DCOA: Double-Check Offloading Algorithm to Road-Side Unit and Vehicular Micro-Cloud in 5G Networks

Bo-Jun Qiu*, Cheng-Ying Hsieh*, Jyh-Cheng Chen* and Falko Dressler†
*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
†School of Electrical Engineering and Computer Science, TU Berlin, Berlin, Germany
qiubj759@cs.nctu.edu.tw, ingyaya36.me03g@nctu.edu.tw,
jcc@cs.nctu.edu.tw, dressler@ccs-labs.org

***Abstract*—Next generation intelligent transportation systems aim at many cooperative perception and cooperative driving functions that need significant computational resources. Offloading such tasks to some mobile edge computing solutions is considered part of the solution, which is currently investigated in the scope of 5G networks. In the automotive context, such edge systems could be road-side units (RSU), which, however, can easily be overloaded at peak times. Vehicular micro-cloud approaches have been proposed to overcome such problems by sharing computational resources of nearby cars. In this study, we propose an offloading system architecture to enable such offloading such vehicular micro-cloud 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 these insights, we propose the Double-Check Offloading Algorithm (DCOA) to obtain the best offloading ratio to the vehicular micro-cloud. Our simulation results show the proposed DCOA has better system performances compared with four other offloading schemes.**

***Index Terms*—Edge computing, offloading, 5G core network, vehicular micro-cloud**

I. Introduction

In recent years, there has been an increasing number of vehicle-related applications, such as collision avoidance, with the need for high calculating resources and low latency. Because of the lake of computing resources on a general vehicle, it is not sufficient to perform the calculations of such applications within a short time. Thus, computing resources are one of the main limitations of vehicle-related applications. A popular solution is to offload the calculating tasks to a road-side unit (RSU) [1] deployed at the edge side of the cellular network. The RSU can complete tasks independently or directly deliver them to a local edge server[2].

However, the performance of the RSU may decrease when there are many vehicles simultaneously using the RSU[3]. A potential solution is to offload the calculating tasks to a remote server through infrastructure networks, such as the 5th generation (5G) core network with the next generation node B (gNB). Vehicular micro-cloud[4] is a promising candidate for remote servers because parked vehicles in a vehicular micro-cloud typically do not use their resources.

In comparison with RSU, offloading tasks to a remote vehicular micro-cloud may result in serving a larger number of users, thus increasing the system scalability. Unfortunately, benefits are always accompanied by overheads, i.e., the task response time increases because of the long distance between the users and the vehicular micro-cloud. Additionally, the cost increases owing to the use of the infrastructure network and the fee for parked vehicles. From the viewpoint of system operators, balancing these benefits and overheads is essential.

In this study, we propose an offloading system architecture and describe it as a queueing model to derive closed-forms of metrics. Then, we use network simulator 2 (ns-2) to verify the correctness of the closed-forms interactively. Additionally, we propose our double-check offloading algorithm (DCOA) to obtain the offloading ratio of the remote vehicular micro-cloud by observing system performance. The proposed DCOA is based on an objective function, in which operators and service providers can decide the weights of metrics to define their priorities.

II. Background

A. Road-Side Unit (RSU)

RSU is a unit placed beside the road and plays an important role in many vehicle-related applications. It communicates with nearby vehicles and provides diverse services, such as calculating tasks[1], locating vehicles[5], transmitting safety-related messages[6] or information[7], and downloading data[8]. For supporting these applications, an RSU should have the ability to compute complex calculations or directly connect to an edge server instead.

B. 5th Generation (5G) Core Network

The 5G core network comprises a data plane and a control plane. The data plane is independent of the control plane[9] and can be deployed close to users to improve communication qualities, such as latency. The control plane is a service-based architecture, which organizes different service blocks into network slices according to the service requirements of users. The service requirements can be mapped to a corresponding *Slice/Service Type* (SST) defined in 3GPP TS23.501, clause 5.15.2.2[10]. The *Ultra-Reliable Low-Latency Communication* (URLLC) slice will enable the use of the Internet with low latency and high reliability[11].

C. Vehicular Micro-Cloud

A vehicular micro-cloud comprises vehicles that are usually parked and rarely use their CPU resources, and it provides services as a server[12], [13], [14]. Additionally, it improves the resource usage rate of the parked vehicles, which may obtain benefits (for example, fees) as feedback. The scale of a vehicular micro-cloud is dynamic because parked vehicles may leave. Therefore, the calculating ability of a vehicular micro-cloud is also dynamic, and depends on the number of vehicles being used .

