Efficient Multi-User Resource Allocation for Urban Vehicular Edge Computing: A Hybrid Architecture Matching Approach

Hongyang Xie[#], Haoqiang Liu[*,#], Huiming Chen, Shaohan Feng, *Member, IEEE*, Zhaobin Wei, Yonghong Zeng, *Fellow, IEEE*

***Abstract*—Advanced in the proliferation of the Internet of Things (IoT), a plethora of functions have been integrated in vehicular networks and thereby transfered it into a smart network. However, the contradiction between the limited on-vehicle computing resource and the massive data collected by these IoT devices hinders the broader adoption of vehicular network as a vast variety of on-vehicle applications are latency-sensitive. To address this issue, vehicular edge computing has become a promising technology as it can offload a large number of tasks from its proximal vehicles. However, the offloading methods recently utilized are inefficient while dealing with multi-user vehicular networks under dynamic scenarios. To design a superior offloading method that can effectively and efficiently offload tasks from vehicles to servers, multiple objectives and constraints with various topologies should be considered. In this paper, instead of constructing a typical multi-user and multi-server vehicular edge computing scenario, a complex scenario with more uncertainties, i.e. urban scenario, is modeled. We propose a Hybrid Architecture Matching Algorithm (HAMA) to minimize the average time latency subject to the constraint on energy consumption and evaluate the proposed algorithm in the above two scenarios. Moreover, HAMA is constructed based on hybrid centralized-distributed architecture, which can process the centralized collected information on a distributed manner. Experimental results demonstrate that the matching algorithm can significantly reduce average time latency, achieving up to a 68% improvement compared to local execution.**

***Index Terms*—Vehicular edge computing, matching algorithm, resource allocation, optimization algorithm**

I. Introduction

*The corresponding author is Haoqiang Liu. # indicates equal contribution. This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research & Development Programme (FCP), and was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62302446 and in part by the Provincial Universities Basic Research Project (Science and Technology) under Grant QRK23011.

Hongyang Xie is with School of Artificial Intelligence, Southeast University, Nanjing, China.

Haoqiang Liu is with Department of Electronic Engineering, Tsinghua University, Beijing, China. Email: lhqbuaa@mail.tsinghua.edu.cn

Huiming Chen is with School of intelligence Science and Technology, University of Science and Technology Beijing, Beijing, China.

Shaohan Feng is with School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China.

Zhaobin Wei is with the College of Electrical Engineering, Sichuan University, Chengdu, China. Email: xiaobiner@stu.scu.edu.cn

Yonghong Zeng is with Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore.

WITH the popularization of 5G and the increase of connected vehicles in intelligent transportation systems, intelligent fields such as autonomous driving, facial recognition, and augmented reality driving have emerged that require a huge amount of computing resources for applications that require real-time processing. This poses a huge challenge to the vehicles with restricted batteries and computing resources.

To deal with this matter, Mobile Cloud Computing (MCC) was proposed enabling task offloading to remote cloud servers that are equipped with sufficient computation resources. However, due to the distant cloud server, MCC does not show benefit regarding the delay. Mobile edge computing (MEC) is introduced to solve this problem, where the MEC server is located at the edge of the wireless access network [1]. By offloading to the MEC server, fast response is envisioned for the vehicles. Nevertheless, MEC servers are often subject to IT resource constraints and communication overhead. Worsely, the dynamic and uncertain vehicular network poses additional challenge to the task offloading to MEC servers. Therefore, for MEC based embedded networks, it is crucial to effectively allocate communication and computing resources to ensure quality of service (QoS).

In the on-board edge computing network, one of the most challenging problems is to obtain the best task offloading decision when the aim is to minimize the total processing time of computation-intensive tasks generated by multiple vehicles. Such an aim becomes much more challenging considering guaranteeing fairness among the vehicles by conducting joint optimization of intensive computation, delay, and energy consumption. To meet the above requirements, effectiveness and adaptability of the strategy need to be taken into account during the algorithm design and performance evaluation. Therefore this study devises an efficient offloading approach, introducing the Hybrid Architecture Matching Algorithm (HAMA), to minimize average time latency while adhering to energy consumption constraints in a complex urban vehicular scenario.

