TSOA: Two-State Offloading Algorithm from Users to Co-Located Vehicular Microclouds

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

***Abstract*—Offloading in edge computing scenarios is considered a prime solution to reduce computational time and also energy resources of the user equipment. This paper focuses on computation offloading from the user equipment to co-located vehicular microclouds. We derive the closed-form system metrics and cross-validate them with simulations. Through a comprehensive observation of vehicular microcloud behavior, the proposed two-state offloading algorithm (TSOA) provides the optimal offloading configuration in both planning and operating states. Finally, our evaluation demonstrates that the proposed TSOA performs optimally among the three offloading schemes.**

***Index Terms*—Edge Computing, Vehicular Microcloud, Computation Offloading, 5G Application.**

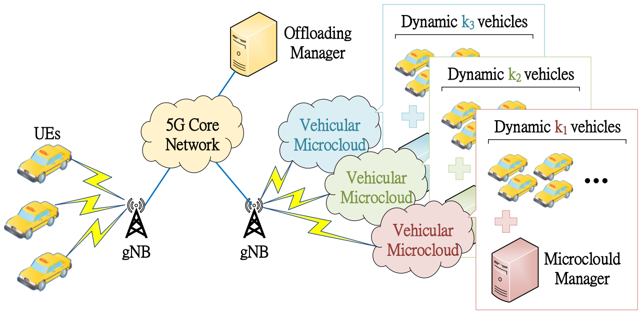


Figure 1. Overview of the offloading system

I. Introduction

Mobile edge computing has become an integral part of fifth-generation (5G) networks. Task offloading is considered one of the most prevalent applications [1], and supports next-generation metaverse applications [2]. Many vehicle-related applications require a significant amount of computational resources [3], an aspect usually overloaded for consumer vehicles, so offloading these tasks to road-side units (RSUs) represents a popular solution [4]. However, the RSU may be overloaded during the peak time [5].

As the deployment of 5G edge computing is rather slow, virtualized edge computing has been considered [6], often using vehicles as they contain significant computational resources [7]–[9]. In such a situation, one potential solution is offloading tasks through core networks to vehicular microclouds [9]–[11]. Parked vehicles often are considered part of such microclouds, which possess substantial idle computational resources [12], [13]. However, offloading to the same destination may overload it, whereas offloading to several vehicular microclouds on average is cost-ineffective because setting up a vehicle instance consumes time and resources.

This paper focuses on the computation offloading issue from the user equipment (UE), which usually refers to consumer vehicles (we call them 'UE' instead of 'UEs' in the rest of this paper), to co-located vehicular microclouds, which may have different instance scales. We reuse part of the model in our previous study [14] to describe a vehicular microcloud and propose the two-state offloading algorithm (TSOA) to obtain answers to the following questions:

- In the *planning* state before operating, how many vehicular microclouds are required? What scales of vehicular microclouds are more optimal? And what are the appropriate offloading ratios between these vehicular microclouds?
- In the *operating* state after planning, which vehicular microcloud should take the additional tasks blocked by other vehicular microclouds?

The rest of the paper is presented as follows: Section II describes selected related work. In Section III, we introduce the offloading system, derive the closed-form metrics, observe the behavior of a single vehicular microcloud, and study cases of labor division between co-located vehicle microclouds. Section IV presents the proposed TSOA. In Section V, we measure the proposed TSOA and analyze the results. Finally, we provide some conclusions in Section VI.

II. Related Work

Offloading in a vehicular environment has been explored from multiple angles. A complete architecture, the vehicular microcloud concept, has been proposed by Higuchi et al. [10]. In this paper, the authors focus on data-downloading services. Furthermore, a local coordination mechanism emerges to maximize the system's benefits. Following up, Higuchi et al. [15] focus on the consisting-event of vehicular microclouds to provide several services. Therefore, they establish a remote server to maintain statistics and design a mechanism to confirm when sufficient resources exist to form a vehicular microcloud. Krijestorac et al. [16] concentrate on offloading the computation to edge servers and vehicles via cellular networks and vehicle-to-vehicle communications separately, presenting a framework to optimize the resource assignment.

Several papers have been published focusing on selected components. Zhan et al. [4] focus on the offloading scenario from vehicles to roadside units and propose a deep reinforcement learning-based method to handle vehicle mobility and dynamic environment challenges. Wang et al. [17] focus on the computation offloading issue from a mobile device to nearby smart vehicles and propose corresponding algorithms to obtain the optimal candidate of the next cloudlet with the best switching time. Xu et al. [18] concentrate on offloading vehicular data from cellular networks to WiFi networks by using nearby vehicles' idle WiFi resources. Moreover, the authors utilize an M/G/1/K queueing model to analyze the efficiency. Celes et al. [19] focus on the actual characteristics of vehicular microclouds in real urban scenarios. They trace vehicle mobility and employ statistical modeling to identify the metric distribution.

