IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 July 2023, accepted 28 July 2023, date of publication 7 August 2023, date of current version 15 August 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3303400*

RESEARCH ARTICLE

# TCOA: Triple-Check Offloading Algorithm for Roadside Units and Vehicular Microclouds in 5G Networks and Beyond

**BO-JUN QIU[1], (Student Member, IEEE), CHENG-YING HSIEH[1], (Student Member, IEEE), JYH-CHENG CHEN[1], (Fellow, IEEE), AND FALKO DRESSLER[2], (Fellow, IEEE)**

[1]Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
[2]School of Electrical Engineering and Computer Science, TU Berlin, 10623 Berlin, Germany

Corresponding author: Jyh-Cheng Chen (jcchen@ieee.org)

This work was supported in part by the National Science and Technology Council of Taiwan under Grant 112-2218-E-A49-023, Grant 112-2218-E-A49-021, and Grant 111-2221-E-A49-093-MY3; and in part by the Federal Ministry of Education and Research (BMBF), Germany, within the 6G Research and Innovation Cluster (6G-RIC) under Grant 16KISK020K.

**ABSTRACT** Next-generation intelligent transportation systems aim to achieve many cooperative perception and cooperative driving functions that require considerable computational resources. Offloading such tasks via mobile edge computing is considered part of the solution; this approach is currently being studied within the scope of 5G networks and beyond. In the automotive context, such edge systems could be roadside units (RSUs), which can easily be overloaded at peak times. Vehicular microcloud approaches have been proposed to overcome such problems by sharing the computational resources of nearby cars. In this study, we propose an offloading system architecture to enable such offloading in a vehicular microcloud interconnected by a 5G core network. We model the system as a queueing model to derive closed-form solutions for selected performance metrics. Based on our insights, we propose the triple-check offloading algorithm (TCOA) to obtain both the best offloading ratio of the vehicular microcloud to the entire offloading system and the optimal maximum number of the remaining vehicle instances in the vehicular microcloud. Our simulation results show that the proposed TCOA achieves better system performance than four other offloading schemes in terms of cost, response time, service rate, and cost response-time production service rate division (CRPSD).

**INDEX TERMS** Instance leaving, dynamic scaling, vehicular microcloud, RSU offloading, 5G networks.

## I. INTRODUCTION

In recent years, there has been an increasing number of vehicle-related applications, such as collision avoidance, that require considerable computational resources and low latency [1]. Because of the lack of computing resources in available vehicles, the calculations required for such applications cannot be completed in a short time. Thus, computing resources are one of the main limitations of vehicle-related applications. One popular solution is to offload the calculating tasks to a roadside unit (RSU) [2] deployed at the edge of a cellular network. The RSU can complete tasks independently or deliver them directly to a local edge server [3].

However, the performance of RSUs may degrade when they are used by many vehicles simultaneously [4]. One potential solution is to offload the calculating tasks to a remote server through an infrastructure network, such as a 5th generation (5G) core network with a 5G base station (gNB). The vehicular microcloud [5] has potential as a remote server because parked vehicles in a *vehicular microcloud* typically do not use their resources.

Compared with simply offloading to an RSU, further offloading to a remote vehicular microcloud could serve a greater number of users, thereby improving system

The associate editor coordinating the review of this manuscript and approving it for publication was Olutayo O. Oyerinde.

scalability. Unfortunately, benefits always entail overhead, i.e., the task response time may increase because of the distance between the users and the vehicular microcloud, and the cost may increase due to the use of the infrastructure network and the fee for parked vehicles. From the viewpoint of system operators, balancing the benefits and the overhead is essential.

In addition to the above trade-off between the benefits and overhead, we consider the scenario in which vehicles leave the microcloud. Because vehicles may be driven away by their owners, the parked vehicles in the vehicular microcloud may leave while they are processing a task. However, in the current studies and articles, no queueing models are used to formulate such a system directly. Thus, designing a dynamically scaling mechanism for vehicle instances is a significant challenge faced by vehicular microcloud operators.

In this study, we go beyond our earlier work in [6] and propose a system architecture formulated with a queueing model for this problem. Vehicle instances in our scheme can leave the system with no considerable effect because we retain some idle vehicle instances. In addition to performing mathematical analysis, we also use Network Simulator 2 (ns-2) to cross-validate the correctness of the closed-form metrics derived by our proposed queueing model. Additionally, we propose the triple-check offloading algorithm (TCOA) to obtain the offloading ratio of the remote vehicular microcloud and the maximum number of the remaining vehicle instances by observing the system performance. Finally, because the proposed TCOA is designed based on an objective function, operators can customize the weighting between cost and performance according to their needs.

The main contributions of this paper are as follows:

- **Challenge**: If we model the case in which any number of vehicle instances can leave the system at any time, the complexity of the mathematical model will be high. Existing models use approximation, so the results are not accurate.
  **Solution**: We propose a new model for the offloading system, in which any number of the vehicle instances can leave the system at any time. At the same time, the model still enables highly accurate system performance through a simple calculation. Furthermore, additional started vehicle instances are retained to reduce the impact of vehicles that leave the microcloud.
- **Challenge**: What are the best values for both the offloading ratio and the maximum number of remaining vehicle instances? **Solution**: We propose TCOA to analyze the best offloading ratio of the vehicular microcloud to the entire system and retain the optimal number of vehicle instances in the vehicular microcloud to balance the cost-effectiveness.
- **Challenge**: Many potential issues are new and have not been previously studied.
  **Solution**: We discuss many potential issues of this system and provide answers according to our current research results.

Thus far, the novelty of this study can be concluded:

- To our best knowledge, we propose the first model that adapts to the scenario in which vehicles may leave the vehicular microcloud, by taking the leaving feature into consideration in the initial design. In contrast, most previous studies assume the leaving/broken rate is low and that there is only one leaving instance at any time, which decreases the model complexity, however, not close to actual cases. Some studies have proposed high-complexity models with several leaving instances.
- Moreover, the proposed model achieves both high accuracy and low model complexity (short calculation time) by designing an instance set-up threshold. In contrast, previous models of the instance leaving/broken issue either require a long calculation time (high model complexity) to increase accuracy or make assumptions (which decrease the accuracy) to derive closed-form performance metrics.
- Additionally, the proposed retention-based model takes the benefits out of the instance-leaving features. The proposed algorithm finds the system's most effective operating configurations (best retained instance maximum and offloading rate) through detailed analyses.

For operators, offloading the computational tasks required by such vehicle-related applications from customers to the RSU and vehicular microcloud, through the proposed queueing model, they can immediately obtain each system metric by using the closed-form solution to develop their scheme/algorithm without actually deploying such large-scale experiments that take significant time and cost. Additionally, operators can use the proposed algorithm to obtain the optimal offloading configurations, including the offloading ratio $\beta$ and the maximum number of remaining instances $N$ in such an offloading system with the instance-leaving feature.

The rest of this paper is organized as follows. In Sec. II, we introduce the background. In Sec. III, we detail the proposed offloading system, the queueing model, and the proposed TCOA. The performance evaluation and experimental results are shown in Sec. IV. In Sec. V, we discuss the potential issues of the proposed system. Finally, we review related work in Sec. VI and summarize the paper in Sec. VII.

## II. BACKGROUND

The background for this paper comprises three parts: RSU, 5G core network, and vehicular microcloud.

1) *Roadside Unit (RSU):* An RSU is a unit beside the road that plays an important role in many vehicle-related applications. RSUs communicate with nearby vehicles and provide versatile services, such as calculating tasks [2], locating vehicles [7], transmitting information [8] or safety-related messages [9], and

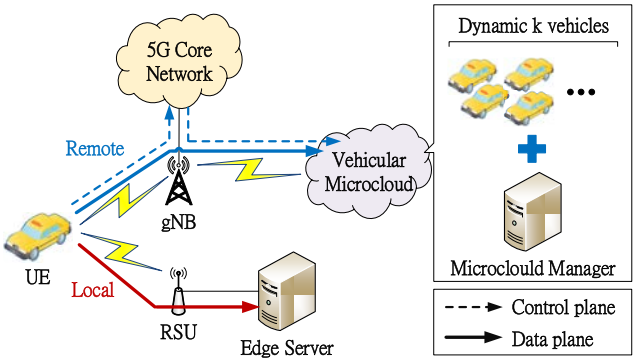



**FIGURE 1.** Proposed offloading system architecture.

downloading data [10]. To support these applications, an RSU should be able to compute complex calculations or directly connect to an edge server.

2) *5th Generation (5G) Core Network:* The 5G core network provides a service-based architecture (SBA) that separates network functions into a data plane and a control plane [11]. In the data plane, operators can deploy network functions close to the users to reduce latency. In addition, a 5G core network provides *ultra-reliable low-latency communication* (URLLC) that enables the use of networks with low latency and high reliability [12].
3) *Vehicular Microcloud:* A vehicular microcloud comprises vehicles, and other users can use the computing resources provided by the microcloud. In this paper, we focus on vehicles that are usually parked and unlikely to use their CPU resources [13], [14], [15]. Additionally, this approach improves the resource utilization of parked vehicles, which may obtain rewards (for example, monetary rewards). The scale of a vehicular microcloud is dynamic because parked vehicles may leave the microcloud; therefore, the computing ability of a vehicular microcloud is dynamic and depends on the number of vehicles used.

## III. PROPOSED OFFLOADING SYSTEM

The proposed offloading system architecture is illustrated in Fig. 1, and the notations are listed in Table 1. In the figure, a car may offload tasks to an RSU/edge server to reduce the task load of the car. In our proposed system, a car can also offload tasks to a vehicular microcloud in addition to an RSU. Here we refer to the car as user equipment (UE). We assume the task arrival rate follows a Poisson distribution with mean $\lambda$. When an RSU is overloaded, offloading tasks to the vehicular microcloud can increase the system service rate; however, offloading too many tasks to the microcloud will result in high costs for the parked vehicles in the microcloud.

We propose the TCOA to determine the offloading ratio $\beta$ and the maximum number of remaining vehicle instances $N$. Fig. 2 shows the flow of our proposed system model, which consists of the following parts:

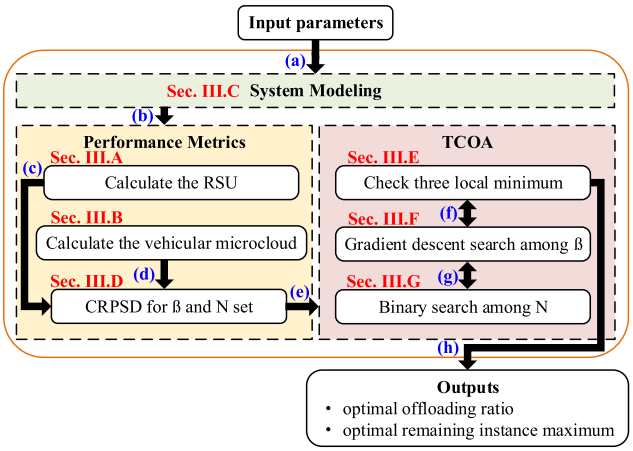



**FIGURE 2.** Input-process-output (IPO) model.

- *Input Parameters*: The required operating information includes the traffic, system capacities, instance abilities, system scale, instance leaving rate, instance set up rate, and weight factors.
- *System Modeling*: Based on the input parameters, we derive a mathematical model to provide a systematic method for system designers or service providers to tune and test different parameters without running simulations or actual deployment. This can save significant time and cost.
- *Performance Metrics*: To assess the performances of the system, the closed forms of the metrics, including cost, response time, and service rate are used. Then, an integrated metric, cost response-time production service rate division (CRPSD), is used to determine whether the result is outstanding according to the preferences of operators.
- *TCOA*: We propose TCOA according to the best interests of the system. TCOA searches for the best offloading ratio and the appropriate maximum number of remaining vehicle instances according to the CRPSD.
- *Outputs*: Based on TCOA, both the optimal offloading ratio and the maximum number of remaining vehicle instances are derived to balance the cost, response time, and service rate over the entire system.

