Collaborative Deadline-sensitive Multi-task Offloading in Vehicular-Cloud Networks

Prasanna Kumar*§, Sushma S A*†§, K Chandrasekaran*, Sourav Kanti Addya*
Cloud and Smart System Services Lab,
*Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, India,
†Siddaganga Institute of Technology,Tumakuru.
Email:{prasanna26.cse, sushmasa.sit}@gmail.com, {kch, souravkaddya}@nitk.edu.in

***Abstract*—With the growing technological advancements in the Internet and advanced functionalities in vehicular networks, it becomes crucial to execute tasks quickly and efficiently. However, the limited onboard computational capacity and vehicle mobility make it challenging to accomplish latency-sensitive tasks efficiently. Task offloading provides a promising solution to overcome these challenges. Cloud data centers provide efficient solutions, but returning the results to the vehicles takes longer due to the large physical distance. Leveraging edge servers to execute latency-sensitive tasks provides a fast, interactive response and less transmission cost. However, in a dynamic network, vehicles will be in constant motion with varying speeds, resulting in frequent handoffs from one base station to another. Our proposed work aims to select the optimal nodes to perform binary offloading with minimum cost using the collaborative vehicular network. We use a greedy-based offloading approach to address these challenges and achieve better quality-of-service and quality-of-experience in a dynamic environment to minimize costs, delay reduction ratio, and satisfaction ratio. The proposed work outperforms the baseline by 60.44%, 53.43% in reducing total system cost, delay reduction ratio, and 36% improvement in the satisfaction ratio compared to baseline algorithms.**

***Index Terms*—Binary offloading, Mobility, Collaborative networks, Delay reduction ratio, Cost Minimization.**

I. Introduction

Autonomous vehicles are rapidly growing with rich, advanced features that provide smarter, intelligent, and connected vehicles to make reliable and safe driving decisions [1]. In recent years, metropolitan cities have witnessed an increase in the number of vehicles market size valued at USD 1,500.3 billion in 2022 and is projected to grow from USD 1,921.1 billion in 2023 to USD 13,632.4 billion by 2030, exhibiting a compound annual growth rate of 32.3% during the forecast period [2]. This has increased urbanization and technological advancement in automotive solutions. Autonomous vehicles comprise various in-built sensors, wireless communication devices, and processing and storage capabilities to support vehicular applications such as image-aided navigation, collision detection, and intelligent vehicle control [3] [4] [5]. The onboard computation unit of the autonomous vehicle cannot execute such complex tasks due to a lack of storage and computation power. In such cases, the offloading technique provides efficient and reliable solutions. Even though the cloud data centers are rich with computation and storage capabilities, they have limitations concerning *delay* and *extra transmission costs* that degrade the *system performance* [6]. To provide a better quality-of-service (QoS), the *edge server* acts as a middle layer between the vehicle and the cloud, reducing the *latency* and *energy*, thus increasing the battery life of the vehicles [7]. These servers also provide faster interactive responses [8]. In the case of full offloading, the maximum delay is experienced when many tasks are offloaded to distant servers. So, it is not recommended for latency-sensitive applications like collision avoidance, where the response must be sent in very short intervals [9]. In such cases, binary offloading helps execute the task in the local (vehicle) or remote (edge/cloud) [10]. Since binary offloading splits the workload across local vehicles and edge service, providing efficient resource utilization, reduced latency, and low communication overhead [11].

In the case of vehicular networks, the offloading decisions are adapted dynamically due to mobility patterns and location information of vehicles. The optimal selection of edge servers considering latency constraints helps in binary offloading with dynamic decisions, which reduces the likelihood of offloading to a distant server [12]. In some cases, tasks can be prioritized based on the deadline and vehicle location so that processing can be performed within the given timeline [13]. Therefore, the vehicular network uses the Vehicle-to-infrastructure (V2I) to efficiently make dynamic offloading decisions and allocate computational resources to provide QoS to users [14]. In a dynamic environment, wireless networks experience variable conditions such as signal fading, interference, and congestion [15]. Therefore, robust offloading mechanisms that can adapt to unpredictable network conditions must be developed [16]. From this perspective, many researchers have done significant work in optimizing latency, energy, resource allocation, and revenue using heuristic, metaheuristics, and exact approaches such as matching theory [17], genetic algorithm (GA) [18], and convex optimization [19] *etc.* Gu *et al.* formulated an energy-efficient cost minimization problem considering the impact of mobility of vehicles using a two-level optimization distributed algorithm [8]. In [20], Liu *et al.* proposed an algorithm to increase the revenue for task offloading using a Genetic Allocation Algorithm to reduce the