The works mentioned above focus on vehicular microcloud features or offloading issues. However, there needs to be a study to assess the offloading configuration among co-located vehicular microclouds, which is this paper's main focus and contribution.

III. Proposed Offloading System

The offloading system's overview appears in Figure 1. UE offloads tasks to vehicular microclouds through the 5G core network and next-generation node B (gNB). The offloading manager can view the whole system to decide the offloading configurations, thereby answering the questions outlines above, and the system offloads tasks to each vehicular microcloud according to those configurations. If tasks are blocked because the destination vehicular microclouds are overloaded, the offload manager must yield new destinations. Additionally, each vehicular microcloud may have a different vehicle instance scale and a corresponding task capacity.

As for the vehicular microcloud, we reuse the model from our previous study [14] to describe its behavior. It was first proposed in another study [20]. In this model, the microcloud manager can dynamically set up and turn off every vehicle instance. If the number of tasks exceeds the number of vehicle instances, it sets an instance up to decrease the response time in the queue; in contrast, if the latter exceeds the former, it turns an instance off immediately to reduce cost. Furthermore, it turns off the setup state vehicle instance first.

A. Derivation of Closed-Form Metric

The notations are listed in Table I. In this paper, we assume that the calculation time per task follows an exponential distribution with a mean of μ^{-1} for each vehicle instance, the task arrival rate follows a Poisson distribution with mean λ, and the instance setup time follows an exponential distribution with a mean α^{-1}. All simulations in this paper set $K_x = 2\ k_x$, $\mu = 1$, and $\alpha = 0.02$ as the default value [14]. Additionally, we define each vehicle instance's cost as its resource consumption during the run time, including the setup procedure, and we finally define the objective function P to assess the system performance as

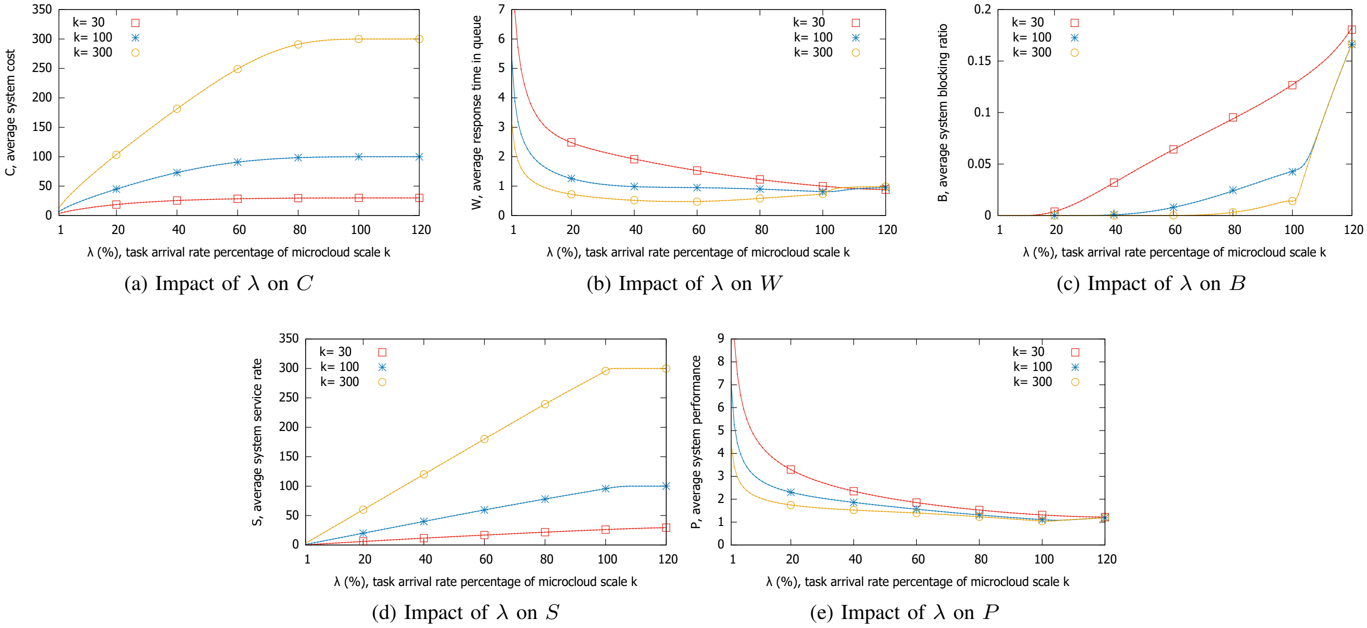
$$P = \frac{C(W+1)(B+1)}{S}. \tag{1}$$

Table I
List of Notations

Notation	Explanation
P	System performance
C	Average system cost
W	Average system response time in queue
B	Average system blocking rate
S	Average system service rate
λ	Task arrival rate in the system
μ	Service rate for each vehicle instance
α	Setup rate for each vehicle instance
X	The number of vehicular microclouds in the system
K_x	The number of maximum tasks can be accommodated in the x^{th} vehicular microcloud
k_x	The number of maximum vehicle instances in the x^{th} vehicular microcloud
C_x	Average cost in the x^{th} vehicular microcloud
W_x	Average response time in queue in the x^{th} vehicular microcloud
B_x	Average blocking rate in the x^{th} vehicular microcloud
S_x	Average service rate in the x^{th} vehicular microcloud
λ_x	Task arrival rate in the x^{th} vehicular microcloud