As shown in Fig. 2, the complete procedure includes eight steps. First, step (a) describes the system behavior with the input parameters using the proposed queueing model. Fig. 2 also indicates that the system modeling is discussed in Sec. III-C. Second, steps (b)–(d) calculate each metric for the RSU and the vehicular microcloud to obtain the CRPSD. Third, in step (e), TCOA finds the best offloading ratio and the appropriate maximum number of remaining vehicle instances by checking three local minima. The details of each process in *Performance Metrics* are discussed in Secs. III-A, III-B, and III-D, respectively. Fourth, in steps (f)–(g), TCOA gradient descent is used to return the best offloading rate with the appropriate maximum number of remaining vehicle instances returned from the TCOA binary

**TABLE 1. List of notations.**

| Notation | Explanation |
|---|---|
| P | System performance |
| $\lambda$ | System task arrival rate |
| $\beta$ | Offloading task ratio to vehicular microcloud |
| C | Average system cost |
| $W_q$ | Average system response time in queues per task |
| S | Average system service rate |
| $\lambda_r$ | Task arrival rate in an RSU |
| $\mu_r$ | Service rate for each instance in an RSU |
| $K_r$ | The maximum number of tasks that can be accommodated in an RSU |
| $C_r$ | Average RSU cost |
| $c_r$ | Average cost for each instance in an RSU |
| $W_{qr}$ | Average RSU response time in the queue per task |
| $S_r$ | Average RSU service rate |
| $p_r$ | Multiple of RSU power (compared with a general vehicle) |
| $\lambda_v$ | Task arrival rate in vehicular microcloud |
| $\mu_v$ | Service rate for each instance in the vehicular microcloud |
| $K_v$ | The maximum number of tasks that can be accommodated in a vehicular microcloud |
| $C_v$ | Average vehicular microcloud cost |
| $c_v$ | Average cost for each vehicle instance in a vehicular microcloud |
| $W_{qv}$ | Average vehicular microcloud response time in the queue per task |
| $S_v$ | Average vehicular microcloud service rate |
| $o_v$ | Multiple of vehicle instance overhead (compared with a general vehicle) |
| k | The number of vehicle instances in the vehicular microcloud |
| $\alpha$ | Setup rate for each vehicle instance in the vehicular microcloud |
| $\gamma$ | Leaving rate for each vehicle instance in the vehicular microcloud |
| N | The maximum number of remaining vehicle instances in the vehicular microcloud |
| $\omega_1$ | Weight factor for C |
| $\omega_2$ | Weight factor for $W_q$ |
| $\omega_3$ | Weight factor for S |

search. Finally, step (h) obtains the global minimum and provides the optimal offloading ratio and the optimal maximum number of remaining vehicle instances. The details of each process in *TCOA* are discussed in Secs. III-E, III-F, and III-G, respectively.

### A. ROADSIDE UNIT (RSU)

In real scenarios, there may be many RSUs. In our analysis, without loss of generality, we consider that there is only one RSU (only one edge server, if any), and its calculation time per task follows an exponential distribution with mean $1/\mu_r$. The RSU always consumes resources because it does not turn off even when there are no tasks. If there are too many tasks, the RSU will buffer some of the tasks in the queue, ignoring the remaining tasks because of overloading. We will discuss the cases of multiple RSUs in Sec. V-D.

### B. VEHICULAR MICROCLOUD

We assume that the vehicular microcloud consists of $k$ vehicles parked in a parking lot. Each vehicle leaves the vehicular microcloud after parking, following an exponential distribution with a mean of $1/\gamma$. Once a vehicle leaves the vehicular microcloud, another parked vehicle joins the vehicular microcloud. The calculation time per task follows an exponential distribution with a mean of $1/\mu_v$ for each vehicle instance. If there are too many tasks, the vehicular microcloud buffers some of the tasks in the queue and ignores the remaining tasks.

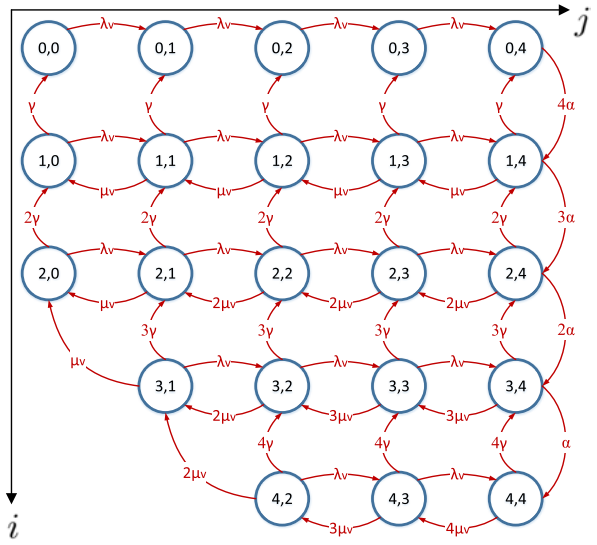
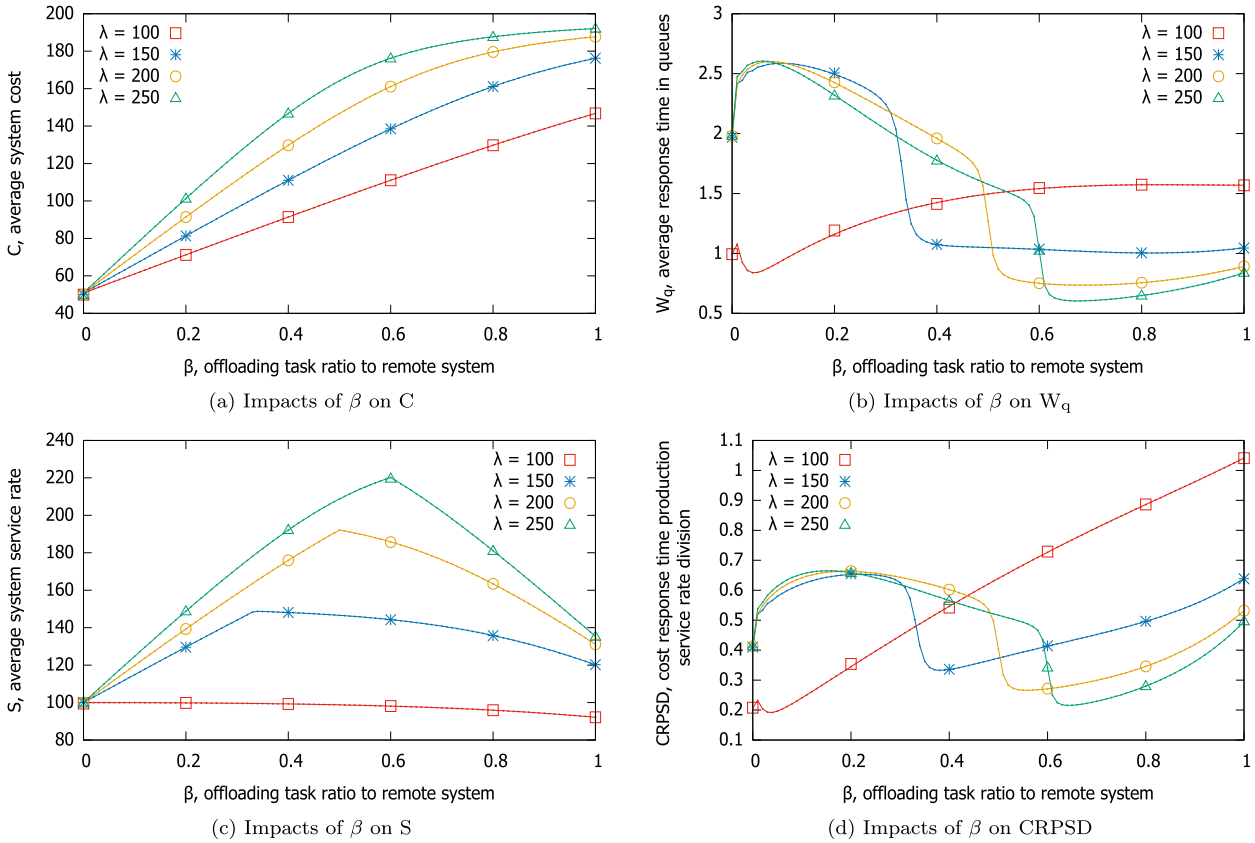



**FIGURE 3. State transition diagram in vehicular microcloud ($k = 4, K_v = 4, N = 2$).**

To reduce the impact of vehicles leaving the microcloud, we design dynamic scaling rules for the vehicular microcloud. As shown in Fig. 3, assuming there are $i$ vehicle instances and $j$ tasks in the queue, only if the offloading system fills the buffer with tasks does the microcloud manager use the nonrunning vehicle instances to reduce the response time. We assume that the setup time follows an exponential distribution with mean $1/\alpha$, and a vehicle instance in the setup state cannot process tasks but consumes resources. If the task buffer is not full, the vehicle instances in the setup state are turned off to save cost. In contrast, if the number of tasks is less than the number of vehicle instances that have started, and the difference in number between them is greater than the threshold $N$, the microcloud manager immediately turns off a running vehicle instance to save resources. In other words, we retain at most $N$ additional started vehicle instances in the system without calculating tasks to reduce the impact caused by a vehicle leaving the microcloud. When a vehicle leaves during a calculating process, the remaining part of the task is returned to the top of the task buffer for completion by one of the additional started vehicle instances. In contrast, if there are no additional started vehicle instances, the remaining task will wait at the top of the task buffer, and this increases the response time in the queue. For more design details, please refer to Sec. V.

### C. MODEL DERIVATION

To evaluate the system performance, we consider three metrics: the average cost of the RSU and vehicular microcloud,




(a) Impacts of $\beta$ on C

(b) Impacts of $\beta$ on $W_q$

(c) Impacts of $\beta$ on S

(d) Impacts of $\beta$ on CRPSD

**FIGURE 4.** Parameter study of the entire system.

the average response time in the queues per task, and the average number of tasks served. The system performance is defined as (1)

$$P = \frac{C^{\omega_1} \times W_q^{\omega_2}}{S^{\omega_3}}. \tag{1}$$

The definition of each notation is shown in Table 1.

The optimization goal of this paper is to make the offloading system perform with a cheaper cost, lower response time, and higher service scale. However, these aforementioned metrics belong to a trade-off, where a significant improvement in one metric may worsen other metrics, which is the primary constraint of this study. Therefore, we first define the objective function P to make a balance/fairness between these metrics to a certain degree, and second, operators can control the weight factors to reflect their preferences on this balance.

In this paper, we refer to $P$ as *cost response-time production service rate division (CRPSD)*: a smaller CRPSD indicates better system performance, such as lower cost, shorter response time, or a great number of serviced tasks. Before calculating the CRPSD, we normalize the three system metrics to a range from 0 to 1. Operators can alter the weight factors in (1) according to their own requirements. For example, if the operators consider the cost to be twice as important as the response time, they can set the weight of the cost $\omega_1$ to twice the weight of the response time $\omega_2$. In this study, we set all weights to 1 by default. For more weight factor settings, please refer to Sec. V-B.

