimise the maximum delay among all mobile devices by jointly optimising the offloading decisions, resource allocation and energy consumption. However, the above existing works on cloud/fog/MEC based computation offloading have not sufficiently considered the mobility of vehicles that may increase the delay and energy consumption of computation offloading.

Taking into account the high mobility of vehicles, Zhou *et al.* [8] proposed a contract-matching approach for computation resource allocation and task assignment optimization to minimise the sum offloading delay of all V-UEs in a vehicular fog-computing system. Liu *et al.* [9] proposed a one-to-many matching-based task offloading algorithm to minimise the total offloading delay of all the V-UEs in a vehicular edge computing system. Wang *et al.* [10] proposed an ellipsoid method based computation offloading algorithm to jointly minimise the sum computation delay and the energy consumption of V-UEs in an MEC enabled vehicular network. Xu *et al.* [11] proposed a non-dominated genetic algorithm to minimise the total offloading delay of all V-UEs by jointly optimising the computation resource utilisation of multiple edge nodes. We note that, the above existing works ignored the queueing delay in their offloading models, but tasks may need to wait in queues at the computing servers before they are processed.

The queueing delay will increase the offloading delay of V-UEs and should be considered in making offloading decisions for V-UEs. In [12], a priority-aware task offloading scheme was proposed to minimise the overall offloading time of all IoT devices' tasks by leveraging multi-level feedback queueing in a fog-computing system, where the offloading decisions were made according to the priority of each task. The authors in [13] proposed to maximise the total sum-rate of an MEC-enabled

This work was supported in part by the European Union Horizon 2020 research and innovation program under the grant agreement No. 734798, in part by the Natural Science Foundation of China under Grant 61901367, in part by the Natural Science Foundation of Shaanxi Province under Grant 2020JQ-844, and in part by the Serving Local Special Scientific Research Project of Education Department of Shaanxi Province under Grant 21JC032.

vehicular network by jointly optimising the computation offloading decisions, power allocation and channel assignment, but neglected local processing or mobility of V-UEs.

In this paper, in order to enable low-latency, computation-intensive applications (e.g., live streaming video [14]) of V-UEs while guaranteeing fairness among them, we minimise the maximum service delay (which includes the transmission delay, queueing delay and processing delay) among all V-UEs in a vehicular network supported by a mixed fog/cloud computing system. More specifically, we propose a three-layer (vehicle-fog-cloud) computation offloading framework to model the service delays of V-UEs locally processing or offloading their computationally intensive applications to the cloud or fog servers, where the queues at the fog node (FN) and the cloud centre are modelled following the M/M/1 and M/M/C queueing models, respectively. Based on the modeled service delays, the maximum service delay among all V-UEs is minimised by devising a mobility and queueing-based offloading decision optimisation algorithm (MQA), which jointly optimises the computation offloading decisions of all the V-UEs while considering their mobility and potential queueing delays at the FN and cloud centre, in conjunction with a bisection method-based algorithm that optimises the FN computation resource allocation for the fog-processing V-UEs. The performance of the proposed algorithms is evaluated through simulations in comparison with the benchmarks, including pure local processing, fog processing, cloud processing, and random processing.

II. System Model

In this section, we first introduce the three-layer architecture of a vehicle-fog-cloud network, then derive the service delay of the local, fog and cloud processing modes, respectively.

A. Vehicle-Fog-Cloud Architecture

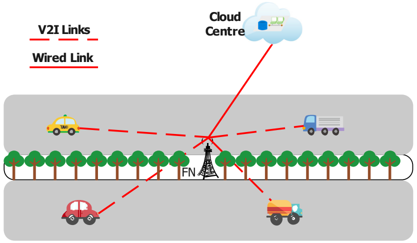

Fig. 1. System architecture of a three-layer computing system.

We consider a three-layer system model as shown in Fig.1. The vehicle layer is composed of V V-UEs, the fog layer includes a fog node (FN) collocated with a roadside unit (RSU), and the cloud layer is mainly a distant cloud centre with C cloud servers. Denote the set of V-UEs and cloud servers as $\mathcal{V} = \{1, 2, ..., V\}$ and $\mathcal{C} = \{1, 2, ..., C\}$, respectively. All V-UEs in coverage are connected to the FN by vehicle-to-infrastructure (V2I) wireless links, while the FN is connected to the cloud centre by a wired link.