§ Equal contribution

number of migrations due to the mobility of the vehicular nodes. In another work, Raza *et al.* analyses the trade-off between computation time and energy consumption using the MACTER approach [21]. Luo *et al.* proposed a swarm optimization-based computation offloading algorithm to obtain the pareto-optimal solutions in minimizing the delay and cost while performing task offloading in the vehicular network [22]. However, task dependency management, network instability, overutilization of resources, and their scalability in large-scale vehicular networks remain an open challenges.

The main contributions of this work are as follows. (i) proposed *greedy-based offloading*, which is a type of random optimization to decide whether to offload vehicular tasks in on-board units, roadside units (RSU), base stations (BS), edge/cloud in a mobility-aware environment to optimize the *cost* of executing tasks. (ii) to overcome the imbalance in allocation of resources in mobility vehicular network we propose an efficient resource allocation based on the *availability* and *sojourn time* of the tasks. (iii) schedule the tasks considering the deadline to provide less *task failure* and efficient execution.

II. System Model

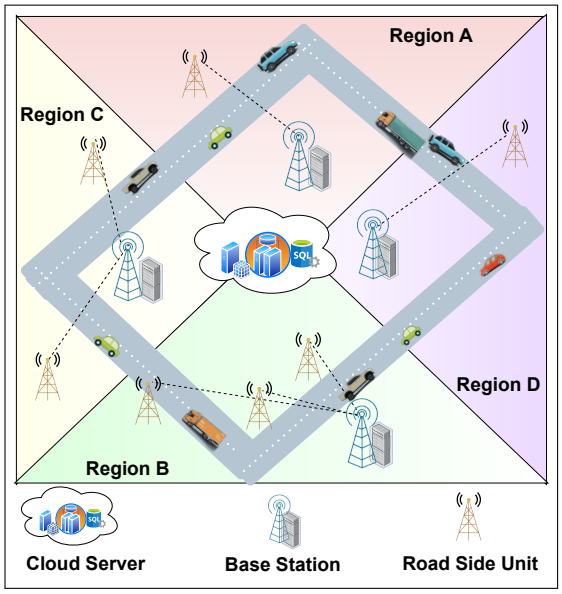


Fig. 1: A scenario of a city divided into four regions with multiple vehicle entities to perform multi-task binary offloading in a collaborative vehicular network.

Our system model in Fig. 1 comprises vehicles, base stations, roadside units, and cloud as entities for executing tasks. We consider a city that is divided into four different regions. Each region has multiple RSUs and a single base station in a collaborative vehicular network. Cloud servers are at a large distance and contain large computation power. In Fig. 2 shows the onboard unit and decision taken by the vehicle based on the battery power and its computation capability. The intra-region edge servers contain RSUs and base stations of the particular region where the vehicle is already present. The edge controller in the base station makes the offloading destination decision. Consider a set of vehicular tasks denoted by $\mathcal{V} = \{v_1, v_2, v_3....v_x\}$ and task set $\mathcal{T} = \{t_1, t_2, t_3...t_n\}$. The base station is located in each region denoted by $\mathcal{N}_b = \{b_1, b_2, b_3, b_4\}$, and they communicate using wired communication. The RSUs are also the vehicle entities whose positions are fixed and are deployed as V2I by providing communication and computational capabilities. The RSU are denoted by $\mathcal{N}_r = \{r_1, r_2, ...r_m\}$ where m needs to be atmost 1 to perform multi-task offloading. The task request $\varrho = (x_i, y_i, I_n, \psi, \vartheta)$ where x_i and y_i denotes the coordinates of the vehicle, I_n denotes the input task size, ψ indicates the number of CPU cycles to execute, and ϑ denotes the maximum tolerable delay required for the task to execute. Vehicles communicate with RSU, edge controller, and cloud using wireless communication. Then, a particular edge controller in each region uses greedy offloading to decide where to offload based on the availability of resources in RSUs, BS, and the cloud. The non-orthogonal multiple access technique provides better transmission and avoids interference [23]. Since the vehicles are in constant motion, there is variability in their position. So, the nearest offloading node to the vehicle is determined using the *Euclidean distance* and *sojourn time* of the vehicle along with the task deadline. The node that satisfies can be used as an offloaded node to transmit data.The data transmission rate denoted by $\mathcal{R}$ is calculated using the Shannon capacity formula in Eq. (1)