The system metrics are a combination of the local metrics and remote metrics, which are defined in (2), (3), and (4).

$$C = C_r + C_v \tag{2}$$

$$W_q = \frac{S_r \times W_{qr} + S_v \times W_{qv}}{S} \tag{3}$$

$$S = S_r + S_v \tag{4}$$

The system uses the remote offloading ratio to offload tasks to the vehicular microcloud, and it offloads the remaining tasks to the RSU, as defined in (5) and (6).

$$\lambda_v = \lambda \times \beta \tag{5}$$

$$\lambda_r = \lambda - \lambda_v = \lambda \times (1 - \beta) \tag{6}$$

Using Equations (1) to (6), we describe the system performance with local and remote metrics. Then, we discuss the forms of these metrics to determine the CRPSD. Because the RSU (or the edge server, if any) is required to calculate tasks for many vehicles in typical scenarios, its calculating ability is $p_r$ times greater than that of a vehicle instance; the resource consumption of an RSU may also be $p_r$ times greater than that of a vehicle instance. However, the vehicular microcloud involves an additional fee and increases the cost of the vehicle instance by a factor of $o_v$. The microcloud operator can set both $p_r$ and $o_v$ according to the use case, and we assume

$p_r = 100$, $o_v = 2$ in this paper. Therefore, the cost ratio between an RSU and a vehicle instance is $p_r/o_v = 50$. We set both the average service rate $\mu_v$ and the average cost $c_v$ for each vehicle instance to 1 as a baseline.

In this paper, the cost of tasks increases to $o_v$ times because of the fee if the tasks are offloaded to a vehicular microcloud. Additionally, because the networks in 5G and beyond are URLLC, and the offloading tasks do not break down or cause damage to the vehicular microcloud if the number of tasks is less than the capacity $K_v$, we treat the transmitting/offloading cost as a constant $o_v$ instead of a variable of the remote offloading ratio $\beta$.

The RSU can be described as an M/M/1/K queueing model. Thus, we have the following equations:

$$C_r = c_r \times 1 = \frac{c_v \times p_r}{o_v} = 50 \tag{7}$$

$$S_r = \min(\lambda_r, \mu_r) = \min(\lambda_r, \mu_v \times p_r) = \min(\lambda_r, p_r) \tag{8}$$

$$W_{qr} = \frac{L_q}{\lambda_{eff}} \tag{9}$$

In Equation (9), $\lambda_{eff}$ denotes the effective arrival rate without blocking due to a full task buffer. The expected value of the task number in the queue is denoted by $L_q$. For the derivation of the M/M/1/K response time in the queue, please refer to Appendix A.

In this study, we chose an M/M/1/K model among several queueing models to describe the RSU for two reasons. One reason is that in IP networks, an M/M/1 model is enough for a single server system in most cases [16], [17], [18], [19]. The other reason is that our simulation matches the mathematical results, which means using an M/M/1-based model such as an M/M/1/K model is appropriate in this study.

The vehicular microcloud can be described by the proposed queueing model. To obtain the value of each metric, the probabilities $\pi$ of each state are essential. We define $\pi_{i,j}$ as the state probability with $i$ vehicle instances and $j$ tasks in the queue, where $I$ is the maximum of $i$ and $J$ is the maximum of $j$. For the derivation of the state probabilities, please refer to Appendix B.

After all state probability values are obtained, the metrics of the vehicular microcloud can be defined as follows: (10), (11), and (12).

$$C_v = c_v\Big(\sum_{(i,j)\in S_{space}} \pi_{i,j} i + \sum_{i=0}^{k} \pi_{i,K_v} \min(k-i, K_v - i)\Big) \tag{10}$$

$$S_v = \sum_{(i,j)\in S_{space}} \pi_{i,j} \min(i,j) \tag{11}$$

$$W_{qv} = \frac{\sum_{(i,j)\in S_{space}} \pi_{i,j} j}{\lambda_v(1 - \sum_{i=0}^{k} \pi_{i,K_v})} - \frac{1}{\mu_v} \tag{12}$$

Notably, the notation $S_{space}$ represents the set of all vehicular microcloud states. The closed forms of the three local and remote metrics are obtained from Equations (7) to (12). Therefore, the system performance metrics defined in Equations (1)–(4) can be derived.

### D. OBSERVATION AND ANALYSIS

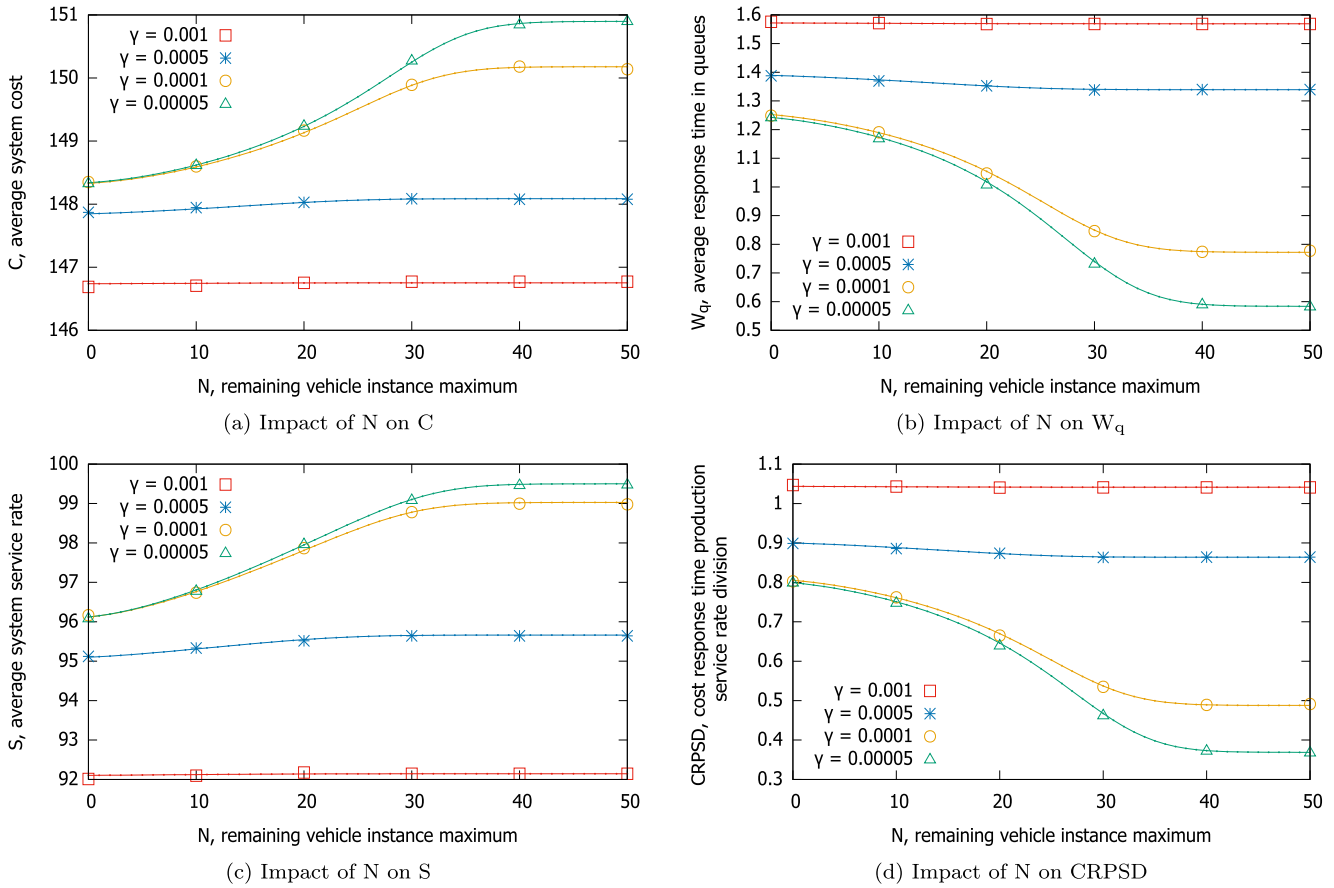
We calculate the system performance metrics defined in (1)–(4) and then use Network Simulator 2 (ns-2) [20] version 2.35 to implement and run the simulation of the proposed offloading system. All the simulations in this paper use the following default values: $K_r = 200$, $K_v = 250$, $k = 150$, and $\alpha = 0.02$.

The plots in Fig. 4 show the impacts of the offloading task ratio on the vehicular microcloud $\beta$ in terms of each metric with different arrival rates $\lambda$. Temporarily, we set the maximum number of remaining vehicle instances to $N = k/2 = 75$ and $\gamma = \alpha/20 = 0.001$ by default. In Fig. 4a, the cost increases as the value of $\beta$ rises because an increasing number of vehicle instances are set up to handle the incoming tasks.

In Fig. 4b, the response time in the queues $W_q$ increases initially because the RSU is overloaded and offloads a few tasks to the vehicular microcloud. The subsequent decreases are caused by additional vehicle instances being available to handle the tasks. The relief of the overloaded RSU causes $W_q$ to decrease drastically, but the low resource utilization of the RSU ultimately increases $W_q$. Furthermore, the global minimum decreases as the arrival rate increases because the capacity of the vehicular microcloud $K_v$ remains constant as an increasing number of vehicle instances are set up to service the tasks. When the vehicular microcloud is not overloaded, a higher arrival rate reduces the response time in the remote system queue $W_{qv}$.

In Fig. 4c, the service rate increases because tasks are appropriately distributed to both the RSU and the vehicular microcloud. In Fig. 4d, the best remote offloading ratio may be located at the local minimum found by a search from the middle value of $\beta$ because of the effects of the high service rate. Moreover, it may be located at the local minimum found by a search from $\beta = 0$, as the arrival rate is small enough to be handled by the RSU, as well as in some particular situations, in which the RSU can perform calculations rapidly without consuming a considerable amount of resources. In contrast, it may be located at the local minimum found by a search from $\beta = 1$ for the same reason. Therefore, the global minimum may be located at the local minimum found in a search from $\beta = 0$, $\beta = 1$, or the middle value of $\beta$, where the global maximum of the service rate is located.

The plots in Fig. 5 show the impact of the maximum number of remaining vehicle instances $N$ on each metric when several leaving rates $\gamma$ are used. As the values $N$ and $\gamma$ do not impact the RSU, we set $\beta = 1$ to focus on the effects on the vehicular microcloud. Because the number of vehicle instances in the vehicular microcloud is $k = 150$, we set the arrival rate $\lambda$ to 100 to assess the proposed dynamic scaling rules.




(a) Impact of N on C

(b) Impact of N on $W_q$

(c) Impact of N on S

(d) Impact of N on CRPSD

**FIGURE 5.** **Parameter study of the vehicular microcloud.**

In Fig. 5a, the cost increases because more started vehicle instances remain in the system. Furthermore, the increase becomes more intense as the leaving rate $\gamma$ decreases. When the tendency of a vehicle to leave decreases, the vehicle instance has a longer parking time during which it remains in the vehicular microcloud. Both of these trends can also be observed for the other metrics.

In Figs. 5b and 5c, the response time in the queue decreases and the system service rate increases, respectively. The reasons for this are the same as those explained in the previous paragraph.

In Fig. 5d, the decreasing trend intensifies initially, then slows, and finally becomes horizontal. The decrease part shows the benefits of retaining additional started vehicle instances, and the horizontal part shows that when $N$ exceeds a certain threshold, further increases in $N$ do not improve the performance because the frequency of setting up vehicle instances cannot be further decreased. This observation does not mean that a larger $N$ is necessarily good. Suppose the task arrival rate $\lambda$ does not follow a Poisson distribution, such as the case with burst traffic. In this case, a larger $N$ will cause many additional started vehicle instances to remain and consume resources. Although burst traffic may not occur frequently, $N$ should be chosen carefully to improve the system performance while remaining as small as possible.

