Offloading Utility Optimization in Parked Vehicular Edge Computing

Xuan-Qui Pham, and Dong-Seong Kim
ICT Convergence Research Center, Kumoh National Institute of Technology, Korea
Email: {pxuanqui, dskim}@kumoh.ac.kr

***Abstract*—Parked vehicular edge computing (PVEC) has recently been introduced as a new paradigm that makes use of parked vehicles to support edge servers in computation offloading. However, most existing research does not consider the local computing capacity of requesting devices and/or the resource cost for offloading tasks. In this paper, we propose a partial computation offloading scheme that allows mobile devices (MDs) to utilize both local and remote resources for parallel task execution in PVEC. The goal is to maximize the total offloading utility of MDs, which is defined as the difference between the benefit of latency reduction and the cost of using computing and networking resources. Simulation results show that the proposed scheme can improve the total offloading utility compared to conventional schemes.**

***Index Terms*—parked vehicular edge computing, partial offloading, resource allocation.**

I. Introduction

With the ever-increasing number of smart mobile devices (MDs), various delay-sensitive and compute-intensive applications are emerging, such as face recognition, virtual/augmented reality, natural language processing, etc. However, MDs are in general resource-constrained and cannot meet the computational demand of these services. Multi-access edge computing (MEC) [1] has been introduced to address this challenge by enabling MDs to offload their tasks to edge servers that provide cloud computing capabilities at WiFi access points and base stations. Despite being widely studied, computation offloading in conventional MEC still presents critical challenges [2], [3]. Considering the high capital expenditure and operating expense, it is infeasible to deploy dense edge servers to face the rapid proliferation of MDs.

Meanwhile, the opportunistic resources offered by a large pool of vehicles in urban areas can be brought together and leveraged to relieve the resource congestion of edge servers during peak times. For example, a collaborative computation offloading system consisting of mobile vehicles, an edge server and the remote cloud is proposed in [4]. However, the high mobility of mobile vehicles may cause intermittent connectivity during the offloading process.

By contrast, parked vehicles (PVs) are relatively static for a long period, and thus offer more reliable task offloading performance. Recent research introduced the concept of parked vehicular edge computing (PVEC), which employs PVs as static edge computing nodes to assist edge servers in offloading tasks from MDs. For example, the authors in [5], [6] utilized the Stackelberg game framework to investigate the problem of workload allocation between an edge server and multiple PVs to minimize the overall offloading cost. In [7], a computation offloading scheme for PVEC was proposed to minimize the latency of offloaded MDs' tasks. However, it is worth noting that the above-mentioned works concentrated on full offloading mechanisms in which the computation task is completely offloaded for remote execution without considering the local computing resources of requesting devices. Partial offloading mechanisms, on the other hand, determine the appropriate offloading ratio and process tasks in parallel using both local and remote resources. Few studies on partial offloading were proposed. However, they were limited by not considering computing resources of edge servers [8] or the costs associated with resource usage [9].

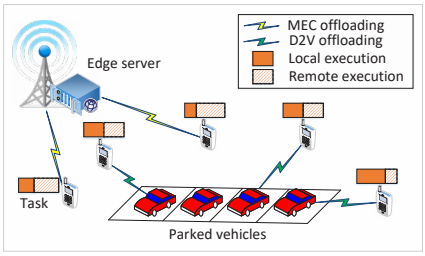


Fig. 1: System model.

To fill this gap, partial computation offloading in PVEC is investigated in this work. We first formulate the total offloading utility maximization problem with joint consideration of task assignment, offloading ratio, and resource allocation. The offloading utility of each MD is defined in terms of the task latency reduction and the associated costs for both computing and communication resources. Then we propose a partial offloading scheme, in which a closed-form expression of the optimal offloading ratio and resource allocation is derived, and a heuristic algorithm is used to solve the task assignment problem. Finally, through simulation results, we verify the superior performance of our proposal over conventional schemes in terms of total offloading utility.

II. System Model and Problem Formulation

Fig. 1 shows the PVEC paradigm that includes a base station (BS) furnished with an edge server, a set $\mathcal{N}$ of MDs, and a set $\mathcal{M}$ of PVs. The BS is employed as a trustful manager to analyze the parking behaviors of PVs to periodically choose reliable PVs as task processors and add them to set $\mathcal{M}$. The PV selection can be done as in [5]. Considering a discrete-time model, MDs and PVs are assumed to be unchanged within a

time slot while they may change across time slots. Each MD requests a computation task denoted by $\{D_i, C_i\}$, where D_i represents the size of input data, C_i represents the computation density. For partial offloading, each MD will locally process $(1-\lambda_i)D_i$ bits and offload the rest to the edge server 0 or a PV through device-to-vehicle (D2V) communication, wherein $\lambda_i \in [0,1]$ is the variable of the offloading ratio.