$$\mathcal{R} = \mathbb{B} * log_2(1 + \frac{C_g * T_p}{N_p}) \tag{1}$$

where $\mathbb{B}$ is the bandwidth (Hz), T_p represents the transmission power of the vehicular task (W), C_g represents the channel gain of the signal, and N_p is the noise power (dBm/Hz).
The vehicular tasks are mobile; therefore, it is challenging to make the offloading decision compared to static offloading techniques. We find the *Euclidean distance* between proximal vehicular tasks and nearest BS, RSU and edge/cloud as $d_{(x,y)} = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}$ where, x_m, x_n, y_m and y_n are the x and y coordinates of the vehicular task and edge node. At the same time, we also compute the *sojourn time* as $\mathbb{S} = \frac{RSU_{range} - d_{(x,y)}}{velocity}$ the time to leave for the vehicle from one edge node to another. where, RSU_{range} determine the range of the RSU and $velocity$ represent the velocity of the vehicle [1]. This helps us determine whether a task can be successfully executed without interruption.

The total offloading delay is the time taken for uplink transmission and execution as in Eq. (2)

$$O_d = \frac{I_n}{\mathcal{R}} + \frac{I_n * \eta}{T_{remote}} \tag{2}$$

where η represents the input-output ratio and T_{remote} represents the computation power of BS, RSU, and cloud allocated for a particular task. In the case of local execution, the delay is only the execution of the task given by Eq. (3)

$$L_d = \frac{I_n * \eta}{T_{local}} \tag{3}$$

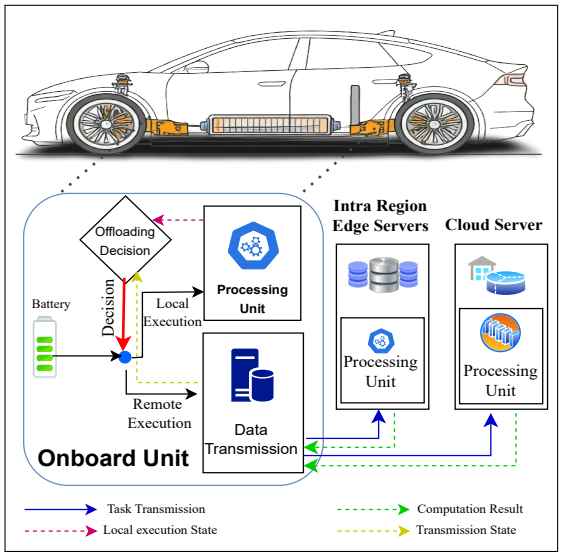


Fig. 2: Proposed architecture of deadline based multi-task offloading in collaborative vehicular-cloud network.

Local energy and offloaded energy are calculated in Eq. (4) and Eq. (5) as

$$E_l = \gamma * L_d \tag{4}$$

where γ represents the energy consumed by the vehicle per CPU revolution and L_d represents the local delay.

$$E_o = T_{remote} * O_d \tag{5}$$

where T_{remote} represents the computation power of the offloaded node(BS, RSU and cloud) and O_d represents the offloading delay.

III. Problem Formulation

For a given vehicular task set $\mathcal{T}_n$, the decision of offloading is taken based on the cost of execution of the task. The cost is calculated based on the weighted average of delay and energy. The delay and energy include the local execution and offloaded execution. The node with the minimum cost will be selected as the best node to perform execution. Then, based on the node offloading, a decision is taken whether to offload or do local execution. Let us consider a binary indicator variable $\pi(v_i, o_j)$, which denotes whether a given task is assigned to an offloaded node or not and is given by Eq. (6).

$$\pi(v_i, o_j) = \begin{cases} 1, & \text{if } v_i \text{ is assigned to } o_j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

$$\mathcal{C}_n = \min \sum_{t_{i=1}}^{n \in \mathcal{T}} (\alpha * max(L_d, O_d) + \beta * max(E_l, E_o)) \tag{7}$$

The objective function minimizes the cost using weighted coefficients α and β for considering the weighted average of latency and energy consumption subject to the following constraints (C1−C4):

C1: **Offloading decision constraint**: Each task must be either processed locally or offloaded to one server: $0 < \lambda_i < 1, \forall n \in \mathcal{T}$.