According to the definition in (1), a lower value of P implies superior system performance, such as lower cost, response time in the queue, and blocking rate, as well as higher service scale. We normalize the metrics to range $[0:1]$ before calculating P. Therefore, once a metric adds a value of 1 after normalizing, it degrades the impact scale of the metric from multiplication to addition. There are several considerations behind this design:

- *Viewpoints of System Operators and Consumers*: From the viewpoint of consumers, they pay the usage fee to offload tasks and anticipate the correct results as expected. In other words, it is acceptable when the response time W is shorter than a certain threshold, and further improving W may not be cost-effective. On the other hand, from the viewpoint of operators, the critical metrics are the cost C and service rate S because the former is overhead and the latter is profit.
- *Make Comparison Fairness*: To clarify the optimal offloading configuration, comparisons between vehicular microclouds with diverse instance and cooperation scales are essential. Because maxima of C and S always present the same value k_x in the model, their normalization impacts on P can offset each other to facilitate a fair comparison if system operators do not add an additional value to them. In contrast, the maxima of W and B are $\alpha^{-1} + \mu^{-1}$ and 1.0; they are constant and incur the same normalization impacts for all vehicular microclouds even if operators add a value of 1 to them.
- *Avoid drastic impacts on P*: In most cases, if task arrival rates λ_x are significantly less than the instance scales k_x, the system blocking rates B_x will be 0, zeroing the performance metric P. Therefore, system operators must add a value on B_x. Additionally, when B_x increases from 0 to 0.01, it creates a minor impact for operators and consumers, whereas it drastically changes P if operators use a multiplication scale on B_x.



(a) Impact of λ on C

(b) Impact of λ on W

(c) Impact of λ on B

(d) Impact of λ on S

(e) Impact of λ on P

Figure 2. Parameter study of a single vehicular microcloud

After clarifying the definition of P in (1), we provide the equations of the other system metrics, which need the corresponding metrics of vehicular microclouds as

$$C = \sum_{x=1}^{X} C_x \tag{2}$$

$$W = \frac{\sum_{x=1}^{X} W_x S_x}{S} \tag{3}$$

$$B = \frac{\sum_{x=1}^{X} B_x \lambda_x}{\lambda} \tag{4}$$

$$S = \sum_{x=1}^{X} S_x \tag{5}$$

Then we derive the closed-form metrics of the vehicular microclouds as

$$C_x = \sum_{i=0}^{k_x} \sum_{j=i}^{K_x} \pi_{i,j} \min(j, k_x) \tag{6}$$

$$W_x = \frac{\sum_{i=0}^{k_x} \sum_{j=i}^{K_x} \pi_{i,j} j}{\lambda_x (1 - \sum_{i=0}^{k_x} \pi_{i,K_x})} - \frac{1}{\mu} \tag{7}$$

$$B_x = \sum_{i=0}^{k_x} \pi_{i,K_x} \tag{8}$$

$$S_x = \sum_{i=0}^{k_x} \sum_{j=i}^{K_x} \pi_{i,j} i \tag{9}$$

In (6) to (9), $\pi_{i,j}$ denotes the probability of a vehicular microcloud state, which runs i vehicle instances calculating j tasks. The detailed derivation of $\pi_{i,j}$ can be found in our previous research [14] (not included because of the page limitation). Additionally, system operators decide the task arrival rate of each vehicular microcloud λ_x in (10) through an offloading algorithm, which can use the TSOA proposed in Section IV:

$$\lambda_x = \begin{cases} \lambda & \text{, if } X = 1 \\ \text{decide by offloading algorithm} & \text{, otherwise} \end{cases} \tag{10}$$