The overall latency of MD i includes the local computation latency t_{ij}^l at the MD and the offloading latency t_{ij}^{off} from MD i to computing node $j \in \mathcal{M} \cup \{0\}$ as follows:

$$t_{ij}^l = \frac{(1-\lambda_i)D_i C_i}{f_i^l}. \tag{1}$$

$$t_{ij}^{off} = \frac{\lambda_i D_i}{R_{ij}} + \frac{\lambda_i D_i C_i}{f_{ij}}, \tag{2}$$

where f_i^l represents the local computing capacity of MD i, R_{ij} is the transmission rate between MD i and computing node j as in [10], and f_{ij} denotes the allocated computing resources of computing node j. In this work, we consider that at a time slot, the edge server can process multiple computation tasks while a PV can serve at most one MD due to the idle resource constraint. As such, in the case of D2V offloading, $f_{ij} = F_j$, where F_j denotes the computation capacity of the PV j.

The overall latency is then calculated as $t_{ij} = max\{t_{ij}^l, t_{ij}^{off}\}$.

Let $\mathbf{a} = \{a_{ij} | i \in \mathcal{N}, j \in \mathcal{M} \cup \{0\}\}$ be the task assignment vector. From the perspective of MDs, the utility is determined by the difference between the gain of latency reduction and the resource cost:

$$W_i = \sum_{j \in \mathcal{M} \cup \{0\}} a_{ij} W_{ij} = \sum_{j \in \mathcal{M} \cup \{0\}} a_{ij}[G_t(t_i^l - t_{ij}) - P^{trans}\lambda_i D_i - P_j^{exe} f_{ij}], \tag{3}$$

where t_i^l is the local computation latency of the whole task, G_t is the unit gain of latency reduction, and P^{trans} and P_j^{exe} p_j are the unit price of communication and computing resources.

The problem formulation is then expressed as follows:

$$\max_{\boldsymbol{\lambda}, \boldsymbol{a}, \boldsymbol{f}} \quad \sum_{i \in \mathcal{N}} W_i \tag{4}$$

$$\text{s.t.} \sum_{j \in \mathcal{M} \cup \{0\}} a_{ij} \le 1, \quad \forall i \in \mathcal{N}, \tag{4a}$$

$$\sum_{i \in \mathcal{N}} a_{ij} \le 1, \quad \forall j \in \mathcal{M}, \tag{4b}$$

$$\sum_{i \in \mathcal{N}} a_{i0} f_{i0} \le F_0, \tag{4c}$$

$$t_{ij}^l = t_{ij}^{off}, \quad \forall i \in \mathcal{N}, j \in \mathcal{M} \cup \{0\}, \tag{4d}$$

$$0 \le \lambda_i \le 1, a_{ij} \in \{0,1\}, f_{i0} \ge 0, \quad \forall i \in \mathcal{N}, j \in \mathcal{M} \cup \{0\}. \tag{4e}$$

where (4a) and (4b) denotes the association constraints between tasks and computing nodes. (4c) ensures the total resource allocation of the edge server does not exceed its computing capacity. (4d) indicates the conditions to reach the minimum latency in the partial computation offloading [10]. And (4e) are the variables' boundaries.

III. Proposed Solution

The original problem can be decomposed into two subproblems as follows:

A. Offloading Ratio and Resource Allocation Policy

1) MEC Offloading: Given $t_{ij}^l = t_{ij}^{off}$, the allocated computing resources are derived from (1) and (2) as follows:

$$f_{i0} = \frac{\lambda_i D_i C_i}{(1-\lambda_i)D_i C_i / f_i^l - \lambda_i D_i / R_{i0}}. \tag{5}$$

Given $t_{i0} = (1-\lambda_i)t_i^l$, substitute (5) into W_{ij} in (3), the utility of MD i is:

$$W_{i0}(\lambda_i) = (\frac{G_t D_i C_i}{f_i^l} - P^{trans} D_i)\lambda_i - \frac{P_0^{exe} D_i C_i R_{i0} f_i^l \lambda_i}{D_i C_i R_{i0} - (D_i C_i R_{i0} + D_i f_i^l)\lambda_i} \tag{6}$$

Taking the derivatives of W_{i0} with respect to λ_i, we have

If $K = (\frac{G_t D_i C_i}{f_i^l} - P^{trans} D_i) \le 0$, then $\frac{\partial W_{i0}}{\partial \lambda_i} < 0$. W_{i0} monotonically decreases and its minimum is at $\lambda_i^* = 0$.