III. Related Work

A. Edge Traffic Offloading

Guo et al.[15] proposed an algorithm based on deadlines to decide whether to offload and then deliver tasks to an edge server or the cloud according to the cost involved. Li [16] focused on the offloading decision and the wireless scheduling to an edge server among many mobile devices. Zhan et al.[1] used deep learning to determine when and how to schedule offloading tasks to the RSUs along a road. Xu et al.[17] used vehicle-to-vehicle communications to offloading data traffic to other vehicles, which use WiFi to deliver these data traffic to the Internet. Dai et al.[18] focused on the relaying scheme to determine when and which vehicle cloudlets to be offload.

In this study, we focus on determining the offloading ratio to the remote vehicular micro-cloud by using the queueing model and the proposed algorithm. Additionally, rather than focusing on the method of offloading traffic to RSUs, we enable tasks to be directly offloaded to a vehicular micro-cloud through the gNBs and the 5G core network.

B. Dynamic Scaling System

Several studies[19], [20], [21] have focused on the tradeoff between cost and performance when turning on or off service instances. These studies observe the relationships between the metrics, propose algorithms or schemes to achieve a balance between these factors.

In this study, we do not focus on the algorithm to dynamically turn on or off service instances. Rather, our proposed algorithm determines the value of the offloading ratio to the vehicular micro-cloud.

IV. Proposed Offloading System

The proposed offloading system architecture is illustrated in Fig 1, and the notations are listed in Table I. The goal of the proposed offloading system is to reduce the RSU loadings by offloading the tasks of user equipment (UE) directly to the vehicular micro-cloud through the gNBs and the 5G core network. The UE may be vehicles or smartphones, the RSU represents the local system, and the vehicular micro-cloud represents the remote system. We assume that the task arrival rate of the system follows a Poisson distribution with mean λ.

When the local system is overloaded, offloading tasks to the remote system can not only increase the system service rate, but also increase the cost. As mentioned in section I, the former can be attributed to the larger number of tasks being handled, whereas the latter can be attributed to the additional fee for parked vehicles. Because of the URLLC slice, we assume the response time does not increase with the longer delivery distance in the infrastructure networks.

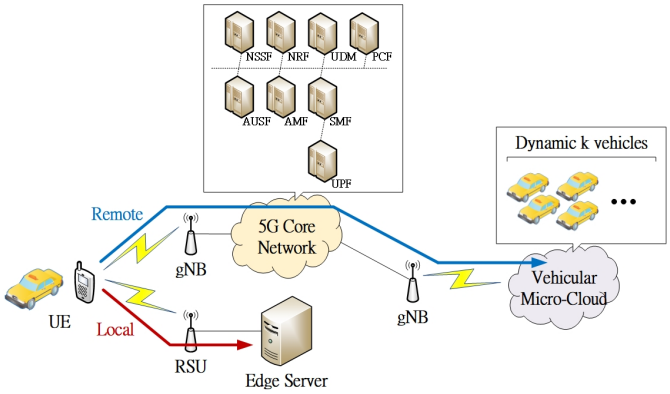


Figure 1. Proposed offloading system architecture

Table I
List of Notations

Notation	Explanation
P	System performance
λ	System task arrival rate
β	Offloading task ratio to remote system
C	Average system cost
W_q	Average system response time in queues per task
S	Average system service rate
λ_l	Task arrival rate in local system
μ_l	Service rate for each instance in local system
K_l	The number of maximum tasks can be accommodated in local system
C_l	Average local system cost
c_l	Average cost for each instance in local system
W_{ql}	Average local system response time in the queue per task
S_l	Average local system service rate
λ_r	Task arrival rate in remote system
μ_r	Service rate for each instance in remote system
K_r	The number of maximum tasks can be accommodated in remote system
C_r	Average remote system cost
c_r	Average cost for each vehicle instance in remote system
W_{qr}	Average remote system response time in the queue per task
S_r	Average remote system service rate
k	The number of vehicle instances in remote system
α	Setup rate for each vehicle instance in remote system
ω_1	Weight factor for C
ω_2	Weight factor for W_q
ω_3	Weight factor for S

A. Local System (RSU)

We assume that there exists only one RSU (only one edge server, if any), and its calculating time per task follows an exponential distribution with mean $1/\mu_l$. 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 buffers some of the tasks in a queue and ignores the remaining tasks because of overloading.