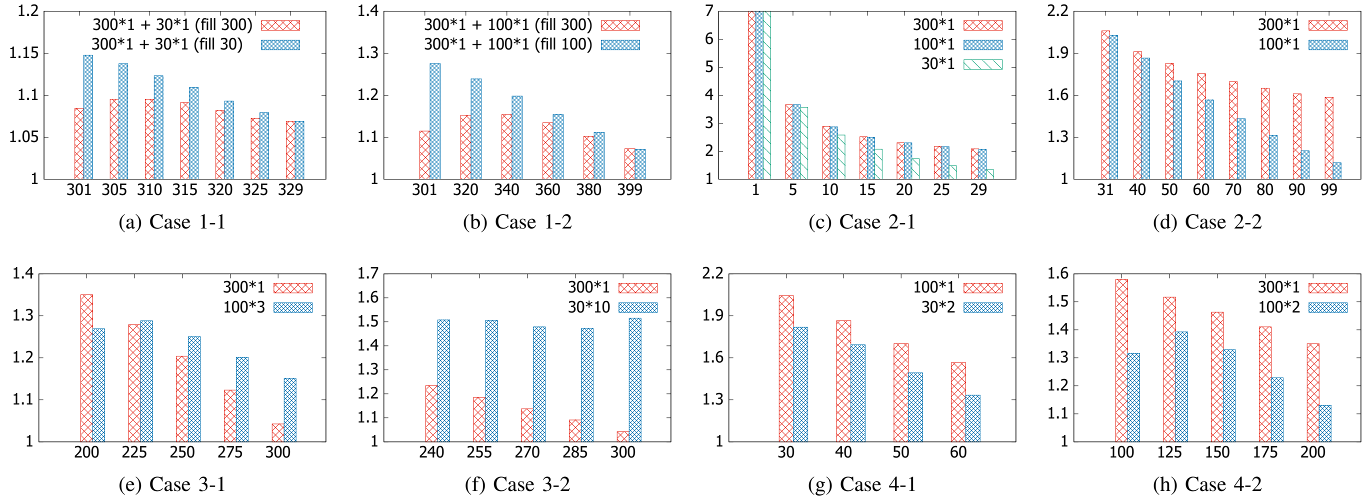
B. Observation of a Single Vehicular Microcloud

To observe the behavior of a single vehicular microcloud, we cross-validate the closed-form metrics with simulations by using the network simulator 2 (ns-2) [21], and the results match, as demonstrated in Figure 2. Additionally, because vehicular microclouds may have diverse vehicle instance scales k_x, we assume there are three primary sizes for the corresponding scenario:

- *Small* size ($k_x = 30$): It consists of vehicles parked along a road on both sides.
- *Medium* size ($k_x = 100$): It consists of vehicles parked in a regular parking lot.
- *Large* size ($k_x = 300$): It consists of vehicles parked in a parking tower of a hypermarket.

To present and compare vehicular microcloud behaviors between these primary sizes graphically, we set the x-axis as the percentage arrival rate $\lambda(\%)$ of each k_x. For example, if the value of $\lambda(\%)$ is 20, it implies λ is 6 for the small size, 20 for the medium size, and 60 for the large size.

As shown in Figure 2a, the cost increases with a higher task arrival rate, and the increasing trend flattens out accordingly. Thus, the overhead per task is more expensive when the task arrival rate is low because the instance setup procedure consumes time. During this period, the system accumulates tasks and sets more instances, which may turn off quickly and create additional waste.



(a) Case 1-1

(b) Case 1-2

(c) Case 2-1

(d) Case 2-2

(e) Case 3-1

(f) Case 3-2

(g) Case 4-1

(h) Case 4-2

Figure 3. Offloading case study of co-located vehicular microcloud configurations (impacts of λ (x-axis) on P (y-axis))

We study the response time in the queues in Figure 2b. It decreases with a higher task arrival rate because the system sets more vehicle instances up to consume tasks. However, the response time increases when the system overloads because there are no more instances to set up. At the same time, the blocking rate increases with a higher task arrival rate (see Figure 2c) because the system becomes busy, especially when it is overloaded. Additionally, the trend differs slightly in the small-size vehicular microcloud because its instances are terminated more easily and thus drastically increase the blocking rate. As shown in Figure 2d, the service rate increases with a higher task arrival rate until the system is overloaded. However, the service rate of the small-size vehicular microcloud still increases even in the overloaded range; this occurs because that vehicular microcloud turns off the vehicle instances easily and thus needs a higher task arrival rate to set up all instances.

Finally, in Figure 2e, the system performance improves with a higher task arrival rate until overloaded. Each scale's best percentage of task arrival rates $\lambda(\%)$ differs (119% for the small size, 105% for the medium size, and 101% for the large size). In other words, overloading benefits vehicular microclouds regarding their efficiency, especially the smaller ones.

Thus far, we can derive a brief conclusion for a single vehicular microcloud:

- The average cost per task is cheaper with a higher task arrival rate.
- The average response time in the queue per task shortens with a higher task arrival rate and a larger instance scale.
- The average blocking rate increases drastically with a higher task arrival rate and a smaller instance scale.
- The average service rate increases linearly with a higher task arrival rate until the system sets up all instances, and a smaller vehicular microcloud requires a higher task arrival rate to set up all vehicle instances.
- The best task arrival rate λ_x is 36 for the small-size vehicular microcloud, 105 for the medium-size one, and 303 for the large-size one.