The main contributions of this paper are as follows:

- The urban city road model incorporates characteristics specific to urban environments, utilizing Vehicle-to-Vehicle (*V2V*) and Vehicle-to-Infrastructure (*V2I*) topologies to create a realistic test environment for the vehicular network model.
- The average time latency for multiple vehicles in a dynamic vehicular network is minimized, accounting for

five distinct components of delay computation.
- In the proposed HAMA, key parameters for the preference list are selected based on varying objectives to maximize individual profits.
- For HAMA, a centralized coordinator gathers global information from multi-server and multi-agent vehicular networks, while the interaction process is designed to facilitate distributed computation.

The remaining parts are organized as follows. Section II introduces related works. Section III introduces the system model and problem formation. Section IV introduces the proposed algorithm. Section V provides an evaluation of algorithms in complex scenarios. Finally, Section VI concludes the paper.

II. Related Work

The existing literature has proposed numerous methods for analyzing the tasks generated in vehicular networks. To summarize these efforts, there are two primary areas of exploration: offloading policies and characteristics of vehicular networks.

A. Offloading Policies

Distributed methods enable efficient agent interaction without centralized control; recommended for migration decisions, comparing centralized and distributed architectures based on power consumption and latency [2], [3]. Distributed algorithms outperform centralized ones; for fairness among users, a low average delay is achieved through distributed game theory, balancing users and servers via offloading decisions [4].

B. Vehicular Network Characteristics

1) Mobility: Vehicle edge computing, primarily for real-time interactive scenarios, often overlooks vehicle dynamic motion, leading to static offloading decisions [5], [6]. However, recent research introduces practical approaches considering time-varying wireless channels and dynamic environments, employing learning-based algorithms to minimize average delay [3], [7].

2) Delay-sensitive and Intensive computation: The vehicle network processes real-time data collected by high computational complexity applications. Due to the direct impact of data processing on subsequent operations, as well as the development of 5G and the Internet of Things, frequent information exchange and large amounts of data require lower response times as computing intensity increases.

III. System Model

A. Overview

This model incorporates two modes of data transmission through *V2I* and *V2V* topologies in Vehicle-to-everything (*V2X*). For each task carrier a random data size is generated for each interval, and the task carrier determines how to accomplish the mission based on whether it is offloaded to the MEC server, server carrier, or local execution.

1) Scenario Assumptions: Fig. 1 illustrates the road scenario with distinct urban city characteristics clearly combined. In comparison to the highway scenario, which is a unidirectional straight road with multiple traffic streams mentioned in [3] and [2], the urban city road scenario is more practical and closer to the real world. The urban road model specifically makes the ensuing postulates:

Firstly, time is equally divided into discrete slots. Vehicle speeds are randomized independently at the beginning of each time slot τ, with no correlation among the speeds of vehicles.

Secondly, vehicles in urban areas are distributed according to a Poisson process, while their speeds follow a Gaussian distribution, denoted as $f(v) \sim N(\mu, \phi^2)$. The speed of vehicles can be zero due to traffic lights, introducing additional fluctuations and uncertainties.

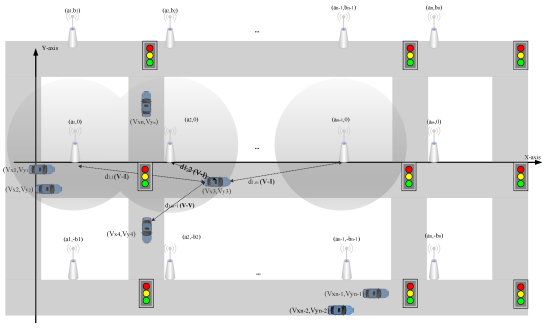


Fig. 1. The system model in urban city road scenario

Thirdly, there are turning directions (left, right or straight) at intersections, and the frequency of handover delay caused by them is higher than that in highway scenarios. Specifically, vehicles may exit the communication coverage of a server $(server-j)$ and subsequently enter the coverage of a different server $(server-j')$, increasing the frequency of handovers.

Fourthly, higher traffic density reduces the distances between vehicles, leading to an increase in task generation with greater computational demands.

Fifthly, urban road transportation requires stringent delay tolerance constraints due to the need for prompt and responsive handling of real-time data.