If $K > 0$, let $\frac{\partial W_{i0}}{\partial \lambda_i} = 0$, we find the local maximum point $\overline{A} = \frac{D_i C_i R_{i0} - D_i C_i R_{i0} f_i^l \sqrt{\frac{P_0^{exe}}{G_t D_i C_i - P^{trans} D_i f_i^l}}}{D_i C_i R_{i0} + D_i f_i^l}$.

Hence, the optimal offloading ratio of MD i in the MEC offloading mode is obtained as follows:

$$\lambda_i^* = \begin{cases} 0, & \text{if} \quad \frac{g_t D_i C_i}{f_i^l} - P^{trans} D_i \le 0 \quad \text{or} \quad \overline{A} \le 0 \\ \overline{A}, & \text{otherwise,} \end{cases} \tag{7}$$

2) D2V Offloading: For the D2V offloading mode, the optimal offloading ratio to minimize the latency is given as

$$\lambda_i^* = \frac{D_i C_i R_{ij} F_j}{D_i C_i R_{ij} f_i^l + (D_i C_i R_{ij} + D_i f_i^l) F_j}. \tag{8}$$

B. Task Assignment

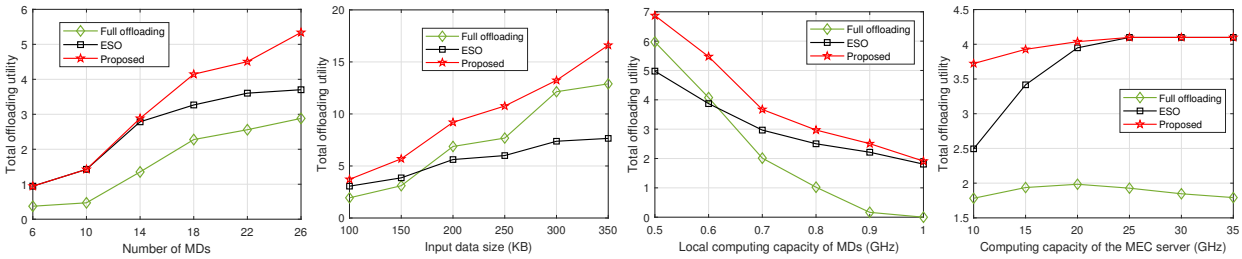
Given the obtained offloading ratio and resource allocation, we can rewrite the problem (4) as follows:

$$\max_{\boldsymbol{a}} \quad \sum_{i \in \mathcal{N}} W_i \tag{9}$$
$$\text{s.t.} \quad (4a), (4b), (4c), a_{ij} \in \{0,1\}.$$

Then we separate the problem (9) into the MEC mode optimization and D2V mode optimization. In the proposed solution, the computation tasks are first greedily assigned to the edge server until reaching its maximum computing capacity, and then the rest of the tasks are processed by the D2V offloading mode. The MEC mode optimization is a typical 0-1 knapsack problem, in which f_{i0}^* and W_{i0} denote the weight and value of each item along with a maximum weight capacity F_0. This problem can be solved by the greedy approximation algorithm, in which the elements (tasks) are first sorted in descending order of value per unit of weight (i.e., W_{i0}/f_{i0}), then the sorted elements are inserted into the knapsack until its maximum capacity is reached. Meanwhile, the D2V mode optimization is an assignment problem between $\mathcal{N}$ tasks and $\mathcal{M}$ PVs. Here, we utilize the *matchpairs* function in the Matlab toolbox to solve this problem.

IV. Evaluation Results and Discussions

We examine a coverage area of 200m $\times$ 200m, where a MEC-enabled base station is centrally located, and 20 MDs along with 10 PVs are distributed within the coverage, unless otherwise stated. The simulation parameters are summarized in Table I [10]. We compare our proposal with two baselines:



(a) (b) (c) (d)

Fig. 2: Total offloading utility versus: (a) Number of MDs, (b) Input data size, (c) Local computing capacity of MDs, (d) Computing capacity of the edge server.