For simplicity, we assume that each V-UE has one application to process and the road-trip time (RTT) can be ignored. A V-UE may process its application by itself (i.e., local processing) or offload the application to the FN or a cloud server for remote processing. Firstly, a V-UE sends an offloading request (including the information about its application, location, velocity, heading direction and channel condition) to the manager in the FN [7]. Upon receiving the request, the manager optimises the offloading decisions of all V-UEs considering the offloading requests of other V-UEs and the available resources in the FN and in the cloud centre.

The offloading decision for V-UE v is denoted by $x_v, y_v, z_v \in \{0, 1\}$, where $x_v = 1, y_v = 1, z_v = 1$ indicate that the application is processed by V-UE v itself, by the FN, or by a cloud server, respectively; otherwise, $x_v = 0, y_v = 0, z_v = 0$; and we have

$$x_v + y_v + z_v = 1, \ \forall v \in \mathcal{V}. \tag{1}$$

Denoting the offloading decisions matrix of all V-UEs by $\mathbb{O}$, where the vth row contains the offloading decision for V-UE v, we have

$$\mathbb{O} = \begin{bmatrix} x_1, & y_1, & z_1 \\ \vdots & \vdots & \vdots \\ x_V, & y_V, & z_V \end{bmatrix}_{V \times 3}. \tag{2}$$

B. Transmission Delay

If the application of a V-UE is locally processed, there is no transmission delay for this application. If the application of a V-UE is for remote processing, then the transmission delay for uploading the input data of the application to a computing server can be calculated according to the offloading decision.

Fog processing : If the application is to be processed by the FN, the maximum achievable transmission rate (in bit/s) from V-UE v to the FN for a selected channel is given by

$$r_{v,f} = W_f log_2 \left(1 + \frac{P g_{v,f}}{N_0}\right), \tag{3}$$

where W_f is the bandwidth of a selected resource block (RB) (in Hz) between a V-UE and the FN, in order to avoid severe interference, we assume that each V2I link is allocated an orthogonal RB, i.e., there is no interference between V2I links; P is the V-UE transmission power that is assumed to be the same for all V-UEs [12]; $g_{v,f}$ is the channel power gain from V-UE v to the FN [15]; and N_0 is the additive white Gaussian noise (AWGN) power at the FN.

The transmission delay from V-UE v to the FN is given by

$$T_{v,f}^{trans} = \frac{D_v}{r_{v,f}}, \tag{4}$$

where D_v is the data size (in bits) of V-UE v's application.

Cloud processing : If the application is to be processed by a cloud server, then the application is forwarded by the FN to the cloud centre. Given $r_{f,c}$ (in bit/s) as the transmission rate on the high-speed wired link from the FN to the cloud centre, the transmission delay from the FN to the cloud centre is given by

$$T_{v,f,c}^{trans} = \frac{D_v}{r_{f,c}}. \tag{5}$$

The total transmission delay from V-UE v to the cloud server is given by

$$T_{v,c}^{trans} = T_{v,f}^{trans} + T_{v,f,c}^{trans}. \tag{6}$$

C. Queueing Delay

We consider a time-slotted system, where $\mathbf{t} = \{0, 1, ..., t\}$ denotes the set of time slots and the length of each time-slot is Δt. When $t < 0$, both the queues at the FN and the cloud centre are empty.

$Fog\ processing$: The queue at the FN follows the M/M/1 queueing model. Based on Little's formula, the average queueing delay of an application in the FN queue is given by [16],

$$T_f^{wait} = \frac{\rho_f}{\mu_f - \lambda_f}, \tag{7}$$

where $\lambda_f \in [1, 3]$ is the rate of application arrival, which follows a Poisson process, μ_f is the service rate of the FN, and $\rho_f = \lambda_f/\mu_f < 1$ is the utilization rate of the FN.

$Cloud\ processing$: The queue at the cloud centre follows the M/M/C model [16]. We assume the same service rate for each cloud server, i.e., μ_{cloud}. The applications arrive at the cloud centre also following a Poisson process with the rate of λ_{cloud}. Then, the utilization rate of the cloud centre is given by $\rho_{cloud} = \lambda_{cloud}/(C\mu_{cloud}) < 1$, where it is assumed that each of the C cloud servers has enough capacity to process the received applications.