2) Mathematical Representation of the Model: For the presentation of the dynamic model, we choose to utilize a two-dimensional Cartesian coordinate system [7]. And the initial speed of the i-th vehicle is presumed to be random and the j-th road side unit (RSU) is located alongside that street at $(a_j, 0)$. While the i-th vehicle is changing location at (x_i, y_i), we can express the time-varying distance from this vehicle to the RSU as:

$$d_i(\tau) = \sqrt{(x_i - a_j)^2 + y_i^2}. \tag{1}$$

For the wireless channel, it is assumed that the transmission rate is primarily influenced by the time-varying distance $d_i(\tau)$ between the MEC server connected to the RSU and the vehicle [7]. And after using Shannon formula, we have transmission power p, channel bandwidth ω, channel power gain $G(t)$, and noise power σ^2 and then we can calculate the uplink transmission rate as:

$$r_i(\tau) = \omega \log_2 (1 + \frac{pG_i(\tau)}{\sigma^2}). \tag{2}$$

While the channel power gain $G(t)$ can be calculated by

$$G_i(\tau) = \rho_0 d_i(\tau)^{-\theta} = \frac{\rho_0}{(y_i^2 + (a_j - x_i)^2)^{\frac{\theta}{2}}}, \tag{3}$$

where θ is the path-loss exponent and ρ_0 is the channel power gain at reference distance $d = 1$. The vehicles adopt orthogonal channels to transmit data, resulting in no interference among the vehicles.

To simplify the model, we assume that each task vehicle generates only one task per time slot and this task can be completed within that single time slot. Moreover, there is no task can be partially offloaded, indicating that a task vehicle can select only a MEC server or a server vehicle to accomplish the task at most. However, handover issues may arise, due to the fact that vehicles may move within a single time slot [5].

The city road is equipped with *N* RSUs, each of which has a MEC server covering a specific communication range. And the computation capacities of these MEC servers can be denoted by $\{f_1, f_2, \ldots, f_N\}$. The vehicles' number is random, of which *n* are server vehicles, which are randomly selected with sufficient computing resources at each time slot, and *M* are the task vehicles. And the vehicle servers' computation capacities can be denoted by $\{f_{N+1}, f_{N+2}, \ldots, f_{N+n}\}$. It is important to note that *M* and *n* are not the same values in each time slot during the movement process. *V2I* or *V2V* edge computing services can be provided by MEC servers to vehicles within the maximum range *R* to offload task assignments. Therefore, it has $N + n + 1$ feasible choices at each time slot for each of the task vehicles [4]. The task offloading decisions of $task\,vehicle\,i$ is $s_i \in \{0, 1, 2, \ldots, N, N+1, N+2, \ldots, N+n\}$, where $s_i = 0$ signifies that the task vehicle opts to execute operations locally, $s_i = j\ (1 \le j \le N)$ signifies the task vehicle chooses to offload its task to MEC server of $RSU\,j$, and $s_i = j\ (N+1 \le j \le N+n)$ signifies the task vehicle chooses to offload its task to $server\,vehicle\,j$. And the sections below delve into the computations of various models and the examination of competitive dynamics.

3) Local Execution: The time for processing will only be related to CPU cycles intensity to complete per bit $\overline{\omega}$, the data size of the task d_{up} and computation capability of vehicle CPU f^{Local}, if vehicle chooses to finish a task locally $(s_i = 0)$ [4]:

$$t_i^{\mathrm{Local}}(\tau) = \frac{d_{\mathrm{up}}\overline{\omega}}{f^{\mathrm{Local}}}. \tag{4}$$

The local execution' energy consumption $e_i^{\mathrm{Local}}(\tau)$ is known based on [7], and is related to cycles intensity $\overline{\omega}$, data size d_{up} and energy consumption of each CPU cycle $\delta_i^{\mathrm{Local}}$:

$$e_i^{\mathrm{Local}}(\tau) = \delta_i^{\mathrm{Local}} d_{\mathrm{up}}\overline{\omega}. \tag{5}$$