Some brief conclusions can be drawn thus far:

- The best offloading ratio $\beta$ may be located at the local minimum found in a search from $\beta = 0$, $\beta = 1$, or the middle value of $\beta$, where the global maximum of the service rate is located.
- $N$ should be chosen carefully to improve the performances while remaining as small as possible.

### E. TRIPLE-CHECK OFFLOADING ALGORITHM (TCOA)

According to the observations in Sec. III-D, we propose TCOA, based on the value of CRPSD to make the fairness between system metrics to a certain degree, to obtain both the best task offloading ratio $\beta$ and an appropriate number of remaining vehicle instances $N$. In TCOA, gradient descent and binary search methods are used to determine the search results. In what follows, these algorithms are introduced, and a performance evaluation is presented.

As shown in Algorithm 1, according to the three possible local minima feature summarized in Sec. III-D with Fig. 4d, TCOA checks the three local minima found in a search from $\beta = 0.0$, $\beta = 1.0$, and the middle value of $\beta$, where the global maximum of the service rate is located, and then identifies the global minimum. A simple method can be used to determine the median value of $\beta$. We assign the RSU as many tasks as possible and offload the rest to the vehicular microcloud.

**Algorithm 1** Triple-Check Offloading Algorithm (TCOA)

**Input:** $P$, $\lambda$, $\mu_r$
**Output:** $\beta_{TCOA}$, $N_{TCOA}$
initial $\beta_{middle}$ to $(\lambda - \mu_r)/\lambda$
set $(\beta_v, N_v)$ by $TCOA_{GD}(-1, 1.0, 0.04, P)$
set $(\beta_m, N_m)$ by $TCOA_{GD}(1, \beta_{middle}, 0.02, P)$
set $(\beta_r, N_r)$ by $TCOA_{GD}(1, 0.0, 0.01, P)$
**if** $P(\beta_v)$ *smallest* **then**
  set $\beta_{TCOA}$ to $\beta_v$
  set $N_{TCOA}$ to $N_v$
**else if** $P(\beta_r)$ *smallest* **then**
  set $\beta_{TCOA}$ to $\beta_r$
  set $N_{TCOA}$ to $N_r$
**else**
  set $\beta_{TCOA}$ to $\beta_m$
  set $N_{TCOA}$ to $N_m$
end

Then, we define the current $\beta$ as the required middle value and denote it as $\beta_{middle}$. Note that when starting from $\beta_{middle}$, TCOA must search over a larger $\beta$ range to check the other choices with a high service rate, if any, because the RSU is overloaded and cannot handle any more tasks.

We use the function *TCOA gradient descent* to determine the local minimum. In the *TCOA gradient descent*, we set the search steps to 0.04, 0.02, and 0.01 as default values for each search. In each search step of a local minimum of the offloading ratio $\beta$, according to the strictly downward trend feature summarized in Sec. III-D with Fig. 5d, we use the function TCOA binary search to find the optimal remaining instance maximum $N$. With the current offloading ratio $\beta$, we can obtain the CRPSD in the current search point through the closed-form solution. The last part of TCOA compares the CRPSD metric with each search result and returns both the $\beta$ and $N$ values of the best result as the final answer.

In terms of time complexity, the optimization problem generates two parameters as the results, the offloading ratio $\beta$ and the maximum number of remaining instances $N$. Therefore, assuming the granularity of the set of $\beta$ is divided into $m$ choices, and there are $k$ vehicle instances in a vehicular microcloud, the time complexity of the optimization problem is $O(mk)$. In contrast, through observations, the proposed TCOA accurately uses the gradient descent three times (time complexity is generally $O(\log m)$) according to the three possible local minimum feature and a binary search (time complexity is generally $O(\log k)$) according to the strictly downward feature to reduce the time complexity of the optimization problem, which is the purpose and reason to propose TCOA. However, in the three-time gradient descent, one may be far from the starting point and a local minimum because there are only two local minima, basically. Therefore, in the worst case, it may search a relatively large range, which increases the time complexity to $O(m)$ for itself and $O(m \log k)$ for TCOA.

**Algorithm 2** TCOA Gradient Descent ($TCOA_{GD}$)

**Input:** $Sign$, $\beta_{init}$, $Step$, $P$
**Output:** $\beta_{TCOA_{GD}}$, $N_{TCOA_{GD}}$
initial $\beta_{current}$ to $\beta_{init}$, $P_{previous}$ to $double_{max}$
initial $Flag_{run}$ to True, $Flag_{border}$ to False
**while** $Flag_{run}$ **do**
  set $N_{current}$ by $TCOA_{BS}(\beta_{current}, P, 0, k)$
  **if** $P(\beta_{current}, N_{current})$ *worse* $P_{previous}$ **then**
    inverse $Sign$
    reduce $Step$
  set $P_{previous}$ by $P(\beta_{current}, N_{current})$
  **if** $Step$ *smaller* $Step_{min}$ **then**
    update $\beta_{current}$ by $Step_{min}$ with $Sign$
    set $N_{current}$ by $TCOA_{BS}(\beta_{current}, P, 0, k)$
    set $Flag_{run}$ to False
  **else**
    update $\beta_{current}$ by $Step$ with $Sign$
  **if** $\beta_{current}$ *larger 1.0* **then**
    set $\beta_{current}$ to 1.0
    **if** $Flag_{border}$ **then**
      inverse $Sign$
      reduce $Step$
    **else**
      set $Flag_{border}$ to True
  **else if** $\beta_{current}$ *smaller 0.0* **then**
    set $\beta_{current}$ to 0.0
    **if** $Flag_{border}$ **then**
      inverse $Sign$
      reduce $Step$
    **else**
      set $Flag_{border}$ to True
  **else**
    set $Flag_{border}$ to False
set $\beta_{TCOA_{GD}}$ to $\beta_{current}$
set $N_{TCOA_{GD}}$ to $N_{current}$
end

### F. TCOA GRADIENT DESCENT

The TCOA gradient descent shown in Algorithm 2 is used to determine the local minimum for the TCOA. To obtain the best $N$ value with the current $\beta$, we use a binary search method TCOA binary search and we select 0 to $k$ as the search range. In this study, we set the minimum search step $Step_{min}$ to 0.01 and reduce the search step by a factor of 2 by default. This algorithm repeats the following instructions in sequence:

- Use TCOA binary search to search for the best $N$ with the current $\beta$.
- If the current search feedback is worse than the previous feedback, change the search direction and reduce the search step.

**Algorithm 3** TCOA Binary Search ($TCOA_{BS}$)

**Input:** $\beta$, $P$, $N_{min}$, $N_{max}$
**Output:** $N_{TCOA_{BS}}$
**if** *horizontal* **then**
  set $N_{TCOA_{BS}}$ to $N_{min}$
**else if** $N_{min}$ *adjacent* $N_{max}$ **then**
  set $N_{TCOA_{BS}}$ to $N_{max}$
**else**
  set $N_{middle}$ by $Average_{up}(N_{min}, N_{max})$
  **if** $P(\beta, N_{middle})$ *equal* $P(\beta, N_{max})$ **then**
    recur $TCOA_{BS}(\beta, P, N_{min}, N_{middle})$
  **else**
    recur $TCOA_{BS}(\beta, P, N_{middle}, N_{max})$
end

- Record the search result for comparison with future results.
- If the search step is smaller than the minimum step, search using the minimum step, use TCOA binary search to obtain the best $N$, and return the current search result to TCOA. Otherwise, move a step.
- If the current $\beta$ value is greater than 1.0, set $\beta = 1.0$. If the next $\beta$ value is still greater than 1.0, change the search direction and reduce the search step.
- If the current $\beta$ value is smaller than 0.0, set $\beta = 0.0$. If the next $\beta$ value is still smaller than 0.0, change the search direction and reduce the search step.

### G. TCOA BINARY SEARCH

The TCOA binary search shown in Algorithm 3 is used to determine the appropriate value for the maximum number of remaining vehicle instances ($N$) for the TCOA gradient descent. When the trend in terms of the CRPSD decreases and then becomes horizontal with increasing $N$, this algorithm recursively finds the smallest value of $N$ in the horizontal part. The following steps are performed each time:

- If $N$ in the given range is horizontal, return the minimum of the given range $N_{min}$.
- If $N_{min}$ and $N_{max}$ of the given range of $N$ are adjacent, return the maximum $N_{max}$.
- Average $N_{min}$ and $N_{max}$ to obtain the middle value $N_{middle}$, and carry $N_{middle}$ if it is not an integer.
- If the middle value $N_{middle}$ is equal to the maximum $N_{max}$ in terms of the CRPSD metric with the input offloading ratio $\beta$, repeat the TCOA binary search with $N_{min}$ and $N_{middle}$ as the new range for $N$. Otherwise, repeat the TCOA binary search with the new range $N_{middle}$ to $N_{max}$.

## IV. EVALUATION AND COMPARISON

To the best of our knowledge, no other studies have proposed a model that can accommodate, without assumptions, any number of instances that leave the system at any time. In other words, to date, the features and requirements of vehicular microclouds have not been met, except in this paper. Therefore, we compare the proposed TCOA with four baseline schemes: *Local*, *Remote sensitive*, *Remote nonsensitive*, and *Intuitive*.

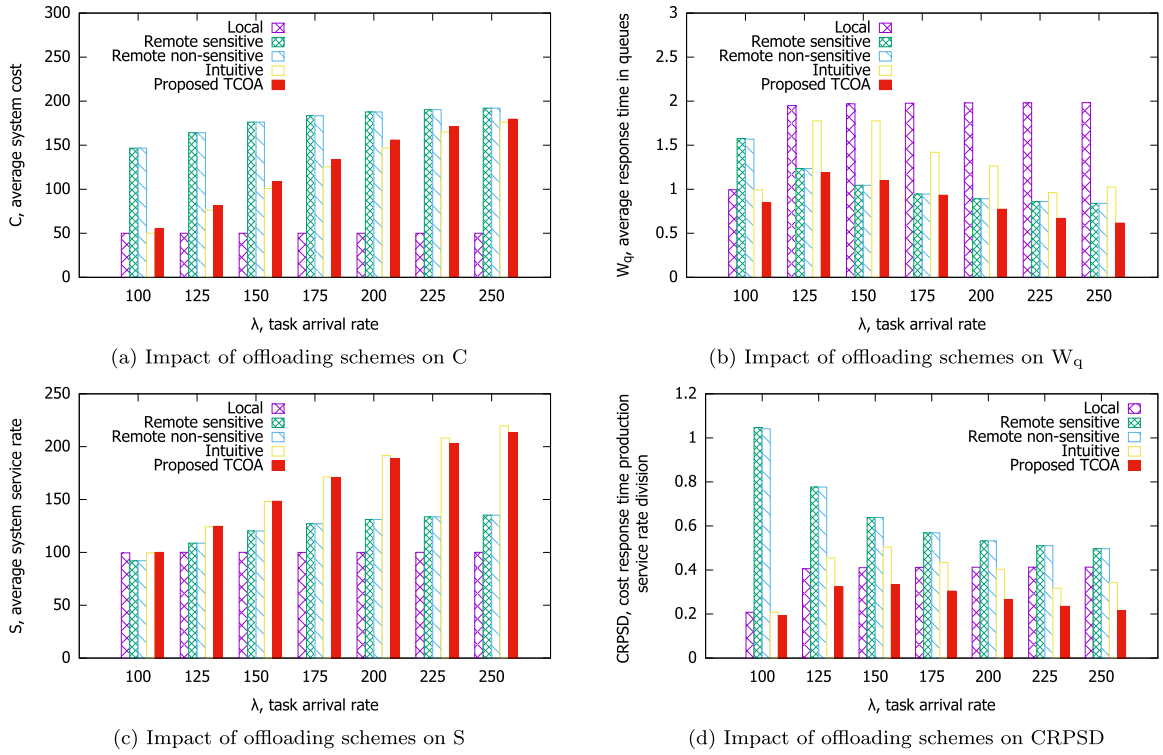
The *Local* scheme offloads all the tasks to the RSU, whereas the *Remote sensitive* scheme offloads all the tasks to the vehicular microcloud and retains no additional started vehicle instances ($N = 0$), so it is expected to be sensitive to departing vehicle instances. In contrast, the *Remote nonsensitive* scheme offloads all tasks to the vehicular microcloud but retains as many started vehicle instances as possible ($N = k$), so it is expected to be less sensitive to departing vehicle instances. The *Intuitive* scheme uses an intuitive method, whereby it allows the local arrival rate $\lambda_r$ to be equal to the RSU service rate ($\mu_r$) and offloads the remaining tasks to the vehicular microcloud to prevent the RSU from becoming overloaded, thereby serving users to the greatest possible extent. Finally, the *Intuitive* scheme sets the maximum number of remaining vehicle instances to $k/2 = 75$ by default. The simulation results are depicted in Fig. 6 and Fig. 7.