The average queueing delay of an application at the cloud centre is given by [16]

$$T_c^{wait} = \frac{(\alpha_{cloud})^C p_0}{C!(C\mu_{cloud})(1 - \rho_{cloud})^2}, \tag{8}$$

where $\alpha_{cloud} = \lambda_{cloud}/\mu_{cloud}$ and

$$p_0 = \frac{1}{\left(\sum_{n=0}^{C-1} \frac{(\alpha_{cloud})^n}{n!} + \frac{(\alpha_{cloud})^C}{C!\,(1 - \rho_{cloud})}\right)}. \tag{9}$$

D. Processing Delay

The application being processed by the V-UE locally, the FN, or a cloud server will lead to a different processing delay.

$Local\ processing$: Denote the local processing capability of V-UE v by f_v^{local} (in CPU cycles/s) , then the processing delay of local processing can be expressed as

$$T_v^{proc} = \frac{A_v}{f_v^{local}}, \tag{10}$$

where A_v represents the total number of CPU cycles are required to process the application of V-UE v and it is given by $A_v = D_v \lambda_v$, where λ_v is the processing density (in CPU cycles/bit) of the application.

$Fog\ processing$: If the application of V-UE v is processed by the FN, given the fog processing capability of V-UE v as f_v^{fog} (in CPU cycles/s), the fog-processing delay is given by

$$T_{v,f}^{proc} = \frac{A_v}{f_v^{fog}}. \tag{11}$$

$Cloud\ processing$: If the application is processed by a cloud server, denoting the cloud processing capability for V-UE v by f_v^{cloud} (in CPU cycles/s), then the cloud-processing delay can be expressed as

$$T_{v,c}^{proc} = \frac{A_v}{f_v^{cloud}}. \tag{12}$$

E. Service Delay

The service delay of an application may include the transmission delay, the queueing delay and the processing delay according to the offloading decision. Thus, the service delay of an application of V-UE v is given by

$$T_v = x_v T_v^{local} + y_v T_v^{fog} + z_v T_v^{cloud}, \tag{13}$$

where T_v^{local} is given in (10) as the application does not need to be transmitted to any remote server, and

$$T_v^{fog} = T_{v,f}^{trans} + T_{v,f}^{proc} + T_f^{wait}, \tag{13a}$$

$$T_v^{cloud} = T_{v,c}^{trans} + T_{v,c}^{proc} + T_c^{wait}. \tag{13b}$$

III. Problem Formulation and Proposed Algorithm

In this section, we first formulate the min-max service delay problem, then propose a mobility and queueing based offloading decision optimisation algorithm, in conjunction with a bisection method based FN computation resource allocation algorithm to solve the min-max problem.

A. Problem Formulation

We propose to minimise the maximum service delay among all V-UEs by jointly optimising the offloading decisions $\mathbb{O}$ and the FN computation resource allocation $\mathbf{f}^{fog} = \{f_1^{fog}, ..., f_V^{fog}\}$. To reduce the computation complexity, we offload applications via selected channels to neglect the communication resource allocation issue [12] and assume that the cloud computation resources for each V-UE are constant at the cloud centre [7]. Accordingly, we formulate the optimisation problem as follows,

$$\mathcal{P}1 : \min_{\mathbb{O}, \mathbf{f}^{fog}} \max_{v \in \mathcal{V}} T_v \tag{14}$$

$$s.t.\ \ x_v, y_v, z_v \in \{0, 1\},\ \forall v \in \mathcal{V}, \tag{14a}$$

$$x_v + y_v + z_v = 1,\ \forall v \in \mathcal{V}, \tag{14b}$$

$$\sum_{v \in \mathcal{V}} y_v f_v^{fog} \leq F^{fog}, \tag{14c}$$

$$0 \leq f_v^{local} \leq f_v^{fog} \lll f_v^{cloud},\ \forall v \in \mathcal{V}, \tag{14d}$$

$$y_v T_v^{fog} + z_v T_v^{cloud} \leq \tau_v,\ \forall v \in \mathcal{V}, \tag{14e}$$

where F^{fog} is the total computation capability in the FN and τ_v is the estimated service delay threshold for V-UE v; (14a) and (14b) are the constraints on the binary offloading decision indicators for each V-UE; (14c) requires that the total allocated computation resources at the FN cannot exceed its computation capability; (14d) indicates that for each V-UE, the amount of computation resource available at the cloud centre is the largest, followed by that at the FN, while that available for local processing is the smallest but should be non-negative; and (14e) indicates that the service delay should be kept below the estimated threshold for each V-UE.