B. Remote System (Vehicular Micro-Cloud)

We assume that the vehicular micro-cloud consists of k vehicles parked in a parking lot. The calculating time per

task follows an exponential distribution with mean $1/\mu_r$ for each vehicle instance. If the number of tasks is greater than the number of started vehicle instances, a non-started vehicle instance is set up to reduce the response time. We assume that the setup time follows an exponential distribution with mean $1/\alpha$, and the vehicle instance consumes resources but can not calculate tasks during the setup time. In contrast, if the number of tasks is lesser than the number of started vehicle instances, a started vehicle instance is turned off immediately to save resources, and vehicle instances, if any, in the setup time are turned off first. If there are too many tasks, the vehicular micro-cloud buffers some of the tasks in the queue and ignores the remaining tasks.

C. Model Analysis

Three metrics are considered in evaluating the system performance: the average cost of the RSU and the vehicular micro-cloud by consuming resources, C, the average response time in the queues per task, W_q, and the average number of serving tasks, S. The system performance P is defined in (1).

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

Based on the definition of P, hereafter, we refer to the system performance as ***Cost Response time Production Service rate Division (CRPSD)***; a smaller CRPSD indicates a better system performance. Before calculating CRPSD, we normalize the three system metrics from 0 to 1. The weight factors in (1) can be set by operators or service providers to reflect their preferences, and we set them to 1 as the default value.

System metrics are the combination of the local metrics and remote metrics defined in (2), (3), and (4).

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

$$W_q = \frac{S_l \times W_{ql} + S_r \times W_{qr}}{S} \tag{3}$$

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

The system follows the remote offloading ratio β to offload tasks to the remote system and offloads the remaining tasks to the local system, as defined in (5) and (6).

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

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

By using the equations from (1) to (6), we express 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, we assume that its calculating ability is 100 times stronger than that of a vehicle instance. The resource consumption of an RSU may also be 100 times greater than that of a vehicle instance based on the previous assumption that the calculating speed of an RSU is 100 times faster than that of a vehicle instance. However, the vehicular micro-cloud involves an additional fee, and we assume that the additional fee increases the cost of the vehicle instance by two times. Therefore, the cost ratio between an RSU and a vehicle instance is 50. We set both the average service rate μ_r and the average cost c_r for each vehicle instance to be 1 as default.

The local system can be described as an M/M/1/N queueing model. Thus, we have the following forms: (7), (8), and (9).

$$C_l = c_l \times 1 = 50 \tag{7}$$

$$S_l = \min(\lambda_l, \mu_l) = \min(\lambda_l, 100) \tag{8}$$

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

$$\lambda_{eff} = \lambda_l \times (1 - pK)$$

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

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

$$\phi = \frac{\lambda_l}{\mu_l}$$

The remote system can be described as a queueing model, as proposed in previous study [21]. Because of the page limitation, we have only defined the key equations in this paper. For more details on the mathematical derivation or proof of the equations, please refer to the cited paper. The remote metrics can be defined as follows:

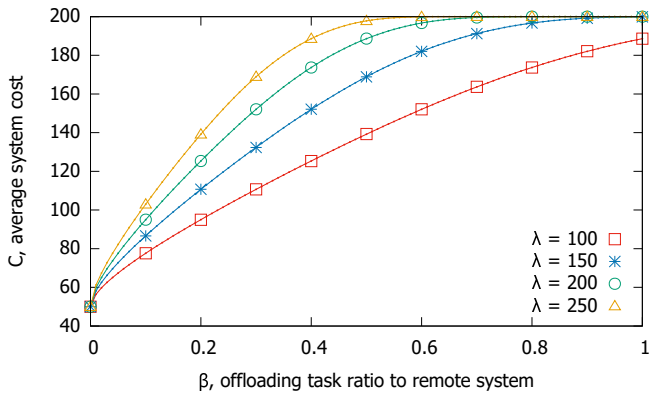
$$C_r = c_r\Big(\sum_{(i,j)\in S_{space}} \pi_{i,j} n_i + \sum_{i=0}^{k} \sum_{j=n_i}^{K_r} \pi_{i,j} \min(j - n_i, k - n_i)\Big) \tag{10}$$