4) Offloading to MEC servers connected to RSUs: If $vehicle\,i$ chooses that the task can be offloaded to one of the MEC servers $(s_i = j\ (1 \le j \le N))$, the computation task's processing time is composed of four stages [5] and [3]: the data upload transmission delay $t_{i,j}^{\mathrm{up}}(\tau)$ from task vehicle to server, the execution delay $t_{i,j}^{\mathrm{com}}(\tau)$ on the server, the data download transmission delay $t_{i,j}^{\mathrm{dow}}(\tau)$ from the MEC server back to the vehicle, the queuing delay $t_{i,j}^{\mathrm{que}}(\tau)$ for waiting other task execution and the handover delay $t_{i,j}^{\mathrm{hand}}(\tau)$. In such case, the sum of delay $t_{sum}^{\mathrm{MEC}}(\tau)$ of offloading the task to $RSU\,j$ at time slot τ is calculated as:

$$t_{sum}^{\mathrm{MEC}}(\tau) = t_{i,j}^{\mathrm{up}}(\tau) + t_{i,j}^{\mathrm{com}}(\tau) + t_{i,j}^{\mathrm{dow}}(\tau) + t_{i,j}^{\mathrm{que}}(\tau) + t_{i,j}^{\mathrm{hand}}(\tau). \tag{6}$$

The energy consumption of a task vehicle during this process $t_{i,j}^{\mathrm{MEC}}(\tau)$ is proportional to the transmission power required for uploading and downloading [7]:

$$e_{i,j}^{\mathrm{MEC}}(\tau) = p \cdot (t_{i,j}^{\mathrm{up}}(\tau) + t_{i,j}^{\mathrm{dow}}(\tau)). \tag{7}$$

a) Task upload transmission delay: We can calculate the first item of (6) as task upload from $vehicle\,i$ to $RSU\,j$ via *V2I* communications [5], and the upload transmission delay $t_{i,j}^{\mathrm{up}}(\tau)$ of uploading the task is:

$$t_{i,j}^{\mathrm{up}}(\tau) = \frac{d_{\mathrm{up}}}{r_i^{\mathrm{up}}(\tau)}, \tag{8}$$

where $t_{i,j}^{\mathrm{up}}(\tau)$ is calculated by uplink transmission rate as (2).

b) Task execution delay: The second delay that offloads the task MEC server connected to $RSU\,j$ have a connection with the computation capability f_j^{MEC} and cycles intensity $\overline{\omega}$ [5]:

$$t_{i,j}^{\mathrm{com}}(\tau) = \frac{d_{\mathrm{up}}\overline{\omega}_n}{f_j^{\mathrm{MEC}}}. \tag{9}$$

c) Handover delay: The handover delay is calculated as the computation result transmitted from $server-j$ to $server-j'$, and then forward to that task vehicle [3]. Assuming the amount of computational data is insignificant compared to the input required for the task, and the handover delay is primarily caused by backhaul delay:

$$t_{j,j'}^{\mathrm{hand}} = (j - j')c_t, \tag{10}$$

where c_t says the handover delay from $server-j$ to $server-(j+1)$, which is often regarded as a constant value.

5) Offloading to the server vehicles: If $vehicle\,i$ chooses to offload the task to one of server vehicles $(s_i = j\ (N+1 \le j \le N+n))$, the processing time are similar to the time offloading to MEC server [3]. On this condition, the sum of delay $t_{sum}^{Sev}(\tau)$ of offloading the task to $server\,vehicle\,j$ at time slot τ can be calculated as:

$$t_{sum}^{SeV}(\tau) = t_{i,j}^{\mathrm{up}}(\tau) + t_j^{com}(\tau) + t_{i,j}^{\mathrm{dow}}(\tau) + t_{i,j}^{\mathrm{que}}(\tau) + t_{i,j}^{\mathrm{hand}}(\tau). \tag{11}$$

This process also features similar energy consumption for the task vehicle $e_{i,j}^{\mathrm{SeV}}(\tau)$ [7] as offloading tasks to the MEC server:

$$e_{i,j}^{\mathrm{SeV}}(\tau) = p \cdot (t_{i,j}^{\mathrm{up}}(\tau) + t_{i,j}^{\mathrm{dow}}(\tau)). \tag{12}$$

According to the aforementioned system model, the processing time of a vehicle computation task within one time slot is as follows:

$$t_i(s_i) = \begin{cases} t_i^{\mathrm{Local}}, & if\ s_i = 0. \\ t_{i,j}^{\mathrm{MEC}}, & if\ s_i = j, 1 \le j \le N. \\ t_{i,j}^{SeV}, & if\ s_i = j, N+1 \le j \le N+n. \end{cases} \tag{13}$$

The above assumptions provide a reasonable approximation for analyzing task offloading and execution dynamics, and offer insights that are largely applicable to real-world scenarios.