B. Mobility and Queueing Based Offloading Decision Optimisation Algorithm

For any given allocation of computation resources at the FN, we propose a mobility and queueing based offloading decision optimisation algorithm (MQA), based on the traditional fireworks algorithm [17], to solve the formulated problem (14). The MQA is summarised in Algorithm 1. Firstly, we initialise

I random offloading decisions for all V-UEs ($\mathbb{O}_1^{(0)},...,\mathbb{O}_I^{(0)}$) in the solution space. In the meantime, we calculate the service delay threshold of V-UE v for remote processing based on its information (i.e., its position with respect to the FN, its moving direction and velocity) as follows,

$$\tau_v = \frac{d_v}{s_v},\ \forall v \in \mathcal{V}, \tag{15}$$

where d_v is the distance between V-UE v and the coverage edge of the FN in its direction of moving, and s_v is the velocity of V-UE v.

For all $v \in \mathcal{V}$ and $i = 1,...,I$, the estimated service delay of V-UE v based on the corresponding initial offloading decision in $\mathbb{O}_i^{(0)}$ for remote-processing, i.e., T_v^{fog} in (13a) and T_v^{cloud} in (13b), will be compared with its service delay threshold τ_v. If $max\{T_v^{fog}, T_v^{cloud}\} \leq \tau_v$, then the smallest estimated service delay between T_v^{fog} and T_v^{cloud} will be selected and the offloading decision for V-UE v in $\mathbb{O}_i^{(0)}$ will be updated accordingly. If $T_v^{fog} \leq \tau_v$ and $T_v^{cloud} > \tau_v$ ($T_v^{cloud} \leq \tau_v$ and $T_v^{fog} > \tau_v$), then T_v^{fog} (T_v^{cloud}) will be selected and the offloading decision for V-UE v in $\mathbb{O}_i^{(0)}$ will be updated to be fog processing (cloud processing). Otherwise, V-UE v can only process its application locally and the offloading decision for V-UE v in $\mathbb{O}_i^{(0)}$ is updated accordingly. After the all the initial offloading decision matrices ($\mathbb{O}_1^{(0)},...,\mathbb{O}_I^{(0)}$) have been updated, they become the initial fireworks, $\mathbb{O}_i^{(1)}, i = 1,...,I$.

In the lth iteration ($l = 1,...,L$, where L is the maximum allowed iteration), each firework $\mathbb{O}_i^{(l)}$ generates $\hat{s}_i^{(l)}$ new offloading decision matrices, which are called explosion sparks [17]. Each explosion spark of firework $\mathbb{O}_i^{(l)}$ is generated by randomly choosing α rows (where $\alpha \in [1, V)$) of $\mathbb{O}_i^{(l)}$ and performing left circular shift by 1 position on each chosen row, while the other $(V - \alpha)$ rows of the explosion spark are the same as the corresponding ones of firework $\mathbb{O}_i^{(l)}$. We take the objective function of (14) as the fitness function, i.e., $F(\mathbb{O}_i^{(l)}) = \max_{v\in\mathcal{V}} T_v(\mathbb{O}_i^{(l)})$, and the number of explosion sparks generated by firework $\mathbb{O}_i^{(l)}$ is given by

$$\hat{s}_i^{(l)} = ceil\left(M\frac{F_{max} - F(\mathbb{O}_i^{(l)}) + \epsilon_1}{\sum\limits_{i=1}^{I}(F_{max} - F(\mathbb{O}_i^{(l)})) + \epsilon_1}\right), \tag{16}$$

where $ceil(\cdot)$ denotes the ceiling function, M is a constant parameter for constraining the number of explosion sparks, $F_{max} = \max_i(F(\mathbb{O}_i^{(l)}))$, and ϵ_1 is an extremely small number to avoid zero division errors.

In addition to the explosion sparks, $\hat{m}$ ($1 \leq \hat{m} \leq I$) mutation sparks are generated by randomly selecting $\hat{m}$ fireworks from the I fireworks ($\mathbb{O}_1^{(l)},...,\mathbb{O}_I^{(l)}$) and randomly resetting some offloading decisions therein.