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

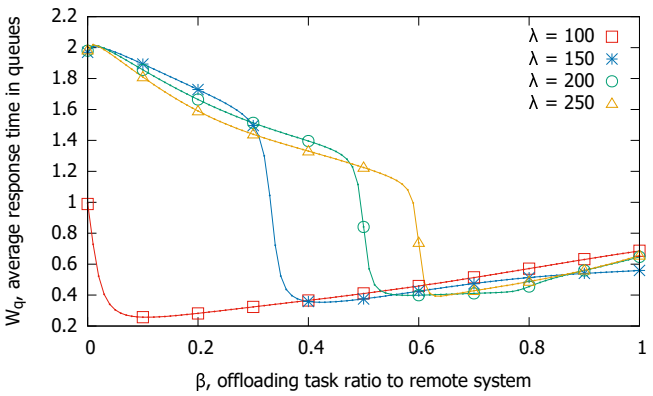
$$W_{qr} = \frac{\sum_{i=0}^{k} \sum_{j=n_i}^{K_r} \pi_{i,j} j}{\lambda_r (1 - \sum_{i=0}^{k} \pi_{i,K_r})} - \frac{1}{\mu_r} \tag{12}$$

In (10) to (12), $\pi_{i,j}$ denotes the probability of a vehicular micro-cloud state, which comprises i vehicle instances calculating tasks (already set up) and j tasks that need to be completed. The notation S_{space} denotes the set of all vehicular micro-cloud states, and n_i denotes the number of vehicles that perform calculations, which has the same value as i.

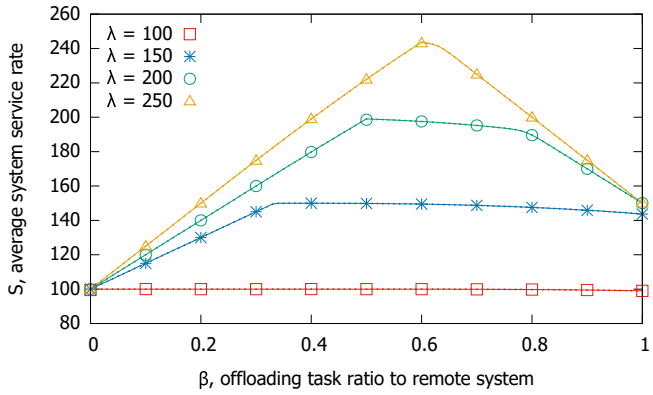
The closed-forms of the three local and remote metrics are obtained from (7) to (12). We used these closed-forms to calculate the system performance metrics defined in (1)–(4), and then used the network simulator 2 (ns-2)[22] version



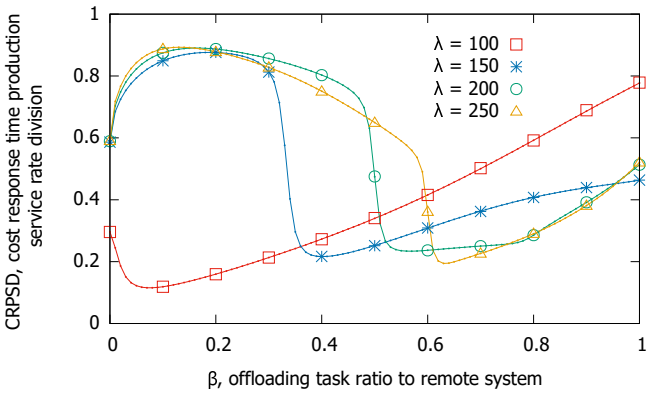
(a) Impacts of β on C



(b) Impacts of β on W_q



(c) Impacts of β on S



(d) Impacts of β on $CRPSD$

Figure 2. Parameter study of the core system metrics

2.35 to implement and run the simulation of the proposed offloading system. All the simulations in this paper run 1,000,000 simulation-time and have the following default values: K_l = 200, K_r = 250, k = 150, and α = 0.02. The results are depicted in Fig. 2a – Fig. 2d. By observing CRPSD in Fig. 2d, we find that the best remote offloading ratio locates at the local minimum searched from β = 1.0. Additionally, we observe that it may locate at the local minimum searched from β = 0.0 in some particular situations, in which the RSU can perform calculations fast without consuming a considerable number of resources.