Fig. 6 shows the impacts of each offloading scheme for several task arrival rates $\lambda$. We set the leaving rate for each vehicle instance to 0.001 by default. In Fig. 6a, the results of the *Local* scheme are constant because it uses only the RSU. Therefore, the cost is $c_r = 50$. In contrast, both *Remote* schemes have higher costs because they do not use the RSU and thus use more vehicle instances to handle the tasks. The *Intuitive* scheme performs slightly better than the TCOA because it requires the RSU to perform more tasks, and thus it requires fewer vehicle instances.

In Fig. 6b, the *Local* scheme has the longest response time in the queue because the task buffer of the RSU is full and the RSU is overloaded. In contrast, the response time of both *Remote* schemes decreases with an increase in the number of tasks. In the vehicular microcloud, more tasks lead to setting up more vehicle instances and to consuming the fixed-size task buffer within a shorter time. For the same reason, the trend of the *Intuitive* scheme increases initially because of the low task arrival rate to the vehicular microcloud $\lambda_v$. Although the trend of the proposed TCOA is similar to that of the *Intuitive* scheme, it performs much better because more vehicle instances are used. Thus, the RSU is not overloaded.

In Fig. 6c, the *Local* scheme performs the worst because the RSU can handle at most 100 tasks, on average, according to the service rate $\mu_r$. Both *Remote* schemes have the same limitation that the vehicular microcloud can handle at most 150 tasks, on average, according to the total service capacity of the service rate of the vehicle instances ($k \times \mu_v$). Compared with the proposed TCOA, the *Intuitive* scheme has a slight advantage because the proposed TCOA may ignore some tasks instead of asking the RSU to finish them to avoid overloading the RSU.

In Fig. 6d, the *Local* scheme performs better than the *Intuitive* scheme when the task arrival rate $\lambda$ is less than or equal to 175, whereas the former shows worse performance when $\lambda$ is greater than or equal to 200. Therefore, the




(a) Impact of offloading schemes on C

(b) Impact of offloading schemes on $W_q$

(c) Impact of offloading schemes on S

(d) Impact of offloading schemes on CRPSD

**FIGURE 6.** Impact of offloading schemes with diverse arrival rates.

vehicular microcloud performs more efficiently with a higher task arrival rate. In contrast, although both *Remote* schemes also show a downward trend, they perform worse because of the zero usage rate of the RSU. In addition, similar behavior is caused by the high instance leaving rate. Finally, the proposed TCOA performs the best because a good balance between metrics is achieved.

The plots in Fig. 7 show the impacts of each offloading scheme with different instance leaving rates $\gamma$. We set the task arrival rate to $k = 150$ by default. In Fig. 7a, the *Local* scheme has the lowest cost because no additional fees are paid to the vehicular microcloud. The cost of the proposed TCOA is slightly higher than that of the *Intuitive* scheme because of the greater number of started vehicle instances. Furthermore, regarding the *Remote* schemes, the *nonsensitive* scheme consumes slightly more resources than the *sensitive* scheme because of the low leaving rate. When the instance leaving rate is low, the additional started vehicle instances remain for a longer period of time, resulting in higher costs. However, the cost difference is small because the former retains additional instances, whereas the latter sets up vehicle instances more frequently.

In Fig. 7b, the *Local* scheme shows a long response time in the queue because the RSU is overloaded. The other schemes have lower response times with smaller instance leaving rates because the vehicular microcloud is more stable and has a longer parking time to handle tasks. In Fig. 7c, the *Intuitive* scheme and the proposed TCOA have almost the same service rate, regardless of the value of the instance leaving rate, because the RSU handles many tasks, and the burden on the vehicular microcloud is not heavy. In contrast, the two *Remote* schemes ask the vehicular microcloud to handle all tasks, where the number of tasks is similar to the number of all vehicle instances ($k$). It is difficult to ask all vehicle instances to handle tasks because some may leave and some require time to set up. This challenge is overcome when the vehicle instance leaving rate decreases, thereby increasing the maximum number of tasks calculated on average.

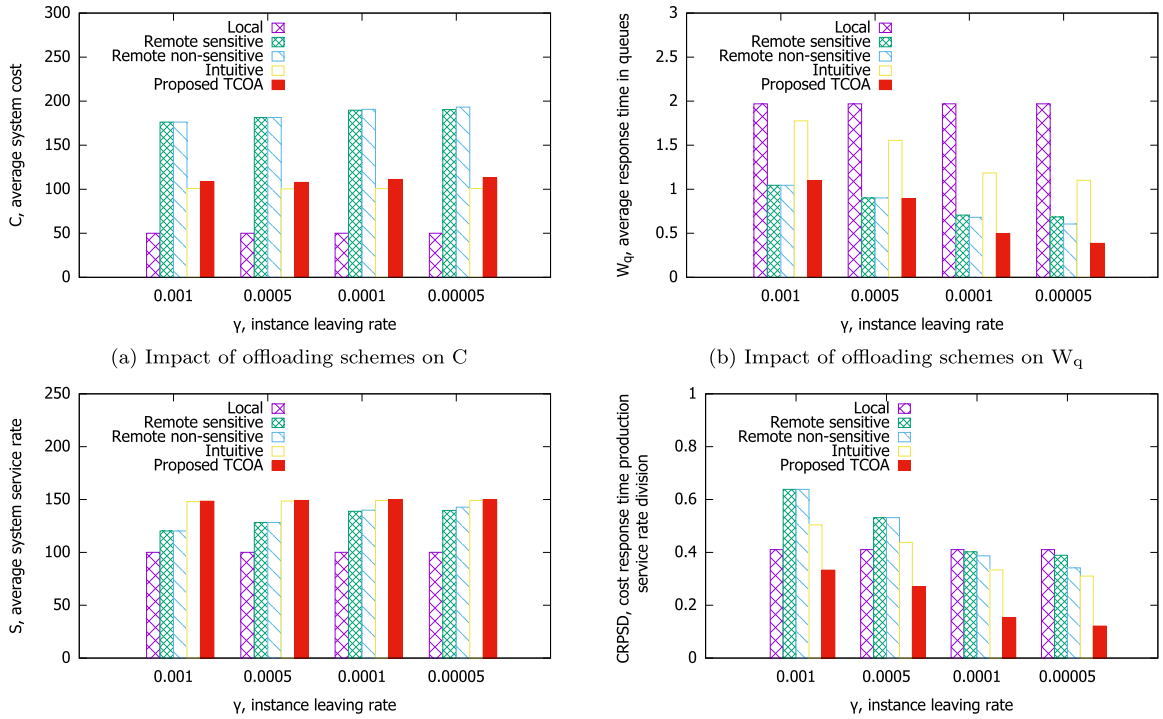
In Fig. 7d, all the schemes, except for the *Local* scheme, perform better with a lower instance leaving rate, and this trend is affected mainly by the response time metric. The proposed TCOA shows the best performance. Even in a harsh environment, such as a high instance leaving rate, within a reasonable range, the TCOA can improve the system performance by finding the best task offloading ratio $\beta$ and the proper maximum number of remaining vehicle instances $N$.

According to the above evaluations, the reasons the proposed TCOA always performs the best in terms of the CRPSD metric can be summarized as follows:

- The proposed TCOA tends to ignore excess tasks to avoid overloading the RSU.
- The proposed TCOA sometimes offloads more tasks to the vehicular microcloud to reduce the response time in the queue $W_q$.

## V. DISCUSSION

Our proposed system has some potential issues, which we separate into the candidate to turn off, weight factor setting,




(a) Impact of offloading schemes on C

(b) Impact of offloading schemes on $W_q$

(c) Impact of offloading schemes on S

(d) Impact of offloading schemes on CRPSD

**FIGURE 7.** Impact of offloading schemes with diverse leaving rates.

RSU cost ratio, many-to-many offloading, non-cooperative selfish scenario, result correctness, and the vehicle setup threshold. In this section, we discuss these issues.

### A. CANDIDATE TO TURN OFF

In Sec. III, if the number of tasks is less than the number of started vehicle instances by threshold $N$, a started vehicle instance is turned off immediately to save resources. In this situation, which vehicle instance should be turned off is an interesting question. According to the queueing model, if we do not want to affect the leaving rate, we should turn off vehicle instances in a balanced manner among the various parking times. To serve this requirement, the vehicle instance with the shortest living time in the vehicular microcloud is a prospective candidate. In addition to the instance with the shortest living time in the vehicular microcloud, the instance with the longest living time should be considered. In addition, there are many possible candidates, such as the closest and furthest instance from the end of the parking time (leaving). As shown in Fig. 8, different candidates show the same performance with an appropriate $N$, such as 30. Therefore, the candidate does not matter for the proposed offloading architecture and TCOA.

### B. WEIGHT FACTOR SETTING

In Sec. III, we set all weights to 1 by default and stated that the operators could modify them according to their preferences. This section shows what happens if some weights are not 1. We list three settings as examples and observe the differences in terms of metrics. As shown in Table 2, *Setting 1* is the baseline, in which $\omega_1$, $\omega_2$, and $\omega_3$ are all $1s$ and the other parameters are set as described in Sec III-D with the arrival rate $\lambda = 250$. In contrast, *Setting 2* considers the cost, so $\omega_1$ is increased to 2. Furthermore, in addition to the cost, *Setting 3* considers the response time in the queues, so $\omega_2$ is also increased to 2.

In Fig. 9, the global minimum of *Setting 2* is $\beta = 0$ because the lowest cost is incurred, as no vehicle instances are running. On the other hand, although the global minimum of *Setting 3* is still in the middle, its value of $\beta$ increases by 0.01 compared with the baseline (*Setting 1*). The reason for this is that compared to the other metrics, the weight of the service rate is smaller, which has less impact on CRPSD; moreover, the global maximum of the service rate is $\beta = 0.6$, whereas the global minimum of the response time is $\beta = 0.68$. In Table 2, *Setting 2* has a lower cost compared with the baseline, whereas its performance is worse in terms of the other metrics. Additionally, *Setting 3* performs better in terms of response time $W_q$, whereas it is slightly worse in terms of service rate $S$ and cost $C$.