B. Problem Formulation

By taking into account the computational information of vehicles, MEC servers, and environmental information, the objective is to minimize the average delay latency for multiple vehicles while ensuring satisfaction of the energy consumption constraint in each time slot. The optimized issue can be formulated as follows, which is close to [3]:

$$\begin{aligned} \min_{\{s_i\}} \quad & \frac{1}{M}\sum_{i=1}^{M} t_i(s_i) \\ \text{s.t.} \quad & \text{energy}(\tau) \leq B, \\ & \text{energy}(\tau) = e_i^{\text{Local}}(\tau)\cdot I_{\{s_i=0\}} + e_{i,j}^{\text{MEC}}(\tau)\cdot I_{\{s_i=j\}} \\ & \qquad + e_{i,j}^{\text{SeV}}(\tau)\cdot I_{\{s_i=j\}}, \quad \forall j \in M. \end{aligned} \tag{14}$$

The task execution decisions are $s_i \in S$, $S \triangleq \{s_1, s_2, \ldots, s_M\}$, and $I_{\{*\}}$ is an indicator, $I_{\{*\}} = 1$ while $*$ is true. B denotes energy consumption constraint in the system.

The model is designed to allocate resources among multiple vehicles and generate optimal offloading decisions at each time slot within the framework of a realistic vehicular network environment. By balancing the trade-off between time delay and energy consumption, this approach maximizes the benefits not only for consumers but also for network operators.

IV. Hybrid Architecture Matching Algorithm

A. Matching Algorithm design

The matching theory serves as a valuable tool for investigating the optimization of reciprocal relations, particularly in the context of task vehicles and MEC servers, which can be conceptualized as agents generating preference lists based on collected *V2I* and *V2V* information. This theoretical framework provides a more beneficial and flexible selection list for both agents.

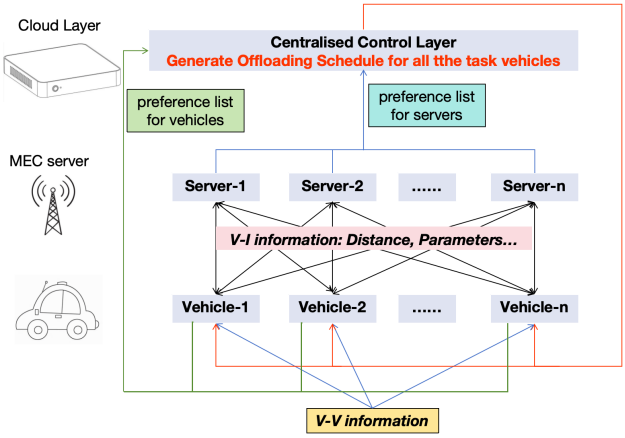


Fig. 2. Matching algorithm block graph

In our specific scenario, characterized by modest data volume and the utilization of optical fibers for transmission within and between servers and the cloud, the efficient and high-speed nature of optical fiber transmission mitigates the potential impact of centralization.

The stable marriage matching algorithm, initially designed for one-to-one matching with an equal number of agents [5], forms the basis for modeling the relationship between vehicles and servers or vehicles in the context of offloading tasks, achieving a balanced one-on-one matching. Extending beyond the one-to-one model, MEC servers or vehicles engage in multitasking facilitated by multitasking vehicles, resulting in a many-to-one matching scenario [8]. In this configuration, each vehicle can select at most one server, while a server can accommodate multiple tasks (quota $Q \geq 1$). For simplicity, we assume a lack of dependency relationships, thereby categorizing the model as typical matching [9]. This implies that the preference list of vehicles is solely influenced by the vehicle they aim to match and is not impacted by other dynamic formations, although the preference list undergoes changes over time.