Table II
Case 5 Configurations and Results

Case	λ	Config. 1	Config. 2	P$_{config.1}$	P$_{config.2}$
5-1	120	100*1+30*1	30*4	1.2127	1.3893
5-2	320	300*1+30*1	100*2+30*4	1.0818	1.2759

C. *Case Study of Co-located Vehicular Microclouds*

To clarify the behaviors among co-located vehicular mircoclouds, we compare different offloading configurations in five cases by running simulations. Figure 3 reveals the simulation results. In this figure, the x-axis represents the task arrival rate λ, and the y-axis describes the system performance P, in which a lower value implies improved performance. Additionally, the figure legend presents the system configuration. For example, the description "100*1+30*2" in a legend implies that the offloading system consists of one medium-size and two small-size vehicular microclouds.

Case 1. *Filling size choice*: In this case, we set two vehicular microclouds in different scales and prioritize offloading tasks to fill one completely ($\lambda_x = k_x$) to observe filling which scale performs better. The results of comparing different scales in Figures 3a and 3b prove that filling the large one is always desirable because of the lower C and B.

Case 2. *Remaining tasks handling*: In this case, we set one vehicular microcloud in diverse scales to handle the remaining tasks. In other words, the tasks are insufficient to fill any vehicular microcloud. The results in different scales in Figures 3c and 3d convey that asking the small one to handle the remaining tasks is most successful because of the lower C.

Case 3. *A single fully-filled larger vehicular microcloud vs. many fully-filled smaller ones*: In this case, we set only one scale of vehicular microclouds at a time to observe the difference between filling a larger one and filling many smaller ones. The results in different scales in Figures 3e and 3f display that filling the larger one is preferred because of the lower W and B. In other words, separating a large-size vehicular microcloud to fill several medium-size or small-size microclouds worsens system performance. Additionally, the performance gap between them decreases and even reverses with a lower task arrival rate because of a lower C.

Case 4. *A single partially-filled larger vehicular microcloud vs. many fully-filled smaller ones*: In this case, we set only one scale of vehicular microclouds at a time to observe the difference between offloading to a partially-filled larger one and fully-filled smaller ones. The results in different scales in Figures 3g and 3h prove that filling at least one smaller vehicular microcloud is better because of a lower C.

Case 5. *Single fully-filled larger and single partially-filled smaller vehicular microclouds vs. many fully-filled smaller ones*: In this case, we observe the difference between two configurations as displayed in Table II: Config. 1 sets a fully-filled larger vehicular microcloud and an additional partially-filled smaller one. In contrast, Config. 2 sets many fully-filled smaller vehicular microclouds. The results in different scales indicate that filling a larger vehicular microcloud, even with an additional partially-filled smaller one, outperforms because of a lower W and B.

Thus far, we can derive a brief conclusion for offloading tasks between co-located vehicular microclouds:

- Always prioritize filling a large vehicular microcloud instead of a smaller one.
- Always offload remaining tasks to a small vehicular microcloud instead of a larger one.
- Never separate a fully-filled vehicular microcloud into many smaller ones.
- Always offload tasks to fully-filled smaller vehicular microclouds instead of a partially-filled larger one (it relies on appropriately fine-grained instance scales).
- Always offload tasks to a fully-filled larger vehicular microcloud instead of fully-filled smaller ones, even if it creates an additional partially-filled smaller one.

IV. Two-State Offloading Algorithm (TSOA)

According to vehicular microcloud observations in Sections III-B and III-C, we propose TSOA offloading tasks from UE to co-located vehicular microclouds. TSOA runs for different purposes during the planning and operating states:

- *Planning* state: In this state, the offloading system makes decisions offline for planning the optimal offloading configurations, including the list of used vehicular microclouds and their task offloading ratios.
- *Operating* state: In this state, the offloading system only has a short time to make decisions online for assigning the appropriate vehicular microcloud as the new offloading destinations for the blocked tasks.

The proposed TSOA performs different reactions according to the system state as demonstrated in Algorithm 1. In the planning state, TSOA calls the corresponding subroutine to obtain the optimal offloading configurations and begins operating the offloading system based on it. On the other hand, in the operating state, TSOA waits to handle blocked tasks.