C2: **Processing Constraint**: The total computational demand (current and new tasks) must not exceed the computational capacity of edge node j: $\sum_{t_i \in \mathcal{T}_j} \psi_i + \psi_{new} \leq \mathcal{C}_j^{max}$

C3: **Latency constraint**: The total latency for the task, including transmission time and processing time, should not exceed the tolerable latency: ϑ: $max(L_d, O_d) \leq \vartheta$.

C4: **Energy constraint**: The energy consumption for offloading, including transmission energy and processing energy, should not exceed the maximum energy:E_{max}: $max(E_d, E_o \leq E_{max})$.

Algorithm 1: Vehicular task offloading with minimum cost

Input: T: Set of tasks, V: Set of vehicles, $\mathcal{N}_r$: Set of RSUs, $\mathcal{N}_b$: Set of Base Stations, $\mathcal{C}$: Cloud
Result: mincost

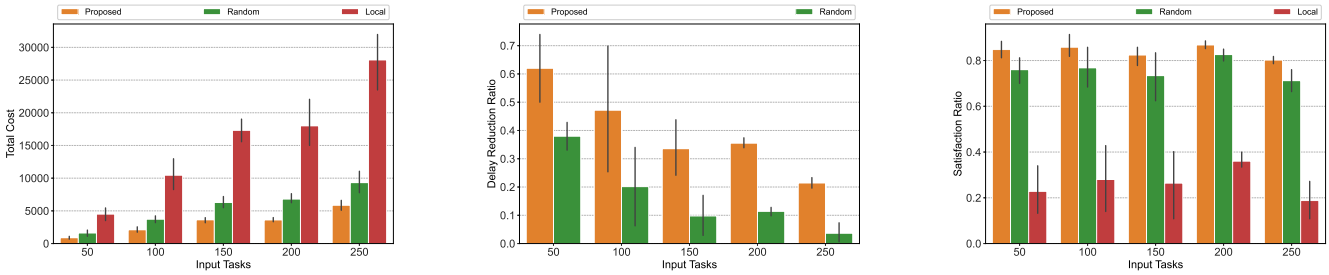
Initialize: $T_{\text{assigned}} = \emptyset$, $T_{\text{status}} = \{P\}$, cost $= 0$;
foreach $t_j \in T$ **do**
  if $(t_j(\textit{transmission time}) > \mathbb{S})$ **then**
    if $(t_j < \vartheta)$ **then**
      Perform local offloading;
      Calculate local delay and energy (Eq. (3) and Eq. (4));
  else
    bestnode ← None;
    foreach $n \in (\mathcal{N}_r \cup \mathcal{N}_b \cup \mathcal{C})$ **do**
      Calculate the cost of task execution using Eq. (7) ;
      if $(\textit{cost}(t_j, n) < \textit{cost}(t_j, \textit{bestnode})$ ***and*** $t_j < (\mathbb{S}, \vartheta))$ **then**
        bestnode ← n;
        mincost ← cost(t_j, n);
    Calculate offloading delay and energy (Eq. (5) and Eq. (2));
return mincost;

IV. Solution Approach

Our proposed algorithm (1) focuses on reducing the system cost and performing binary offloading in a mobility-based vehicular network involving four regions in a city. Initially, the Euclidean distance between the vehicular task and the proximal node, i.e. (RSU, BS, or cloud) positions are taken to find the nearest node in the particular region. The Edge controller in each base station maintains two important status variables $\mathcal{T}_{assigned}$ and $\mathcal{T}_{status}$ to monitor whether the task is assigned to the local or offloaded node. If the task is offloaded, the value is assigned to 1; otherwise, 0. If the resources are busy executing the tasks, they are in the M/M/1 queue. Since the transmission delay is greater than the queuing delay, it can be ignored when calculating the total delay. Initially, the task is 'Pending' until the edge controller makes the offloading decision. The selection of the best node involves the following steps. Initially, the cost of execution of the task is based on the current resources, and the node's computation power is calculated. In the case of local execution, communication delay is ignored.