TABLE I: Simulation parameters.

Parameter	**Value**
A task: D_i and C_i	[50,150] KB; [1,2] mc/KB
Computing capacity of the edge server	15 GHz
Computing capacity of MDs and PVs	[0.5,1] GHz; [1,1.5] GHz
MDs' transmission power	30 dBm
Channel bandwidth, path loss exponent, and noise	10 MHz; -3.4; -114dBm
Unit gain of latency reduction	2.5
Unit price of computing resources	0.1
Unit price of communication resources	0.15

Full offloading (FO) scheme where each MD's task is fully processed by the MD or fully offloaded to the edge server or a PV; and *Edge server only (ESO)* scheme where each computation task is partially offloaded by only the MEC offloading mode.

Fig. 2a shows that the total offloading utility increases as the number of MDs N increases. For a small N, our proposal has the same total utility as the *ESO* scheme since the tasks can be efficiently offloaded using only the edge server. As N increases, the proposed scheme can achieve better performance than the *ESO* scheme by optimizing both the MEC and D2V offloading modes. Moreover, the total offloading utility can be greatly improved in the partial offloading schemes, compared to that of the *FO* scheme. This phenomenon occurs because the partial offloading schemes can increase the number of tasks to be offloaded by determining the appropriate offloading ratio to tradeoff between the benefit of latency reduction and the costs associated with resource usage.

Fig. 2b illustrates the total offloading utility under different input data sizes. Compared to the *ESO* scheme, the utilization of D2V offloading mode can significantly enhance the total offloading utility as the input data size increases. Besides, offloading the whole task with increased data size can negatively impact the offloading utility due to the increase in transmission latency and communication resource cost. As a result, the *FO* scheme shows worse performance than our proposed scheme.

As the local computing capacity of MDs increases, a larger number of tasks can be executed locally, leading to a reduction in the total offloading utility as shown in Fig. 2c. When the local computing capacity is large enough, the total utility can even decrease to 0, as in the *FO* scheme. Fig. 2d shows that when the computing capacity of the edge server F_0 increases, the performance of the *FO* scheme first increases but then decreases. By contrast, the performance of the partial offloading schemes can continue to increase up to a certain value as F_0 increases. Moreover, our proposal can obtain better performance compared to the baselines.

V. Conclusion

This paper introduced a utility optimization approach to the PVEC paradigm. We proposed an efficient partial computation offloading scheme optimizing both the MEC and D2V offloading modes. The experimental results demonstrate the superiority of our approach over traditional baseline methods. For the future work, the research on the security aspect of the PVEC paradigm will be taken into account.

Acknowledgment

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Innovative Human Resource Development for Local Intellectualization support program (IITP-2023-2020-0-01612) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation). It was also supported by Priority Research Centers Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2018R1A6A1A03024003).

References

[1] P. Cruz *et al.*, "On the edge of the deployment: A survey on multi-access edge computing," *ACM Comput. Surv.*, vol. 55, no. 5, dec 2022.

[2] C. Jiang *et al.*, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131 543–131 558, 2019.

[3] X.-Q. Pham *et al.*, "Joint service caching and task offloading in multi-access edge computing: A qoe-based utility optimization approach," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 965–969, 2021.

[4] ——, "Joint node selection and resource allocation for task offloading in scalable vehicle-assisted multi-access edge computing," *Symmetry*, vol. 11, no. 1, 2019.

[5] X. Huang *et al.*, "Parked vehicle edge computing: Exploiting opportunistic resources for distributed mobile applications," *IEEE Access*, vol. 6, pp. 66 649–66 663, 2018.

[6] Q. Peng *et al.*, "A task assignment scheme for parked-vehicle assisted edge computing in iov," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–5.

[7] A. Zhou, X. Ma, S. Gao, and S. Wang, "Providing reliable service for parked-vehicle-assisted mobile edge computing," *ACM Trans. Internet Technol.*, vol. 22, no. 4, nov 2022.

[8] X. Huang *et al.*, "Task-container matching game for computation offloading in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 10, pp. 6242–6255, 2021.

[9] X. Hu *et al.*, "Joint load balancing and offloading optimization in multiple parked vehicle-assisted edge computing," *Wirel. Commun. Mob. Comput.*, vol. 2021, pp. 1–13, 11 2021.

[10] X.-Q. Pham *et al.*, "Partial computation offloading in parked vehicle-assisted multi-access edge computing: A game-theoretic approach," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 10 220–10 225, 2022.