V. Double-Check Offloading Algorithm (DCOA)

A. DCOA Overview

According to the observations of Fig. 2d, we propose the DCOA in algorithm 1, which checks the two local minimums searched from β = 0.0 and β = 1.0 to identify the correct global minimum. We use a gradient descent method, DCOA gradient descent, to determine the local minimum. If the feedback result worsens, the algorithm changes the searching direction and reduces the searching step simultaneously. Fig. 2d depicts that the distance during the search for the local minimum starting from β = 1.0 is much greater than that starting from β = 0.0. Therefore, we set a large searching step (0.16) when starting from β = 1.0 and a small searching step (0.02) when starting from β = 0.0.

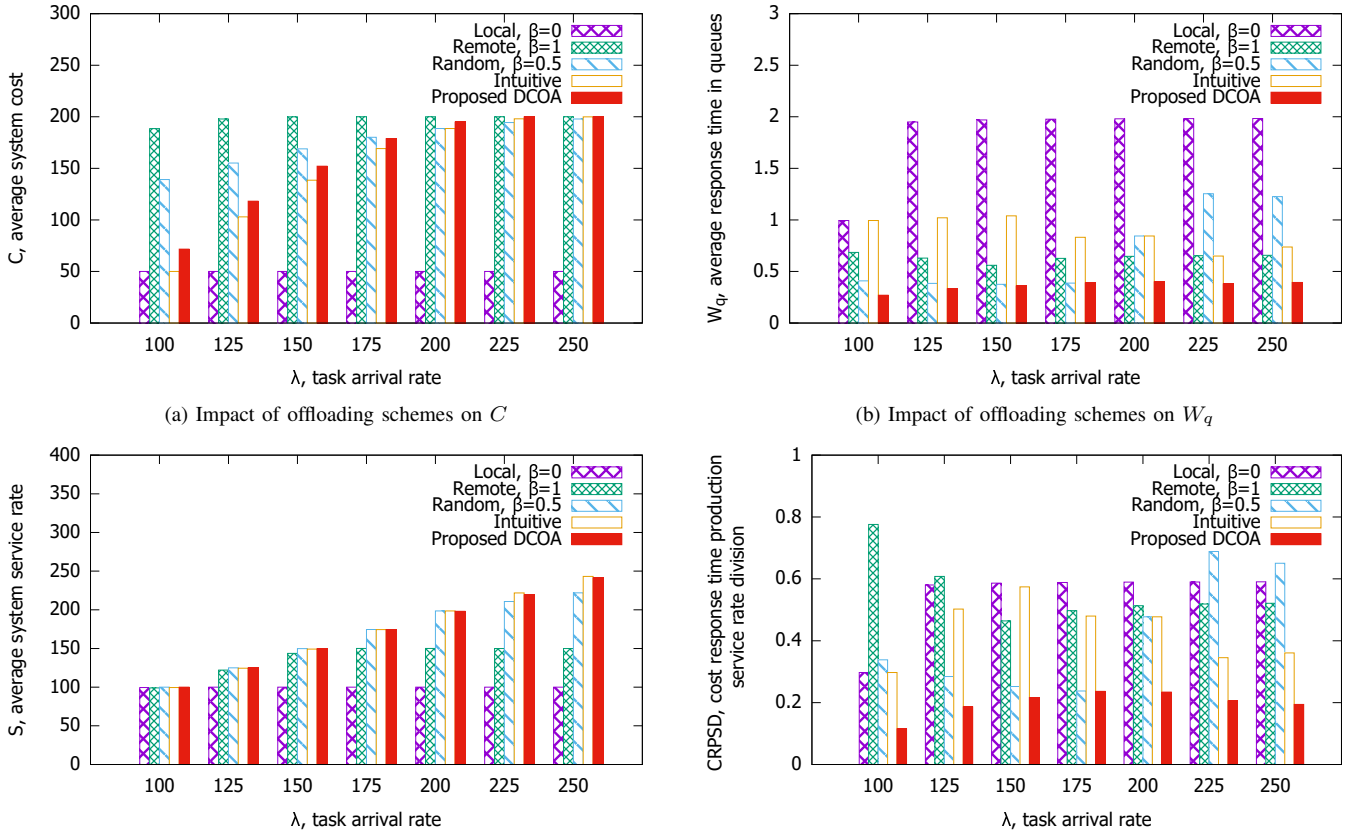
Algorithm 1: Double-Check Offloading Algorithm (DCOA)