For each firework, explosion spark and mutation spark, the FN computation resource allocation is obtained by using the bisection method [18] (which will be presented in Section III-C), and accordingly the fitness function value is calculated. Among all the fireworks, explosion sparks and mutation sparks, the one with the smallest fitness value is selected as firework $\mathbb{O}_1^{(l+1)}$ for the next iteration. Denoting the set of all fireworks, explosion sparks and mutation sparks excluding $\mathbb{O}_1^{(l+1)}$ in the lth iteration by $R_{est}^{(l)}$ (i.e., $\mathbb{O}_1^{(l+1)} \notin R_{est}^{(l)}$), the other ($I$ - 1) fireworks ($\mathbb{O}_2^{(l+1)},...,\mathbb{O}_I^{(l+1)}$) are selected from $R_{est}^{(l)}$ according to the roulette wheel selection method [17], where the probability of $\mathbb{A}_n^{(l)}$ ($\mathbb{A}_n^{(l)} \in R_{est}^{(l)}, n = 1,...,|R_{est}^{(l)}|$) being selected is determined based on the Manhattan distance [19] as follows,

$$p(\mathbb{A}_n^{(l)}) = \frac{R(\mathbb{A}_n^{(l)})}{\sum\limits_{m=1}^{|R_{est}^{(l)}|} R(\mathbb{A}_m^{(l)})}, \tag{17}$$

where $R(\mathbb{A}_n^{(l)})$ is the sum of Manhattan distances between matrix $\mathbb{A}_n^{(l)}$ and all the other matrices in set $R_{est}^{(l)}$, which is given by

$$R(\mathbb{A}_n^{(l)}) = \sum_{m=1, m\neq n}^{|R_{est}^{(l)}|} \parallel \mathbb{A}_n^{(l)} - \mathbb{A}_m^{(l)} \parallel. \tag{18}$$

When the iteration converges or reaches the maximum allowed iteration, among all the fireworks, explosion sparks and mutation sparks, the one with the smallest fitness value is chosen as the optimal offloading decision $\mathbb{O}^*$ and the corresponding computation resource allocation at the FN returns the optimal fog computation resource allocation $\mathbf{f^{fog*}}$.

C. Fog Computation Resource Allocation

In the lth iteration, for each offloading decision matrix ($\mathbb{O}_1^{(l)},...,\mathbb{O}_I^{(l)}$), the problem in (14) reduces to the optimisation of computation resource allocation at the FN, i.e.,

$$\mathcal{P}2: \min_{\mathbf{f^{fog}}} \max_{v\in\mathcal{V}^{fog}} \frac{A_v}{f_v^{fog}} + \Delta_v \tag{19}$$
$$s.t.\ \ (14c), (14d),$$

where $\mathcal{V}^{fog}$ denotes the set of fog-processing V-UEs, for $\forall v \in \mathcal{V}^{fog}, x_v = z_v = 0, y_v = 1$, and $\Delta_v = D_v/r_{v,f} + T_f^{wait}$ is a constant. Letting $\Theta = \max_{v\in\mathcal{V}^{fog}}\{A_v/f_v^{fog} + \Delta_v\}$, the problem $\mathcal{P}2$ is converted to

$$\mathcal{P}3: \min_{\mathbf{f^{fog}}, \mathbf{\Theta}} \Theta \tag{20}$$
$$s.t.\ \ (14c), (14d),$$
$$\frac{A_v}{f_v^{fog}} + \Delta_v \leq \Theta,\ \forall v \in \mathcal{V}_{fog}. \tag{20a}$$

Since $A_v/f_v^{fog} \geq 0$ and based on (14c) and (20a), we have $\sum_{v\in\mathcal{V}^{fog}} A_v/(\Theta - \Delta_v) \leq \sum_{v\in\mathcal{V}^{fog}} f_v^{fog} \leq F^{fog}$. As $A_v/(\Theta - \Delta_v)$ is a monotonically decreasing function of Θ, the maximum service delay among all the fog-processing V-UEs is minimised when $\sum_{v\in\mathcal{V}^{fog}} A_v/(\Theta - \Delta_v) = \sum_{v\in\mathcal{V}^{fog}} f_v^{fog} = F^{fog}$, and $\mathcal{P}3$ can be converted to

$$\mathcal{P}4: \min_{\mathbf{\Theta}} \Theta \tag{21}$$
$$s.t.\ \sum_{v\in\mathcal{V}^{fog}} \frac{A_v}{\Theta - \Delta_v} = F^{fog}. \tag{21a}$$

We adopt the bisection method to solve problem $\mathcal{P}4$ as summarised in Algorithm 2, where V^{fog} is the number of fog-processing V-UEs; Θ^* denotes the minimum value of Θ, and the optimal fog computation resource allocation for V-UE v is given by $f_v^{fog*} = A_v/(\Theta^* - \Delta_v)$.