Thus far, we can draw a brief conclusion: the setting of the weight factors affects the CRPSD, thus further impacting the $\beta$ obtained by the proposed TCOA. Therefore, operators need to set the weight factors carefully, and the suggested value is 1 if there is no preference. Although the weight factors genuinely affect the CRPSD, the proposed TCOA can correctly derive both the best offloading ratio $\beta$ and the optimal maximum number of remaining vehicle instances

**TABLE 2. Weight settings and impacts.**

| Setting | $\omega_1$ | $\omega_2$ | $\omega_3$ | $\beta$ | C | $W_q$ | S |
|---|---|---|---|---|---|---|---|
| Setting 1 | 1 | 1 | 1 | 0.64 | 179.57 | 0.62 | 213.4 |
| Setting 2 | 2 | 1 | 1 | 0 | 50 | 1.98 | 100 |
| Setting 3 | 2 | 2 | 1 | 0.65 | 180.3 | 0.61 | 211.6 |

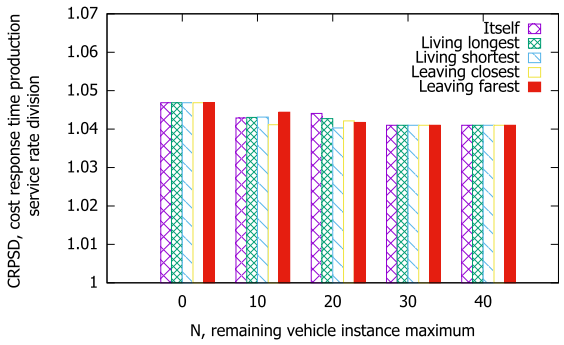



**FIGURE 8. Impact of the turned-off candidate with various remaining maxima.**

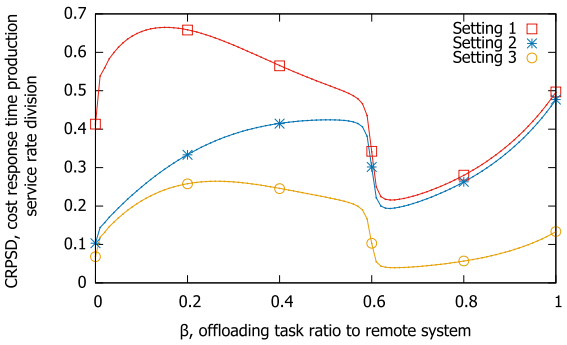



**FIGURE 9. Impact of weight factor settings with various offloading ratios.**

$N$ according to the CRPSD, regardless of the weight factor settings.

### C. RSU COST RATIO

As indicated in Equation (7) in Sec. III-C, the cost of an RSU $c_r$ is defined by three variables, including a multiple of the RSU power $P_r$, multiple of vehicle instance overhead $o_v$, and cost of a vehicle instance $c_v$, which is set to 1 as a baseline. To observe the effects of $c_r$ (the cost ratio between an RSU and a vehicle instance), we manipulated $o_v$ instead of $P_r$ to avoid affecting the computational ability of the RSU. The results are presented in Fig. 10. When $c_r$ is low, the optimal offloading ratio $\beta$ is decreased to zero because the RSU is highly cost-efficient. In contrast, with a higher $c_r$, all values increase, especially for a lower offloading ratio $\beta$. Thus far, we know that the value of $c_r$ affects the offloading system; however, no matter what the value $c_r$ is, the proposed TCOA can always determine the optimal results for the offloading system.

### D. MANY-TO-MANY OFFLOADING

In Sec. III, we analyze only the one-to-one offloading scenario, which has one RSU and one vehicular microcloud.

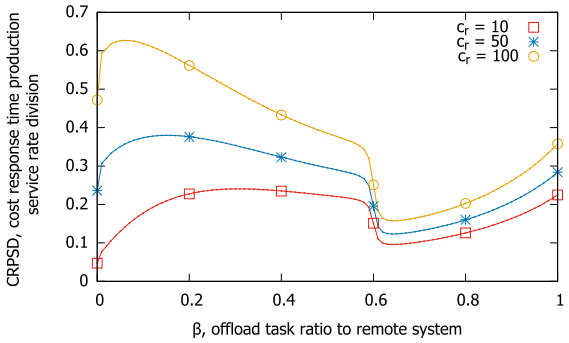



**FIGURE 10. Impact of RSU cost ratios with various offloading ratios.**

How to offload tasks among several RSUs and vehicular microclouds is the next challenge. By observation, we note that the RSU may become overloaded when it handles too many tasks, and the threshold is approximately 90 percent of its service rate. On the other hand, the vehicular microcloud prefers to take on many tasks to reduce the response time in the queue. The overload threshold still exists, and it is determined by the instance leaving rate. Temporarily, we assume that the overload threshold is also approximately 90 percent; then, a simple offloading policy can be determined as follows.

- If the usage rate of the RSU is less than 90 percent, ask the RSU to manage all tasks.
- If the usage rate of the RSU is close to 90 percent, offload the remaining tasks to the vehicular microcloud.
- Choose the vehicular microcloud with the largest scale $k$.
- Offload tasks to the chosen vehicular microcloud.
- If the usage rate of the chosen vehicular microcloud is close to 90 percent, offload the remaining tasks to the next chosen vehicular microcloud.
- If a task is ignored because a task buffer is full, offload the task to the previous or the next chosen vehicular microcloud.

By means of the above offloading policy, several vehicular microclouds can cooperate with each other so that tasks are not ignored because of a full buffer.

### E. NON-COOPERATIVE SELFISH SCENARIO

The non-cooperative scenario is closer to the actual conditions for operating such an offloading system. In the non-cooperative scenario, we have two solutions to motivate all the nodes (cars, edge servers, and vehicular microclouds) to join the operation of the offloading system and further follow the offloading algorithm.

One solution is that operators use incentive mechanisms with rewards [21], [22]. For example, at the beginning of each time slot, the operator asks all possible edge servers and vehicular microclouds (computational nodes) to obtain their prices of computational resources; according to the total amount of computational requirements from the cars (customers), the operator selects the cheaper computational nodes to serve the customers.

The other solution is that operators perform as a trade-matching platform, which presents the prices of computational resources. Specifically, computational nodes register the prices of the computational resources to the operator, and the operator displays all reported prices; customers select the offloading destination by themselves when they need the offloading services. Furthermore, according to the game theory, the operator can add an additional fee to the price of each computational node to influence the customers' offloading decisions.

### F. RESULT CORRECTNESS

After the calculation, the vehicle instance returns the result. However, users may worry about whether the calculated answer is correct. Thus, a simple quality-of-service (QoS) policy is required. To ensure a high QoS level, the vehicular microcloud distributes the same task to several vehicle instances and compares whether the calculated results are similar before returning the result to the user. In this way, we can guarantee the quality of the calculation to a large extent.

### G. VEHICLE SETUP THRESHOLD

In Sec. III, the dynamic scaling rules set up vehicle instances only when the task buffer is full. This design can substantially reduce the complexity of the queueing model. If the vehicle setup threshold is smaller than $K_v$, the probability of each state cannot easily be derived in sequence because the transitions are complicated.

In this situation, a matrix analysis method is required. However, assumptions and approximations rather than accurate values are involved in the calculation, which gives rise to the results of the closed-form solution being different from the simulation results. In addition, the high computational complexity increases the time required for the calculation because the calculation scale is considerable. For example, if the value of $k$ is 150, the size of the matrix is $150 \times 150 = 22500$ elements.

Furthermore, because the Denman-Beavers iterative algorithm [23] must be used to solve a quadratic matrix, which contains negative elements, the accuracy of the result depends on the number of iterations of the calculation. Therefore, the time required for the calculation is further extended, whereas the obtained result is only an approximation.

A conclusion can thus be drawn: a good design for the dynamic scaling rules of a vehicular microcloud should comprehensively consider the system performance, the calculation accuracy, and the time required for the calculation.

## VI. RELATED WORK

There are three main categories of related research, namely, edge traffic offloading [2], [24], [25], [26], [27], [28], [29], dynamic scaling systems [30], [31], [32], [33], [34], and server breakage and repair [35], [36], [37], [38], [39].

### A. EDGE TRAFFIC OFFLOADING

Guo et al. [24] proposed an algorithm based on deadlines to determine whether to offload and then deliver tasks to an edge server or to the cloud according to the cost involved. Li [25] focused on the offloading decision and wireless scheduling for an edge server and many mobile devices. Zhan et al. [2] used deep learning to determine when and how to schedule offloading tasks to RSUs along a road. Xu et al. [26] used vehicle-to-vehicle communications to offload data traffic to other vehicles by WiFi to deliver data traffic to the Internet. Dai et al. [27] focused on a relay scheme to determine when to offload, and which vehicle cloudlets to offload to. Zhou et al. [28] formulated the offloading and service caching problem in an edge computing-based smart grid as a mix-integer non-linear program (MINLP) to minimize the system cost. Specifically, they decomposed the optimization problem and proposed gradient descent and game theory-based algorithms for resource allocation and computing strategies. Zhou et al. [29] focused on the offloading problem in a three-tier mobile-cloud-edge scenario and used a deep reinforcement learning-based mechanism to optimize the offloading, service-caching, and resource-allocating strategies. Specifically, they formulated the optimization problem as an MINLP and proposed an asynchronous advantage actor-critic-based (3AC-based) algorithm as a solution.

In this study, we focus on determining the offloading ratio to remote vehicular microclouds using a queueing model and the proposed algorithm. Additionally, rather than focusing on a method for offloading traffic to RSUs, we enable tasks to be directly offloaded to a vehicular microcloud through the gNBs and the 5G core network. Moreover, because the vehicle instances in a vehicular microcloud may leave without expectations, the proposed queueing model presents this unique behavior and has different features from the other cloud-based clusters.

### B. DYNAMIC SCALING SYSTEM

Song et al. [30] proposed a hybrid algorithm to decide when and how many instances should be reserved to improve resource utilization and reduce the service cost in geo-distributed clouds. Ma et al. [31] provided a fast approximation algorithm for static request admissions and an online algorithm for dynamic request admissions to cost-efficiently enable virtualized network functions (VNFs). Phung-Duc et al. [32] proposed an algorithm to address the budget-performance trade-off for cloud systems. Rahman et al. [33] proposed a negotiation game-based service chain autoscaling method that considered computation, memory, and network bandwidth resources for VNFs. Guo et al. [34] proposed a shadow routing-based approach by packing virtual machines onto physical machines in the cloud.

In this study, we consider the scenario of vehicles leaving the microcloud to design a dynamic scaling method. Furthermore, our proposed algorithm determines both the

value of the offloading ratio to the vehicular microcloud and the maximum number of remaining vehicle instances.

### C. SERVER BREAKAGE AND REPAIR

Gao et al. [35] analyzed an M/G/1 model with two types of server breakdown assumptions: if the server is idle, it needs some time to be repaired after the breakdown; otherwise, it can be repaired immediately without any delay. Chakka et al. [36] used the joint-state approach to model the performance of a multi-node system with breakdowns and repairs. Schwefe et al. [37] considered a cluster system having multiple nodes with a high-variance repair duration, where each server provides a degraded service when a fault occurs, and they represented it as an M/MMPP/1 model. Li et al. [38] proposed a scheduling algorithm for cloud workflow execution to guarantee the deadline constraint and resource utilization with several kinds of resource failures. Alam et al. [39] first presented a multi-server queueing system with a mode in which one server is broken, and they then extended it to include more than one server breakdown.