V. Experimental Setup and Evaluation

The weighted average of latency and energy is compared with the given budget provided it satisfies the latency, communication, processing, and deadline constraints. When the



(a) Total cost *vs.* Input tasks (b) Delay reduction ratio *vs.* Input tasks (c) Satisfaction ratio *vs.* Input tasks

Fig. 3: Performance metrics for multi-task offloading in vehicular networks

resources are available, and the tasks are executed, the task status is updated to 'In Progress', and once the tasks are successfully executed, the task is updated to 'Success'. This process repeats until all the tasks are executed. If the task is interrupted due to a failure in network transmission or not satisfying deadline constraints, they are 'Terminated' from the queue. Since it is a multi-task offloading, we have used *'asyncio'* and *'multiprocessing'* function calls. We consider each task a thread to execute, sharing the resources at the edge nodes. Finally, once all the task execution is completed, irrespective of whether it has been done locally or offloading, we calculate the task execution cost.

We consider a *2-lane road* network with vehicles traversing the area with a coverage of 4 km, divided into four regions. Each region will have almost one base station and at least 1 RSU. The cloud server is located at a remote location of 200 km. The input task size considered is [1,10] MB, which requires 1 billion cycles per MB for processing. The computational frequency of the vehicle is [1-2] GHz, and of BS, RSU is 20 GHz and 10 GHz shared among the vehicles [24]. The cloud server has a computational power of 10 GHz per task. The transmission power of the vehicle task is 2 W. The network bandwidth varies from [5-10] MHz. The deadline for the task is [5-20] seconds. The vehicles move at a velocity ranging from 40 km to 90 km. The input-output ratio is between 0.01 to 0.5. The entire simulation is performed using Python IDE. The proposed algorithm is compared with other baseline algorithms, such as *random offloading* and *local execution*. In *random offloading*, the tasks are randomly allocated based on the resources available in the four regions. No constraint regarding resources available or latency constraints is taken into consideration, whereas in *local offloading*, all the tasks are executed in the onboard unit. We compare the proposed algorithm with the following evaluation metrics.

- **Total cost**: Fig. 3a compares the total system cost of the vehicular tasks with an increase in input tasks. The greedy offloading decision in the proposed work always tries to allocate to the best optimal node, decreasing the system cost by around 60.44%. The offloading decision is based on the computing capabilities of the task along with the deadline and sojourn time. In the case of *local execution*, low computation power takes more time, which increases the cost, and in *random offloading*, the cost varies since it is allocated randomly.
- **Delay reduction ratio**: Fig. 3b visualizes the delay reduction ratio with increased tasks. The delay reduction ratio is the total delay reduction to the total local execution. As the size of the tasks is less, the server or edge device is underutilized, allowing quick processing of offloaded tasks. This results in a significant reduction in delay of around 53.43% compared to local execution. However, there is a slight decrease in the ratio as the task demand increases along with network transmission and server processing delays. In case of *random offloading* it is difficult to predict the delay reduction ratio due to its unpredictable behavior in allocation of resources.
- **Satisfaction Ratio**: Fig. 3c provides the satisfaction ratio where the user's performance requirements are met according to the given constraints in the objective function. It is measured as the total number of tasks completed during the execution to the total number of tasks participated in the offloading process. In the case of *random offloading*, some tasks might get successfully executed due to their random behavior. In contrast, in *local offloading*, only such tasks are executed for the tasks that can satisfy OBUs' computational power.

VI. Conclusion and Future Work

The increasing demands of latency-sensitive applications in vehicular networks highlight the critical need for fast, reliable task execution amidst challenges like limited onboard resources and high mobility. To address this, our work focuses on optimize node selection using a greedy-based offloading strategy to provide cost-efficient multi-task offloading by leveraging the edge server. The numerical finding reveals that the proposed algorithm reduces the total system cost, delay reduction ratio by 60.44%, 53.43% and improves the satisfaction ratio by 36% to minimize the total system cost and provide better QoS and QoE.