Procedure $DCOA\,(P)$
- $\beta_r = DCOAGradientDescent(-1, 1.0, 0.16, P)$;
- $\beta_l = DCOAGradientDescent(1, 0.0, 0.02, P)$;
- **if** $P(\beta_r) < P(\beta_l)$ **then**
  - $\beta_{DCOA} = \beta_r$;
- **else**
  - $\beta_{DCOA} = \beta_l$;
- return β_{DCOA};

B. DCOA Gradient Descent

DCOA gradient descent shown in algorithm 2 is used to determine the local minimum for DCOA:

- If the searching feedback is worse than the previous one, change the searching direction and reduce the searching step.
- Record the searching result for comparison with that obtained the next time.
- If the step becomes sufficiently small (< 0.01), keep step 0.01 and return the next searching result to DCOA.



(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 3. Impact of offloading schemes

- If the step is greater than a thread (0.01), move a step.
- If the current β value is larger than 1.0, set β = 1.0 and change the searching direction and reduce the searching step.
- If the current β value is lesser than 0.0, set β = 0.0 and change the searching direction and reduce the searching step.

C. Evaluation and Comparison

We compare DCOA with four other schemes: *Local*, *Remote*, *Random*, and *Intuitive*. The *Local* scheme offloads all the tasks to the RSU, whereas the *Remote* scheme offloads all the tasks to the vehicular micro-cloud. The *Random* scheme offloads tasks to the RSU and the vehicular micro-cloud randomly (50%). The *Intuitive* scheme uses an intuitive method, whereby it allows the local arrival rate λ_l equal to the RSU service rate μ_l, and offloads the remaining tasks to the vehicular micro-cloud, to prevent the RSU from getting overloaded and to serve users to the greatest possible extent. The simulation results are depicted in Fig. 3a – Fig. 3d.

The proposed DCOA exhibits the best CRPSD at each arrival rate. DCOA performs well, particularly with respect to response time. The performance of the *Intuitive* scheme is extremely close to that of DCOA in terms of cost and service rate, but significantly worse than DCOA with respect to response time. This is because the RSU in the *Intuitive* scheme undergoes heavy loading. Although the task does not accumulate and overflow in the queue, the response time increases considerably.

The *Local* scheme exhibits the best cost because only the RSU consumes resources, whereas no started vehicle instances exist in the vehicular micro-cloud. In contrast, it performs the worst in terms of response time and service rate.

A non-institute trend of the *Remote* scheme exhibits in terms of CRPSD, which decreases with the increasing task arrival rate from 100 to 150. This trend is because the vehicular micro-cloud always consumes resources for vehicle instances in setup time. In contrast, the vehicle instances sometimes immediately turn off if any task is completed during this period. Therefore, the cost of the *Remote* scheme is extremely close to the maximum, k, while the service rate is increasing and then improves the CRPSD. Additionally, this trend affects other offloading schemes except for the *Local* scheme.

VI. Conclusion

In this study, we proposed an offloading system architecture comprising the RSU, 5G core network, and vehicular micro-cloud. Then, we interactively verified the correctness between the closed-forms and ns-2 simulations. Additionally, we proposed DCOA, including the DCOA gradient descent, and compared it with other offloading schemes. The simulation

Algorithm 2: DCOA Gradient Descent

Procedure $DCOAGradientDescent$ ($Sign$, β_{init}, $Step$, P)

$\beta_{current} = \beta_{init}$;
$P_{previous} = 2.0$;
$Flag_{running} =$ True;
$Flag_{terminal} =$ False;
while $Flag_{running}$ **do**
  if $P(\beta_{current}) > P_{previous}$ **then**
    $Sign = Sign \times (-1)$;
    $Step = Step/2$;
  $P_{previous} = P(\beta_{current})$;
  if $Step < 0.01$ **then**
    $\beta_{current} = \beta_{current} + Sign \times 0.01$;
    $Flag_{running} =$ False;
  else
    $\beta_{current} = \beta_{current} + Sign \times Step$;
  if $\beta_{current} > 1.0$ **then**
    $\beta_{current} = 1.0$;
    if $Flag_{terminal}$ **then**
      $Sign = Sign \times (-1)$;
      $Step = Step/2$;
    else
      $Flag_{terminal} =$ True;
  else if $\beta_{current} < 0.0$ **then**
    $\beta_{current} = 0.0$;
    if $Flag_{terminal}$ **then**
      $Sign = Sign \times (-1)$;
      $Step = Step/2$;
    else
      $Flag_{terminal} =$ True;
  else
    $Flag_{terminal} =$ False;
$\beta_{DCOAGradientDescent} = \beta_{current}$;
return $\beta_{DCOAGradientDescent}$;