Algorithm 1: Two-State Offloading Algorithm (TSOA)

```
Input: $State, Model, \lambda, X, K_x, k_x, j_x$
Output: $List_{use}, \lambda_x, Target_{offload}$
if $State\ is\ Planning$ then
    $List_{use}$ and $\lambda_x \leftarrow$ call TSOA$_{planning}$
    offload $\lambda$ to $List_{use}$ with $\lambda_x$
if $State\ is\ Operating$ then
    for $State\ is\ Operating$ do
        if $Task\ is\ blocked$ then
            $Target_{offload} \leftarrow$ call TSOA$_{operating}$
            offload $Task$ to $Target_{offload}$
```

Algorithm 2: Planning State TSOA (TSOA$_{planning}$)

```
Input: $Model, \lambda, X, K_x, k_x$
Output: $List_{use}, \lambda_x$
for each vehicular microcloud do
    $Table_{best}$ $\leftarrow$ record best $\lambda_x$ through $Model$
$\lambda_{current} \leftarrow \lambda$
$List_{use} \leftarrow$ call TCOA$_{search}$
$\lambda_{block} \leftarrow$ sum of $\lambda_x \times B_x$ in $List_{use}$
$\lambda_{current} \leftarrow \lambda + \lambda_{block}$
$List_{use} \leftarrow$ call TCOA$_{search}$ again
return $List_{use}$ with $\lambda_x$
```

Algorithm 3: TSOA Search Scheme (TSOA$_{search}$)

```
Input: $\lambda_{current}, Table_{best}, X, K_x, k_x$
Output: $List_{use}, \lambda_x$
$List_{use} \leftarrow$ clear
for $\lambda_{current} > 0$ do
    if $\lambda_{current} <$ everyone in $Table_{best}$ then
        $List_{use}$ $add$ smallest instance scale one
        $\lambda_x \leftarrow \lambda_{current}$
        $\lambda_{current} \leftarrow 0$
        break
    $\lambda_{use} \leftarrow$ search $Table_{best}$ for largest one, which
      $< \lambda_{current}$ and is not in $List_{use}$
    $List_{use}$ $add$ the searched one
    $\lambda_x \leftarrow \lambda_{use}$
    $\lambda_{current} \leftarrow \lambda_{current} - \lambda_{use}$
return $List_{use}$ with $\lambda_x$
```

Algorithm 4: Operating State TSOA (TSOA$_{operating}$)

```
Input: $List_{use}, X, k_x, j_x$
Output: $Target_{offload}$
$List_{nonblocking} \leftarrow$ search non-blocking vehicular
  microclouds in $List_{use}$
$Target_{offload} \leftarrow$ search smallest instance scale in
  $List_{nonblocking}$
return $Target_{offload}$
```

Once a vehicular microcloud becomes blocked, TSOA calls the corresponding subroutine to obtain the new offloading destination and offloads the blocked tasks.

The subroutine for the planning state appears in Algorithm 2. It derives and records the best offloading ratio for each vehicular microcloud from the closed-form solution mentioned in Section III-A. Assessing the number of blocked tasks, TSOA$_{planning}$ calls the subroutine, which searches the optimal configurations according to the policies and roughly obtains the number of blocked tasks. Considering the task arrival rate and the blocked amount, TSOA$_{planning}$ calls the same subroutine again to update the offloading configuration for the final result returned to the TSOA.

Algorithm 3 presents the subroutine, which obtains the optimal offloading configurations according to the input task arrival rate and search policies summarized in Section III-C, used in TSOA$_{planning}$ to update the optimal configurations. Following the search policies, TSOA$_{search}$ searches the records $Table_{best}$ for the largest one that is less than the current task arrival rate under planning $\lambda_{current}$. TSOA$_{search}$ places the search results into the using list $List_{use}$, sets the corresponding offloading ratio λ_x, and updates the remaining $\lambda_{current}$. Before returning the configurations, TSOA$_{search}$ selects the remaining smallest vehicular microcloud additionally to handle the remaining tasks if $\lambda_{current}$ is not zero.

Finally, in Algorithm 4, the subroutine for the operating state attempts to find the optimal vehicular microcloud as the offload target for the blocked tasks. As the first step, TSOA$_{operating}$ searches the list $List_{use}$ to obtain information about all non-blocking vehicular microclouds. Next, by searching the obtained list $List_{nonblocking}$, TSOA$_{operating}$ selects the one with the smallest instance scale and returns it to TSOA as the final offloading target. In TSOA$_{operating}$, the search range of the target is limited to using list $List_{use}$. In other words, it cannot invite a new vehicular microcloud to join the offloading system.

V. Evaluation

To assess the proposed TSOA, we further propose two intuitive methods as baselines for comparison:

- *Distributed* scheme: It distributes tasks uniformly with the same percentage of the instance scales to avoid blocking ($\lambda_x = \lambda \times k_x/k_{sum}$) and offloads blocked tasks to the idlest target (the largest gap between K_x and j_x).
- *Centralized* scheme: It properly centralizes tasks to large vehicular microclouds ($\lambda_x = k_x \times 90\%$, higher if λ/k_{sum} > 90%) to achieve cost-effectiveness and avoid blocking, offloading blocked tasks to the idlest target as well.

We provide a composite scenario including one large-size, three medium-size, and six small-size vehicular microclouds (300*1+100*3+30*6) for the offloading system, and Figure 4 exhibits the simulation results.