In this study, we did not focus on how to repair broken servers. Instead, our proposed queueing model permits any number of instances to leave the system at any time, without assumptions or heavy computation.

## VII. CONCLUSION

This study proposes an offloading system architecture comprising an RSU, 5G core network, and vehicular microcloud. We design dynamic scaling rules for vehicle instances and propose, without making assumptions, a queueing model in which any number of instances can leave at any time. To evaluate the system performance, we conduct extensive simulations to cross-validate our closed-form solutions. In addition, we propose TCOA to optimize the offloading ratios and remaining vehicle instances. The simulation results demonstrate that TCOA exhibits the best system performance of the five schemes. In the future, we intend to further investigate this scenario by considering more factors, such as the mobility of vehicles, the effects of 5G core network slices, and the other issues mentioned in the discussion section.

## APPENDIX A
## DERIVATION OF THE M/M/1/K RESPONSE TIME IN THE QUEUE

The response time in the queue for the M/M/1/K model (RSU) is derived in this section. To achieve this goal, the state probabilities of this system when there are different numbers of tasks in the queue must be derived first.

### A. STATE PROBABILITY ANALYSIS

We define $p_n$ as the state probability with $n$ tasks in the queue, and the range of $n$ is from zero to $K_r$ because of the task buffer limitation.

According to the balance equations of the state transitions:

$$p_0 \times \lambda_r = p_1 \times \mu_r$$

$$p_1 \times (\lambda_r + \mu_r) = p_0 \times \lambda_r + p_2 \times \mu_r$$

Leading to

$$p_1 = p_0 \times \frac{\lambda_r}{\mu_r}$$

$$p_2 = p_0 \times \frac{\lambda_r^2}{\mu_r^2}$$

By mathematical induction, we derive

$$p_n = p_0 \times \frac{\lambda_r^n}{\mu_r^n} \tag{13}$$

Thus far, all state probabilities $p_n$ are defined. Next, we obtain the value of each state probability $p_n$ by means of Equation (14), which is shown below.

$$\sum_{n=0}^{K_r} p_n = 1 \tag{14}$$

### B. RESPONSE TIME IN THE QUEUE

After obtaining the state probabilities, we derive the response time in the queue by following Little's law:

$$W_{qr} = \frac{L_q}{\lambda_{eff}} \tag{15}$$

In Equation (15), $\lambda_{eff}$ denotes the effective arrival rate when there is no blocking due to a full task buffer, and the expected value of the task number in the queue is denoted as $L_q$.

$$\lambda_{eff} = \lambda_r \times (1 - pK) \tag{16}$$

In Equation (16), $pK$ denotes the blocking probability, which is equal to the probability of a full task buffer.

$$pK = \frac{p_{K_r}}{\sum_{n=0}^{K_r} p_n}$$

Leading to

$$pK = \begin{cases} \dfrac{1}{K_r + 1}, & \text{if } \phi = 1 \\ \dfrac{(1-\phi) \times \phi^{K_r}}{1 - \phi^{K_r+1}}, & \text{otherwise} \end{cases} \tag{17}$$

In Equation (17), $\phi$ denotes the traffic intensity.

$$\phi = \frac{\lambda_r}{\mu_r} \tag{18}$$

Thus far, we have obtained the value of $\lambda_{eff}$. In addition, we need the value of $L_q$, which is equal to the number of tasks in the system except for the serving task.

$$L_q = L - (1 - p_0)$$

$$L = \sum_{n=0}^{K_r} n \times p_n$$

Leading to

$$L_q = \begin{cases} \dfrac{K_r \times (K_r - 1)}{2 \times (K_r + 1)}, & \text{if } \phi = 1 \\ \dfrac{\phi}{1-\phi} - \dfrac{\phi \times (K_r \times \phi^{K_r} + 1)}{1 - \phi^{K_r+1}}, & \text{otherwise} \end{cases} \tag{19}$$

Using Equations (15) to (19), we derive the response time in queue ($W_{qr}$) for the M/M/1/K model (RSU).

## APPENDIX B
## DERIVATION OF STEADY-STATE PROBABILITY

The state probabilities of the vehicular microcloud are derived in this section. We define $\pi_{i,j}$ as the state probability with $i$ vehicle instances and $j$ tasks in the queue, where $I$ is the maximum of $i$ and $J$ is the maximum of $j$. The state probabilities can be derived in order based on the range of $i$, as follows:

Assume $\pi_{I,J}$ is already known.

### C. FOR I = I

$$\pi_{I,j} \times (I\gamma + min(i,j) \times \mu_v + \lambda_v) = \pi_{I,j-1} \times \lambda_v + \pi_{I,j+1} \times min(i, j+1) \times \mu_v \quad (20)$$

$$\pi_{I,I-N} \times (I\gamma + min(i,j) \times \mu_v + \lambda_v) = \pi_{I,I-N+1} \times min(i, j+1) \times \mu_v \quad (21)$$

Assume $\pi_{I,j} = \pi_{I,j+1} \times a_j^{(I)}$,
leading to

$$a_{I-N}^{(I)} = \frac{min(I, I-N+1) \times \mu_v}{I\gamma + min(I, I-N) \times \mu_v + \lambda_v} \quad (22)$$

$$a_j^{(I)} = \frac{min(I, j+1) \times \mu_v}{I\gamma + min(I,j) \times \mu_v + \lambda_v - a_{j-1}^{(I)} \times \lambda_v} \quad (23)$$

To derive the state probabilities with $i$ in the next range, $\pi_{I-1,J}$ must be obtained.

$$\pi_{I-1,J} \times \alpha = (\sum_{j=I-N}^{J} \pi_{I,j} \times I\gamma) + \pi_{I,I-N} \times min(I, I-N) \times \mu_v$$

$$\pi_{I-1,J} = \frac{(\sum_{j=I-N}^{J} \pi_{I,j} \times I\gamma) + \pi_{I,I-N} \times min(I, I-N) \times \mu_v}{(min(I,J) - (I-1)) \times \alpha} \quad (24)$$

Thus far, the state probabilities $\pi_{i,j}$ are defined within range $i = I$, and the state probability $\pi_{I-1,J}$ is also known.

### D. FOR I > I > N

$$\pi_{i,j} \times (I\gamma + min(i,j) \times \mu_v + \lambda) = \pi_{i,j-1} \times \lambda_v + \pi_{i,j+1} \times min(i, j+1) \times \mu_v + \pi_{i+1,j} \times (i+1) \times \gamma \quad (25)$$

$$\pi_{i,i-N} \times (I\gamma + min(i, i-N) \times \mu_v + \lambda_v) = \pi_{i,i-N+1} \times \mu_v + \pi_{i+1,i-N+1} \times min(i+1, i-N+1) \times \mu_v \quad (26)$$

Assume $\pi_{i,j} = \pi_{i,j+1} \times a_j^{(i)} + b_j^{(i)}$,
leading to

$$a_{i-N}^{(i)} = \frac{min(i, i-N+1) \times \mu_v}{i\gamma + min(i, i-N) \times \mu_v + \lambda_v} \quad (27)$$

$$b_{i-N}^{(i)} = \frac{\pi_{i+1,i-N+1} \times min(i+1, i-N+1) \times \mu_v}{i\gamma + min(i, i-N) \times \mu_v + \lambda_v} \quad (28)$$

$$a_j^{(i)} = \frac{min(i, j+1) \times \mu_v}{i\gamma + min(i,j) \times \mu_v + \lambda_v - a_{j-1}^{(i)} \times \lambda_v} \quad (29)$$

$$b_j^{(i)} = \frac{b_{j-1}^{(i)} \times \lambda_v + \pi_{i+1,j} \times (i+1)\gamma}{i\gamma + min(i,j) \times \mu_v + \lambda_v - a_{j-1}^{(i)} \times \lambda_v} \quad (30)$$

To derive the state probabilities with $i$ in the next range, $\pi_{i-1,J}$ must be obtained.

$$\pi_{i-1,J} \times min(J-i+1, I-i+1) \times \alpha = (\sum_{j=i-N}^{J} \pi_{i,j} \times i\gamma) + \pi_{i,i-N} \times min(i, i-N) \times \mu_v$$

$$\pi_{i-1,J} = \frac{(\sum_{j=i-N}^{J} \pi_{i,j} \times i\gamma) + \pi_{i,i-N} \times min(i, i-N) \times \mu_v}{min(J-i+1, I-i+1) \times \alpha} \quad (31)$$

Thus far, the state probabilities $\pi_{i,j}$ are defined within range $i > N$, and the state probability $\pi_{N,J}$ is also known.

### E. FOR I = N

$$\pi_{N,j} \times (i\gamma + min(N,j) \times \mu_v + \lambda_v) = \pi_{N,j-1} \times \lambda_v + \pi_{N,i+1} \times min(N, j+1) \times \mu_v + \pi_{N+1,j} \times (N+1) \times \gamma \quad (32)$$

$$\pi_{N,0} \times (N\gamma + \lambda_v) = (\pi_{N,1} + \pi_{N+1,1}) \times \mu_v \quad (33)$$

Assume $\pi_{N,j} = \pi_{N.j+1} \times a_j^{(N)} + b_j^{(N)}$, leading to

$$a_0^{(N)} = \frac{\mu_v}{N\gamma + \lambda_v} \quad (34)$$

$$b_0^{(N)} = \frac{\pi_{N+1,1} \times \mu_v}{N\gamma + \lambda_v} \quad (35)$$

$$a_j^{(N)} = \frac{min(N, j+1) \times \mu_v}{N\gamma + min(N,j) \times \mu_v + \lambda_v - a_{j-1}^{(N)} \times \lambda_v} \quad (36)$$

$$b_j^{(N)} = \frac{b_{j-1}^{(N)} \times \lambda_v + \pi_{N+1,j} \times (N+1)\gamma}{N\gamma + min(N,j) \times \mu_v + \lambda_v - a_{j-1}^{(N)} \times \lambda_v} \quad (37)$$

To derive the state probabilities with $i$ in the next range, $\pi_{N-1,J}$ ($N \geq 1$) must be obtained.

$$\pi_{N-1,J} \times min(J-N+1, I-N+1) \times \alpha = \sum_{j=0}^{J} \pi_{N,j} \times N\gamma$$

$$\pi_{N-1,J} = \frac{\sum_{j=0}^{J} \pi_{N,j \times N\gamma}}{min(J-N+1, I-N+1) \times \alpha} \quad (38)$$

Thus far, the state probabilities $\pi_{i,j}$ are defined within range $i \geq N$, and the state probability $\pi_{N-1,J}$ ($N \geq 1$) is also known.

### F. FOR N > I > 0

$$\pi_{i,j} \times (i\gamma + min(i,j) \times \mu_v + \lambda_v) = \pi_{i,j-1} \times \lambda_v + \pi_{i,j+1} \times min(i,j+1) \times \mu_v + \pi_{i+1,j} \times (i+1)\gamma \quad (39)$$

$$\pi_{i,0} \times (i\gamma + \lambda_v) = \pi_{i,1} \times \mu_v + \pi_{i+1,0} \times (i+1)\gamma \quad (40)$$

Assume $\pi_{i,j} = \pi_{i,j+1} \times a_j^{(i)} + b_j^{(i)}$, leading to

$$a_0^{(i)} = \frac{\mu_v}{i\gamma + \lambda_v} \quad (41)$$

$$b_0^{(i)} = \frac{\pi_{i+1,0} \times (i+1)\gamma}{i\gamma + \lambda_v} \quad (42)$$

$$a_j^{(i)} = \frac{min(i,j+1) \times \mu_v}{i\gamma + min(i,j) \times \mu_v + \lambda_v - a_{j-1}^{(i)} \times \lambda_v} \quad (43)$$

$$b_j^{(i)} = \frac{b_{j-1}^{(i)} \times \lambda_v + \pi_{i+1,j} \times (i+1)\gamma}{i\gamma + min(i,j) \times \mu_v + \lambda_v - a_{j-1}^{(i)} \times \lambda_v} \quad (44)$$

To derive the state probabilities with $i$ in the next range, $\pi_{i-1,J}$ must be obtained.

$$\pi_{i-1,J} \times min(J-i+1, I-i+1) \times \alpha = \sum_{j=0}^{J} \pi_{i,j} \times i\gamma$$

$$\pi_{i-1,J} = \frac{\sum_{j=0}^{J} \pi_{i,j} \times i\gamma}{min(J-i+1, I-i+1) \times \alpha} \quad (45)$$

Thus far, the state probabilities $\pi_{i,j}$ are defined within range $i > 0$, and the state probability $\pi_{0,J}$ is also known.