Algorithm 1 Mobility and Queueing Based Offloading Decision Optimisation Algorithm (MQA)

```
Generate $I$ random fireworks $\{\mathbb{O}_1^{(0)}, ..., \mathbb{O}_I^{(0)}\}$.
For each firework, randomly allocate fog computation resources.
for $v = 1 : V$ do
    Calculate the service delay threshold of V-UE $v$ using (15).
    for $i = 1 : I$ do
        Calculate the estimated service delay of V-UE $v$ using (13).
        if $min\{T_v^{fog}, T_v^{cloud}\} > \tau_v$ then
            The $(v, 1)$th element of $\mathbb{O}_i^{(0)}$ is substituted by $x_v = 1$.
        else if $T_v^{fog} < T_v^{cloud}$ then
            The $(v, 2)$th element of $\mathbb{O}_i^{(0)}$ is substituted by $y_v = 1$.
        else
            The $(v, 3)$th element of $\mathbb{O}_i^{(0)}$ is substituted by $z_v = 1$.
        end if
    end for
end for
return The updated fireworks as $\{\mathbb{O}_1^{(1)}, ..., \mathbb{O}_I^{(1)}\}$ .
Input: $l = 1, F^{(0)} = 0, \epsilon = 10^{-6}$.
while $l \leq L$ do
    for $i = 1 : I$ do
        For firework $\mathbb{O}_i^{(l)}$, run Algorithm 2 and calculate its fitness value.
        Calculate $\hat{s}_i^{(l)}$ according to (16).
        Generate $\hat{s}_i^{(l)}$ explosion sparks from firework $\mathbb{O}_i^{(l)}$.
        For each explosion spark, run Algorithm 2.
        Calculate the fitness value of each explosion spark.
    end for
    Generate $\hat{m}$ mutation sparks.
    For each mutation spark, run Algorithm 2.
    Calculate the fitness value of each mutation spark.
    The firework, explosion spark or mutation spark with the smallest fitness value is chosen as $\mathbb{O}_1^{(l+1)}$, and the smallest fitness value is denoted by $F^{(l)}$.
    if $|F^{(l)} - F^{(l-1)}| < \epsilon$ then
        break;
    else
        Fireworks $(\mathbb{O}_2^{(l+1)}, ..., \mathbb{O}_I^{(l+1)})$ are selected according to (17), (18).
        $l$ = $l$ + 1;
    end if
end while
return The optimal offloading decision $\mathbb{O}^* = \mathbb{O}_1^{(l+1)}$ if $l < L$, otherwise $\mathbb{O}^* = \mathbb{O}_1^{(L+1)}$, and the corresponding fog computation resource allocation $\mathbf{f^{fog}}^*$.
```

Algorithm 2 Fog Computation Resource Allocation

```
Initialise the precision $\epsilon_2 > 0$, $\Theta_{down} = \max_{v \in \mathcal{V}^{fog}} \Delta_v$
    and $\Theta_{up} = \sum_{v \in \mathcal{V}^{fog}} (A_v V^{fog}/F^{fog} + \Delta_v)$
repeat
    $\Theta = (\Theta_{up} + \Theta_{down})/2$.
    if $\sum_{v \in \mathcal{V}^{fog}} (A_v/\Theta - \Delta_v) > F^{fog}$ then
        $\Theta_{down} = \Theta$.
    else
        $\Theta_{up} = \Theta$.
    end if
until $|\Theta_{up} - \Theta_{down}| \leq \epsilon_2$.
$\Theta^* = |\Theta_{up} - \Theta_{down}|/2$.
Output: $\mathbf{f^{fog}}^*$
```

IV. Simulation Results

In this section, we present the simulation results. We assume a single cell scenario with one FN located in the centre of a 500 m × 500 m urban area as illustrated in Fig.1. There is a straight two-lane road (with one lane in each direction) passing through the middle of the considered square area, dividing the area into two equal rectangles. The width of each lane is 6 metres. Moreover, we assume that all the V-UEs are uniformly distributed in the rectangular area of 12m x 500m spanned by the two-lane straight road, where the movement direction of each V-UE is determined by the direction of the line that it locates in, and the local processing capability f_v^{local} is uniformly distributed in [50,400] M cycles/s. All parameter values used in the simulation are given in TABLE I [7], [15], [14], unless otherwise specified.