In terms of cost C, the distributed scheme sets instances in each vehicular microcloud with the ratio of λ_x to k_x, which is relatively low compared to other methods, thus causing poor cost-efficiency. Besides the cost-efficiency factor of the offloading ratio, the centralized scheme performs slightly worse than TSOA by offloading the remaining tasks to a large vehicular microcloud.

In terms of response time W, the distributed scheme possesses longer W in low task arrival rates λ because it is more likely to consume a non-negligible amount of time to set up instances before calculating tasks. In contrast, the centralized scheme yields shorter W in the same range of λ because it sets up more instances to consume the task buffers.

In terms of blocking rate B, in the range of the lower task arrival rate λ, TSOA incurs relatively higher B compared to other methods because TSOA employs smaller vehicular microclouds to save cost. A finer granularity of instance scales k_x for vehicular microclouds eases and decreases this blocking feature with increasing λ. In other words, TSOA relies on the fine granularity of k_x, as mentioned in Section III-C.

In terms of service rate S, the values in all schemes appear close to the task arrival rate λ because the system offloads blocked tasks to new destinations for computational services.

Finally, in terms of performance P, the proposed TSOA always offers advantages over the baselines because of the combined effect of the aforementioned reasons.

VI. Conclusion

This paper considers offloading computational tasks from UE to co-located vehicular microclouds. We derive closed-form metrics and summarize the features of single and co-located vehicular microclouds with diverse instance scales and task arrival rates. The proposed TSOA yields the optimal offloading configuration and ratios in the planning state as well as the offloading target for the blocked tasks in the operating state. Our evaluation exhibits that TSOA provides advantages over other intuitive schemes. In the future, we will consider additional factors, such as diverse tasks and vehicle instance types.

Acknowledgment

This work was supported in part by the National Science and Technology Council of Taiwan under grant numbers 111-2221-E-A49-093-MY3, 111-2218-E-A49-023, and 111-3114-E-A49-001 as well as by the Federal Ministry of Education and Research (BMBF, Germany) within the 6G Research and Innovation Cluster 6G-RIC under Grant 16KISK020K.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, Oct. 2017.

[2] K. Li, Y. Cui, W. Li, T. Lv, X. Yuan, S. Li, W. Ni, M. Simsek, and F. Dressler, “When Internet of Things meets Metaverse: Convergence of Physical and Cyber Worlds,” *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4148–4173, Mar. 2023.

[3] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, 2014.

[4] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, “Deep Reinforcement Learning-Based Computation Offloading in Vehicular Edge Computing,” in *IEEE GLOBECOM 2019*. Waikoloa, HI: IEEE, Dec. 2019.

[5] Z. Y. Rawashdeh and S. M. Mahmud, “Admission Control for Roadside Units Based on Virtual Air-Time Transmissions,” in *IEEE GLOBECOM 2011*. Houston, TX: IEEE, Dec. 2011.