results demonstrated that DCOA exhibited the best system performance. In the future, we intend to further investigate this issue by considering more factors, such as the mobility of vehicles, previously turning on and slowly turning off the vehicle instance in the vehicular micro-cloud, and attempting to simulate the effects of 5G core network slices.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Science and Technology of Taiwan under grant numbers 109-2218-E-009-004, 108-2221-E-009-042-MY3, and MOST 108-2218-E-009-028.

REFERENCES

[1] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep Reinforcement Learning-Based Computation Offloading in Vehicular Edge Computing," in *IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019.

[2] *3GPP TR 22.885 - Study on LTE support for Vehicle to Everything (V2X) services*.

[3] Z. Y. Rawashdeh and S. M. Mahmud, "Admission Control for Roadside Units Based on Virtual Air-Time Transmissions," in *IEEE Global Communications Conference (GLOBECOM)*, Houston, Texas, USA, 2011.

[4] T. Higuchi, R. V. Rabsatt, M. Gerla, O. Altintas, and F. Dressler, "Cooperative Downloading in Vehicular Heterogeneous Networks at the Edge," in *IEEE Global Communications Conference (GLOBECOM 2019), Workshop on New and Disruptive Technologies and Applications for Mobile Edge/Fog Computing (MobileEdgeCom 2019)*. Waikoloa, HI: IEEE, 12 2019.

[5] R. Zhang, F. Yan, W. Xia, S. Xing, Y. Wu, and L. Shen, "An Optimal Roadside Unit Placement Method for VANET Localization," in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017.

[6] P. Gu, C. Hua, R. Khatoun, Y. Wu, and A. Serhrouchni, "Cooperative Anti-Jamming Relaying for Control Channel Jamming in Vehicular Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017.

[7] T. Liu, S. Zhou, and Z. Niu, "Joint Optimization of Cache Allocation and Content Placement in Urban Vehicular Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, 2018.

[8] N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment: Waiting or downloading now?" in *IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, CA, USA, 2016.

[9] *3GPP TS 23.214 - Architecture enhancements for control and user plane separation of EPC nodes*.

[10] *3GPP TS 23.501 - System architecture for the 5G System (5GS)*.

[11] V. Millnert, J. Eker, and E. Bini, "Achieving Predictable and Low End-to-End Latency for a Network of Smart Services," in *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, 2018.

[12] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Vehicular Micro Cloud in Action: On Gateway Selection and Gateway Handovers," *Elsevier Ad Hoc Networks*, vol. 78, pp. 73–83, 2018.

[13] F. Malandrino, C. Casetti, C.-F. Chiasserini, C. Sommer, and F. Dressler, "Content Downloading in Vehicular Networks: Bringing Parked Cars Into the Picture," in *23rd IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2012)*. Sydney, Australia: IEEE, 9 2012, pp. 1534–1539.

[14] F. Malandrino, C. Casetti, C.-F. Chiasserini, C. Sommer, and F. Dressler, "The Role of Parked Cars in Content Downloading for Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 14720546, pp. 4606 – 4617, 2014.

[15] H. Guo, J. Liu, H. Qin, and H. Zhang, "Collaborative Computation Offloading for Mobile-Edge Computing over Fiber-Wireless Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017.

[16] B. Li, "Optimal Offloading for Dynamic Compute-Intensive Applications in Wireless Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019.

[17] 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 *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, 2018.

[18] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-Based Cloudlet Relaying for Mobile Computation Offloading," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 18246123, pp. 11 181 – 11 191, 2018.

[19] 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 *IEEE Global Communications Conference (GLOBECOM)*, Washington, DC USA, 2016.

[20] Y. Ma, W. Liang, M. Huang, and S. Guo, "Profit Maximization of NFV-Enabled Request Admissions in SDNs," in *IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, UAE, 2018.

[21] 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," *CoRR*, vol. abs/1609.09368, 2016. [Online]. Available: http://arxiv.org/abs/1609.09368

[22] "The network simulator - ns-2", Available: http://www.isi.edu/nsnam/ns/.