TABLE I
SIMULATION PARAMETERS [7], [15],[14]

Parameters	Value
Transmit bandwidth, W_f	180 kHz
Transmit power of V-UE v, P_v	200 mW
The noise power density at the FN, N_0	-174 dBm/Hz
Data size of an application of V-UE v, D_v	0.42 MB
Processing density of the application of V-UE v, λ_v	297.62 cycles/bit
Total computation capability of the FN, F^{fog}	2 G cycles/s
Cloud processing capability for V-UE v, f_v^{cloud}	5 G cycles/s
The service rate of the FN/cloud server, μ_f/μ_c	6
Wired link rate between the FN and the cloud, $r_{f,c}$	1 Mb/s
The number of cloud servers, C	2
The average vehicular velocity	70 km/h
The number of fireworks, I	6
The number of total explosion sparks, M	4
The number of mutation sparks, $\hat{m}$	1
The maximum number of iterations of FA, L	100

Fig.2 plots the maximum service delay $T_{service}$ versus the iterations of the outer loop in Algorithm 1. We can see that Algorithm 1 converges after the third iteration.

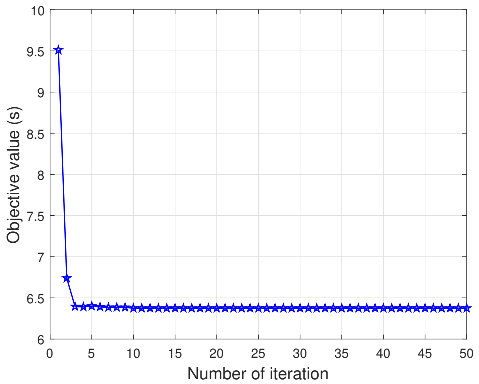

Fig. 2. Convergence of Algorithm 1, where $V = 5$.

Fig. 3 shows the maximum service delay versus the number of V-UEs, where 'MQA' denotes our proposed Algorithm 1, 'Local-Processing', 'Fog-Processing' and 'Cloud-Processing' denote the cases where all applications of the V-UEs are processed locally, by the FN, or by the cloud servers, respectively, and 'Random-Processing' denotes the case where each V-UE's application has the equal probability of being processed by itself locally, the FN, or a cloud server. We can see that the maximum service delay increases with the number of V-UEs in all the considered cases, among which MQA performs the best for any given number of V-UEs due to the joint optimization of offloading decisions for all the V-UEs while considering their mobility and queueing delays at the FN and cloud centre. When the number of V-UEs is lager than 7, fog-processing leads to the highest maximum service delay. This is due to the long queueing and processing delays caused by many applications sharing the limited computation capacity of the FN.

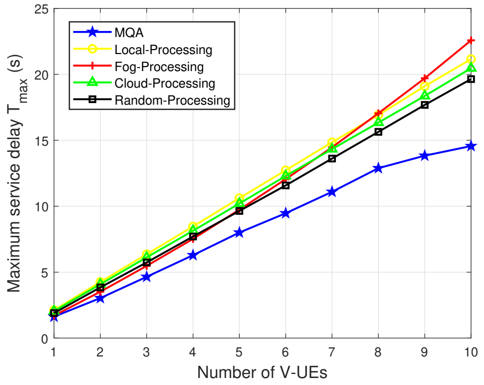

Fig. 3. Maximum service delay versus the number of V-UEs.

Fig. 4 and Fig. 5 show how the individual application's data size D_v and processing density λ_v affect the maximum service delay, respectively, where $V = 5$. We can see that the larger the data size or the higher processing density of each application, the higher the maximum service delay in each considered case. MQA always performs the best among all the considered cases. Moreover, local-processing is most significantly affected by a large data size or a high processing density of an application, due to the limited computation capability at each V-UE.