### G. FOR I = 0

$$\pi_{0,j} \times \lambda_v = \pi_{0,j-1} \times \lambda_v + \pi_{1,j} \times \gamma \quad (46)$$

$$\pi_{0,0} \times \lambda_v = \pi_{1,0} \times \gamma \quad (47)$$

Assume $\pi_{0,j} = \pi_{0,j-1} \times a_j^{(0)} + b_j^{(0)}$, leading to

$$a_0^{(0)} = 0$$

$$b_0^{(0)} = \frac{\pi_{1,0} \times \gamma}{\lambda_v} \quad (48)$$

$$a_j^{(0)} = 0$$

$$b_j^{(0)} = \frac{b_{j-1}^{(0)} \times \lambda_v + \pi_{1,j} \times \gamma}{\lambda_v} \quad (49)$$

Thus far, all the state probabilities $\pi_{i,j}$ are defined. If $N = 0$, because all values of $i$ are already included in the range $I \geq i \geq N$, the remaining part in the range $N > i \geq 0$ is unreasonable and should be ignored. In contrast, if $N = I$, all values of $i$ are already included in the range $N \geq i \geq 0$; thus, the other part in the range $I \geq i > N$ is unreasonable and should be ignored. We can then obtain the values of each state probability $\pi_{i,j}$ by means of Equation (50), which is shown below.

$$\sum_{(i,j) \in S_{space}} \pi_{i,j} = 1 \quad (50)$$

The notation $S_{space}$ denotes the set of all vehicular microcloud states.

## REFERENCES

[1] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[2] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep reinforcement learning-based computation offloading in vehicular edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
[3] *Study on LTE Support for Vehicle to Everything (V2X) Services*, Standard 3GPP TR 22.885, 2017.
[4] Z. Y. Rawashdeh and S. M. Mahmud, "Admission control for roadside units based on virtual air-time transmissions," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2011, pp. 1–6.
[5] T. Higuchi, R. V. Rabsatt, M. Gerla, O. Altintas, and F. Dressler, "Cooperative downloading in vehicular heterogeneous networks at the edge," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–5.
[6] B.-J. Qiu, C.-Y. Hsieh, J.-C. Chen, and F. Dressler, "DCOA: Double-check offloading algorithm to road-side unit and vehicular micro-cloud in 5G networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
[7] R. Zhang, F. Yan, W. Xia, S. Xing, Y. Wu, and L. Shen, "An optimal roadside unit placement method for VANET localization," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.
[8] T. Liu, S. Zhou, and Z. Niu, "Joint optimization of cache allocation and content placement in urban vehicular networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[9] P. Gu, C. Hua, R. Khatoun, Y. Wu, and A. Serhrouchni, "Cooperative anti-jamming relaying for control channel jamming in vehicular networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.
[10] N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment: Waiting or downloading now?" in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[11] *System architecture for the 5G System (5GS)*, Standard 3GPP TS 23.501, 2020.
[12] V. Millnert, J. Eker, and E. Bini, "Achieving predictable and low end-to-end latency for a network of smart services," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.
[13] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Vehicular micro cloud in action: On gateway selection and gateway handovers," *Ad Hoc Netw.*, vol. 78, pp. 73–83, Sep. 2018.
[14] F. Malandrino, C. Casetti, C. F. Chiasserini, C. Sommer, and F. Dressler, "Content downloading in vehicular networks: Bringing parked cars into the picture," in *Proc. IEEE 23rd Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2012, pp. 1534–1539.
[15] F. Malandrino, C. Casetti, C.-F. Chiasserini, C. Sommer, and F. Dressler, "The role of parked cars in content downloading for vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 9, pp. 4606–4617, Nov. 2014.
[16] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of Vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.
[17] R. D. Yates and S. Kaul, "Real-time status updating: Multiple sources," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 2666–2670.
[18] M. Moltafet, M. Leinonen, and M. Codreanu, "On the age of information in multi-source queueing models," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 5003–5017, Aug. 2020.
[19] M. Z. Hasan, F. Al-Turjman, and H. Al-Rizzo, "Analysis of cross-layer design of quality-of-service forward geographic wireless sensor network routing strategies in green Internet of Things," *IEEE Access*, vol. 6, pp. 20371–20389, 2018.
[20] *The Network Simulator Ns-2*. Accessed: Feb. 20, 2020. [Online]. Available: http://www.isi.edu/nsnam/ns/
[21] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, Aug. 2021.
[22] H. Zhou, T. Wu, X. Chen, S. He, D. Guo, and J. Wu, "Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing," *IEEE Trans. Mobile Comput.*, early access, Jul. 18, 2022, doi: 10.1109/TMC.2022.3189050.
[23] E. D. Denman and A. N. Beavers, "The matrix sign function and computations in systems," *Appl. Math. Comput.*, vol. 2, no. 1, pp. 63–94, Jan. 1976.
[24] H. Guo, J. Liu, H. Qin, and H. Zhang, "Collaborative computation offloading for mobile-edge computing over fiber-wireless networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[25] B. Li, "Optimal offloading for dynamic compute-intensive applications in wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
[26] W. Xu, H. Wu, J. Chen, W. Shi, H. Zhou, N. Cheng, and X. S. Shen, "ViFi: Vehicle-to-vehicle assisted traffic offloading via roadside WiFi networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[27] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11181–11191, Nov. 2018.
[28] H. Zhou, Z. Zhang, D. Li, and Z. Su, "Joint optimization of computing offloading and service caching in edge computing-based smart grid," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1122–1132, Apr./Jun. 2023.
[29] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, "Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An A3C-based approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1326–1338, May 2023.
[30] Y. Song, J. Peng, K. Liu, F. Jiang, W. Liu, and Z. Huang, "A hybrid particle swarm ant colony based resource reservation for geo-distributed cloud service," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[31] Y. Ma, W. Liang, M. Huang, and S. Guo, "Profit maximization of NFV-enabled request admissions in SDNs," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.
[32] T. Phung-Duc, Y. Ren, J.-C. Chen, and Z.-W. Yu, "Design and analysis of deadline and budget constrained autoscaling (DBCA) algorithm for 5G mobile networks," 2016, *arXiv:1609.09368*.
[33] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee, "Auto-scaling network service chains using machine learning and negotiation game," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1322–1336, Sep. 2020.
[34] Y. Guo, A. L. Stolyar, and A. Walid, "Online VM auto-scaling algorithms for application hosting in a cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 889–898, Jul. 2020.
[35] S. Gao, J. Zhang, and X. Wang, "Analysis of a retrial queue with two-type breakdowns and delayed repairs," *IEEE Access*, vol. 8, pp. 172428–172442, 2020.
[36] R. Chakka, E. Ever, and O. Gemikonakli, "Joint-state modeling for open queuing networks with breakdowns, repairs and finite buffers," in *Proc. 15th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Oct. 2007, pp. 260–266.
[37] H.-P. Schwefel and I. Antonios, "Performability models for multi-server systems with high-variance repair durations," in *Proc. 37th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2007, pp. 770–779.
[38] Z. Li, V. Chang, H. Hu, H. Hu, C. Li, and J. Ge, "Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds," *Inf. Sci.*, vol. 568, pp. 13–39, Aug. 2021.
[39] M. Alam and V. Mani, "Recursive solution technique in a multi-server bi-level queueing system with server breakdowns," *IEEE Trans. Rel.*, vol. 38, no. 4, pp. 416–421, Oct. 1989.

**BO-JUN QIU** (Student Member, IEEE) received the B.S. degree in computer science from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2017, where he is currently pursuing the Ph.D. degree with the Department of Computer Science. His research interests include 5G core networks, vehicular microcloud, and queueing analysis.

**CHENG-YING HSIEH** (Student Member, IEEE) received the B.S. degree in mechanical engineering and the Ph.D. degree in computer science from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2014 and 2022, respectively. His research interests include 5G mobility networks, queueing analysis, and deep learning.

**JYH-CHENG CHEN** (Fellow, IEEE) received the Ph.D. degree from the State University of New York at Buffalo, in 1998. He was a Research Scientist with Bellcore/Telcordia Technologies, Morristown, NJ, USA, from 1998 to 2001, and a Senior Scientist with Telcordia Technologies, Piscataway, NJ, USA, from 2008 to 2010. He was with the Department of Computer Science, National Tsing Hua University (NTHU), Hsinchu, Taiwan, as an Assistant Professor, an Associate Professor, and a Full Professor, from 2001 to 2008. He has been a Faculty Member with National Yang Ming Chiao Tung University (NYCU), formerly National Chiao Tung University (NCTU), since 2010, where he is currently the Chair Professor with the Department of Computer Science and the Dean of the College of Computer Science. He is also a Distinguished Member of the Association for Computing Machinery (ACM). He was a member of the Fellow Evaluation Committee, IEEE Computer Society. He has received numerous awards, including the Outstanding Teaching Award from NCTU and NTHU, the Outstanding I. T. Elite Award, Taiwan, the Mentor of Merit Award from NCTU, the Medal of Honor and the K. T. Li Breakthrough Award from the Institute of Information and Computing Machinery, the Outstanding Engineering Professor Award from the Chinese Institute of Engineers, the Outstanding Research Award from the Ministry of Science and Technology, the Best Paper Award for Young Scholars from the IEEE Communications Society Taipei and Tainan Chapters, the IEEE Information Theory Society Taipei Chapter, and the Telcordia CEO Award.

**FALKO DRESSLER** (Fellow, IEEE) received the M.Sc. and Ph.D. degrees from the Department of Computer Science, University of Erlangen, in 1998 and 2003, respectively. He is currently a Full Professor and the Chair of Telecommunication Networks with the School of Electrical Engineering and Computer Science, TU Berlin. He authored the textbooks *Self-Organization in Sensor and Actor Networks* (Wiley & Sons) and *Vehicular Networking* (Cambridge University Press). His research interests include adaptive wireless networking (sub-6GHz, mmWave, visible light, and molecular communication) and wireless-based sensing with applications in ad-hoc and sensor networks, the Internet of Things, and cyber-physical systems. He is a member of the German National Academy of Science and Engineering (acatech). He has been chairing conferences, such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, and IEEE GLOBECOM. He has been an IEEE Distinguished Lecturer and an ACM Distinguished Speaker. He is an ACM Distinguished Member. He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. He has been the Associate Editor-in-Chief of IEEE TRANSACTIONS ON MOBILE COMPUTING and *Computer Communications* (Elsevier) and an Editor of journals, such as IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, *Ad Hoc Networks* (Elsevier), and *Nano Communication Networks* (Elsevier).