In the future, we aim to extend our work for load balancing to provide efficient adaptive and scalable solutions for next-generation vehicular systems.

REFERENCES

[1] J. B. D. da Costa, A. M. de Souza, R. I. Meneguette, E. Cerqueira, D. Rosário, C. Sommer, and L. Villas, "Mobility and deadline-aware task scheduling mechanism for vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 10, pp. 11 345–11 359, 2023.

[2] F. B. Insights, "Autonomous vehicle market size, share and covid-19 impact analysis, by level (l1, l2, l3,l4 and l5), by vehicle type (passenger cars and commercial vehicles), and regional forecast, 2023– 2030," 2024. [Online]. Available: https://www.fortunebusinessinsights.com/autonomous-vehicle-market-109045

[3] A. Waxman, J. LeMoigne, L. Davis, B. Srinivasan, T. Kushner, E. Liang, and T. Siddalingaiah, "A visual navigation system for autonomous land vehicles," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 124–141, 1987.

[4] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[5] C. You, J. Lu, D. Filev, and P. Tsiotras, "Autonomous planning and control for intelligent vehicles in traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2339–2349, June 2020.

[6] P. Soni, O. C. Deshlahre, A. Satpathy, and S. K. Addya, "Lba: Matching theory based latency-sensitive binary offloading in iot-fog networks," in *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2024, pp. 165–170.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[8] X. Gu and G. Zhang, "Energy-efficient computation offloading for vehicular edge computing networks," *Computer Communications*, vol. 166, pp. 244–253, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420320168

[9] A. Yoganandhan, S. Subhash, J. H. Jothi, and V. Mohanavel, "Fundamentals and development of self-driving cars," *Materials today: proceedings*, vol. 33, pp. 3303–3310, 2020.

[10] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, thirdquarter 2017.

[11] S. Wang, D. He, M. Yang, and L. Duo, "Cost-aware task offloading in vehicular edge computing: A stackelberg game approach," *Vehicular Communications*, vol. 49, p. 100807, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209624000822

[12] M. S. Bute, P. Fan, L. Zhang, and F. Abbas, "An efficient distributed task offloading scheme for vehicular edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 149–13 161, 2021.

[13] W. Fan, Y. Su, J. Liu, S. Li, W. Huang, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for vehicular edge computing based on v2i and v2v modes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4277–4292, 2023.

[14] S. O. Ogundoyin and I. A. Kamil, "Optimal fog node selection based on hybrid particle swarm optimization and firefly algorithm in dynamic fog computing services," *Engineering Applications of Artificial Intelligence*, vol. 121, p. 105998, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623001823

[15] S. Zhang, N. Yi, and Y. Ma, "A survey of computation offloading with task types," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[16] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint optimization of computation offloading and task scheduling in vehicular edge computing networks," *IEEE Access*, vol. 8, pp. 10 466–10 477, 2020.

[17] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: A matching-theoretic framework," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 100–106, 2019.

[18] S. Abuthahir and S. P. Peter, "Tasks offloading in vehicular edge computing network using meta-heuristic algorithms - a study of selected algorithms," in *15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–10.

[19] K. Tan, L. Feng, G. Dán, and M. Törngren, "Decentralized convex optimization for joint task offloading and resource allocation of vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 12, pp. 13 226–13 241, 2022.

[20] Z. Liu, X. Wang, D. Wang, Y. Lan, and J. Hou, "Mobility-aware task offloading and migration schemes in scns with mobile edge computing," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

[21] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for iov using 5g nr-v2x communication," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10 397–10 410, 2022.

[22] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2897–2909, 2022.

[23] D. Zheng, Y. Chen, L. Wei, B. Jiao, and L. Hanzo, "Dynamic noma-based computation offloading in vehicular platoons," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 10, pp. 13 000–13 010, 2023.

[24] Q. Jiang, X. Xu, M. Bilal, J. Crowcroft, Q. Liu, W. Dou, and J. Jiang, "Potential game based distributed iov service offloading with graph attention networks in mobile edge computing," *IEEE Transactions on Intelligent Transportation Systems*, 2024.