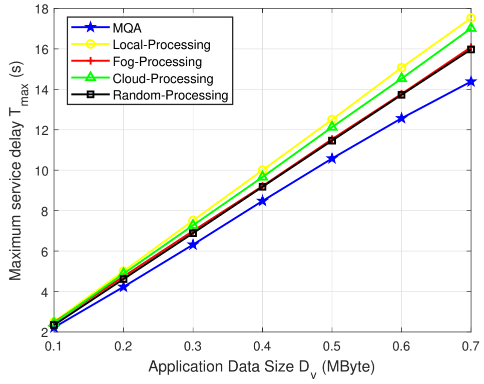

Fig. 4. Maximum service delay versus the data size of the application.

V. Conclusion

In this paper, we have proposed a mobility and queueing-based offloading decision optimisation algorithm, in conjunction with a bisection method-based FN computation resource allocation algorithm to minimise the maximum service delay of all V-UEs in an IoV system, where each V-UE may offload its task to a fog or cloud computing server or process it locally. The simulation results demonstrate that the proposed algorithms achieve a much lower maximum service delay than local-processing, fog-processing, cloud-processing, and random-processing.

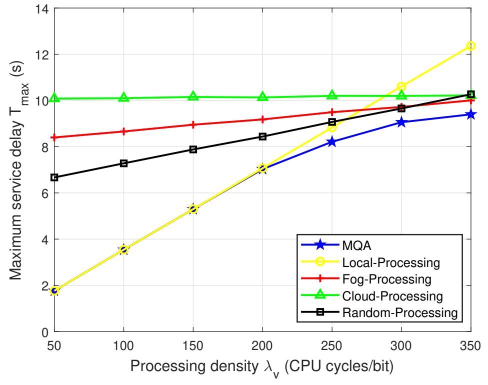

Fig. 5. Maximum service delay versus the processing density.

References

[1] Y. Yang, X. Luo, X. Chu, and M. T. Zhou, "Fog computing architecture and technologies," *Fog-Enabled Intelligent IoT Systems,Springer*, 2020.

[2] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Yla-Jaaski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE IoTJ*, pp. 4150–4161, June. 2018.

[3] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, April. 2017.

[4] Q. Liu, Z. Su, and Y. Hui, "Computation offloading scheme to improve qoe in vehicular networks with mobile edge computing," in *WCSP*, 2018.

[5] H. Feng, S. Guo, L. Yang, and Y. Yang, "Collaborative data caching and computation offloading for multi-service mobile edge computing," *IEEE Transactions on Vehicular Technology*, 2021.

[6] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.

[7] J. Du, L. Zhao, X. Chu, R. Yu, J. Feng, and C.-L. I, "Enabling low-latency applications in lte-a based mixed fog/cloud computing systems," *IEEE Transactions on Vehicular Technology*, pp. 1757–1771, 2018.

[8] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE TVT*, vol. 68, no. 4, pp. 3113–3125, April. 2019.

[9] P. Liu, J. Li, and Z. Sun, "Matching-based task offloading for vehicular edge computing," *IEEE Access*, vol. 7, pp. 27 628–27 640, April. 2019.

[10] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. S. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62 624–62 632, May. 2019.

[11] X. Xu, Y. Xue, X. Li, L. Qi, and S. Wan, "A computation offloading method for edge computing with vehicle-to-everything," *IEEE Access*, vol. 7, pp. 131 068–131 077, Sep. 2019.

[12] M. Adhikari, M. Mukherjee, and S. N. Srirama, "Dpto: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE IoTJ*, vol. 7, pp. 5773–5782, 2019.

[13] Z. Ning, X. Wang, J. J. Rodrigues, and F. Xia, "Joint computation offloading, power allocation, and channel assignment for 5g-enabled traffic management systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3058–3067, May. 2019.

[14] 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 TCOM*, vol. 66, no. 4, pp. 1594–1608, 2017.

[15] *Technical Specification Group Radio Access Network: Study on LTE-Based V2X Services (Rel. 14) 3GPP*, Std. TR 36.885 V2.0.0, Jul. 2016.

[16] D. Gross and C. M. Harris, "Simple markovian birth-death queuein models," in *Fundamentals of queueing theory, 3rd ed*, 1998, p. 53–115.

[17] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *International conference in swarm intelligence*. Springer, 2010, pp. 355–364.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[19] D.-J. Chang, A. H. Desoky, M. Ouyang, and E. C. Rouchka, "Compute pairwise manhattan distance and pearson correlation coefficient of data points with gpu," in *ACIS SNPD*, 2009, pp. 501–506.