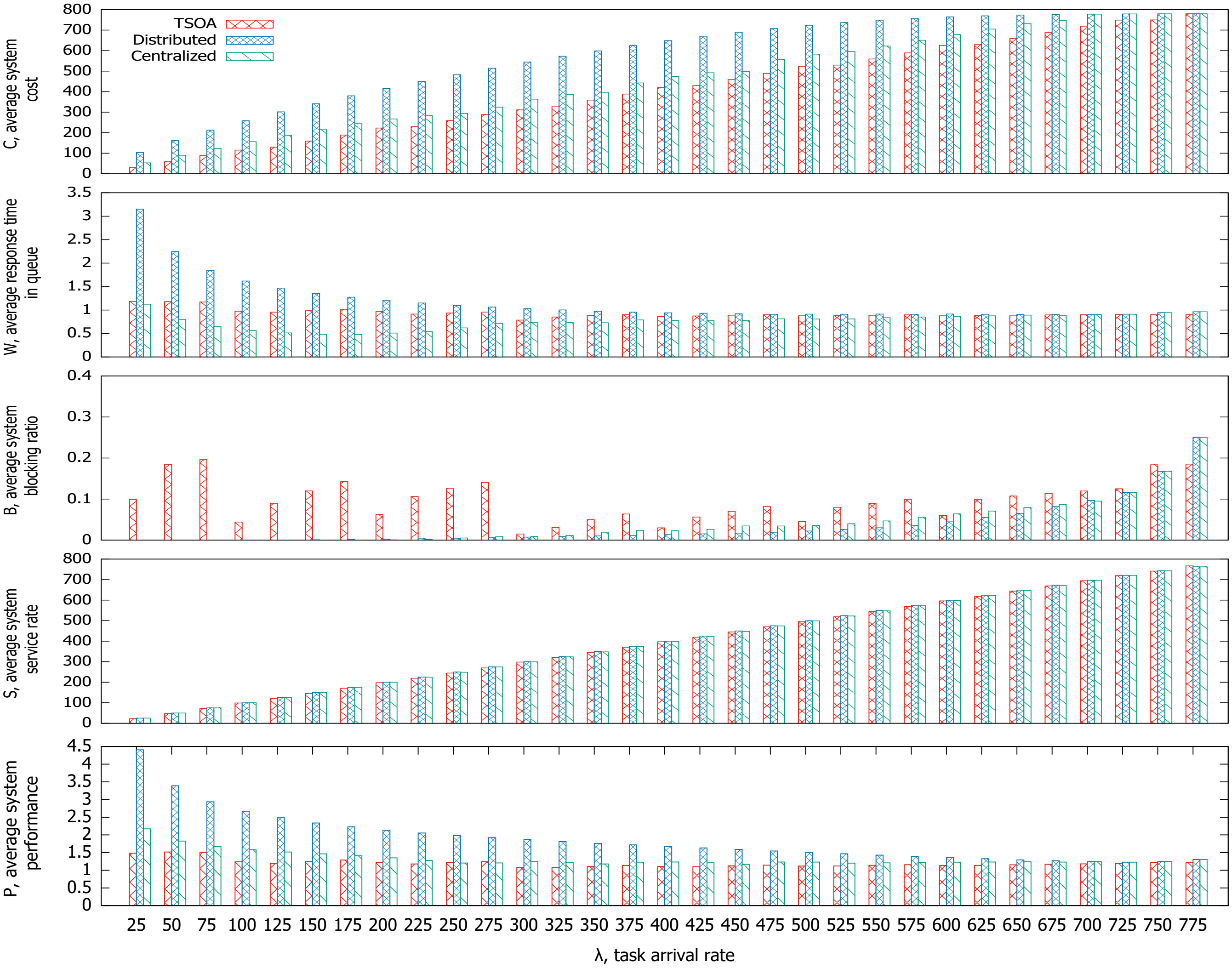


Figure 4. Impact of offloading schemes with diverse arrival rate

[6] F. Dressler, C. F. Chiasserini, F. H. P. Fitzek, H. Karl, R. Lo Cigno, A. Capone, C. E. Casetti, F. Malandrino, V. Mancuso, F. Klingler, and G. A. Rizzo, "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," *IEEE Network*, vol. 36, no. 3, pp. 24–31, May 2022.

[7] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[8] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual Edge Computing Using Vehicular Micro Clouds," in *IEEE ICNC 2019*. Honolulu, HI: IEEE, Feb. 2019.

[9] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative Task Offloading in Vehicular Edge Multi-Access Networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48 – 54, Aug. 2018.

[10] T. Higuchi, R. V. Rabsatt, M. Gerla, O. Altintas, and F. Dressler, "Cooperative Downloading in Vehicular Heterogeneous Networks at the Edge," in *IEEE GLOBECOM 2019, MobileEdgeCom Workshop*. Waikoloa, HI: IEEE, Dec. 2019.

[11] 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, Sep. 2018.

[12] F. Malandrino, C. E. Casetti, C. F. Chiasserini, C. Sommer, and F. Dressler, "Content Downloading in Vehicular Networks: Bringing Parked Cars Into the Picture," in *IEEE PIMRC 2012*. Sydney, Australia: IEEE, Sep. 2012, pp. 1534–1539.

[13] ——, "The Role of Parked Cars in Content Downloading for Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4606–4617, Nov. 2014.

[14] 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 *IEEE GLOBECOM 2020*. Taipei, Taiwan: IEEE, Dec. 2020.

[15] T. Higuchi, F. Dressler, and O. Altintas, "How to Keep a Vehicular Micro Cloud Intact," in *IEEE VTC 2018-Spring*. Porto, Portugal: IEEE, Jun. 2018.

[16] E. Krijestorac, A. Memedi, T. Higuchi, S. Ucar, O. Altintas, and D. Čabrić, "Hybrid Vehicular and Cloud Distributed Computing: A Case for Cooperative Perception," in *IEEE GLOBECOM 2020*. Taipei, Taiwan: IEEE, Dec. 2020.

[17] 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. 11, pp. 11 181–11 191, Nov. 2018.

[18] 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 GLOBECOM 2018*. Abu Dhabi, United Arab Emirates: IEEE, Dec. 2018.

[19] C. Celes, A. Boukerche, and A. A. F. Loureiro, "Revealing and Modeling Vehicular Micro Clouds Characteristics in a Large-Scale Mobility Trace," in *IEEE ICC 2021*. Virtual Conference: IEEE, Jun. 2021.

[20] Y. Ren, T. Phung-Duc, J.-C. Chen, and Z.-W. Yu, "Dynamic Auto Scaling Algorithm (DASA) for 5G Mobile Networks," in *IEEE GLOBECOM 2016*. Washington, D.C.: IEEE, Dec. 2016.

[21] *The network simulator - ns-2*, available: http://www.isi.edu/nsnam/ns/.