Considering the centralized matching algorithm's system model block diagram illustrated in Fig. 2 above, this paper proposes the HAMA. It effectively divides the optimization problem into two sub-algorithms, enabling each task vehicle to be matched to a server while ensuring that no vehicle has motivation to transfer for negotiation. This approach proves beneficial to all agents involved in the offloading process.

Algorithm 1 The Hybrid Architecture Matching Algorithm with hybrid centralized–distributed architecture

Require: task vehicles' parameters; servers' parameters;
Ensure: task offloading decision $S = \{s_1, s_2, \ldots, s_M\}$ for task vehicles;
Initialize: offloading decision: execute locally for all task vehicles $S = \{0, 0, \ldots, 0\}$
Each vehicle and server generate its preference list based on all the information from centralised layer and locally collected
while any task vehicle is unmatched **do**
  for each vehicle **do**
    Each vehicle proposes to the most preferred server in its preference list;
  end for
  for each server **do**
    if $server_j$ receives only one proposal **then**
      Match the $vehicle_i$ with $server_j$ directly;
    else if $server_j$ receives more than one proposal
      Calculate the $server_j$ capacity and compare with quota;
      if current capacity is larger than quota **then**
        Rank the $vehicle_i$ and compare with existing vehicles
        if $vehicle_i$ has higher rank **then**
          Add $vehicle_i$ to the $server_j$ and remove the lowest rank vehicle out of the $server_j$
        else
          The $vehicle_i$ remains unmatched
        end if
      else if current capacity is larger than quota **then**
        Match the $vehicle_i$ with $server_j$ directly;
      end if
    end if
  end for
end while

B. Hybrid centralised-distributed architecture

Two sub-algorithms are presented to address the dynamic nature of vehicular networks, with updated frequencies employed at each time slot, facilitating a comprehensive analysis of the process.

1) Preference list generation: In the first sub-algorithm, agents update preference lists based on information, aiming

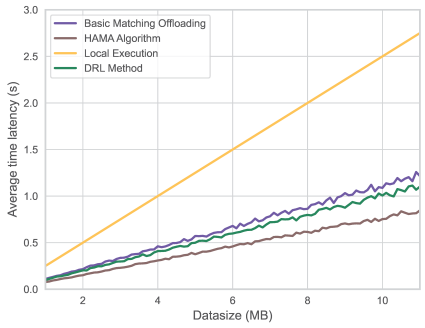


Fig. 3. Data size (MB) versus Average time latency (s)

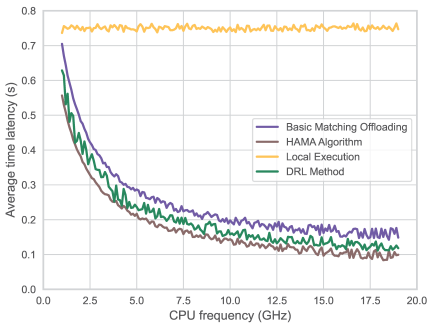


Fig. 4. Server CPU cycle frequency versus Average time latency

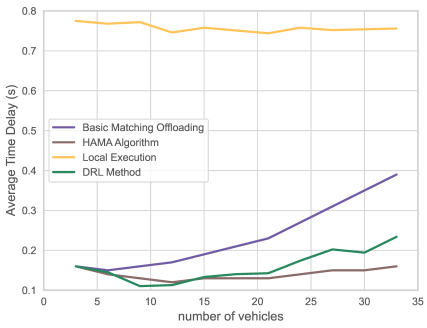


Fig. 5. Running time versus number of vehicles

to minimize processing time, energy consumption, and transmission delays while maximizing profitability. This involves broadcasting vehicle information, calculating transmission delays, and finalizing preference lists for both parties.

2) Matching game with guarantees: In dynamic edge computing networks, a self-learning approach is employed to achieve lower running time and computational complexity [2]. Task vehicle location information collected by the cloud layer is crucial for addressing handover challenges, transmitted to target servers for matching using Algorithm 1.

Task vehicles update preference lists and propose tasks to the most selected servers. Rejected tasks are offered to the next selected server, prioritized based on preferences. The algorithm terminates when all tasks are matched, minimizing delay and benefiting agents. In distributed processing, frequent communication introduces delays, while centralized methods entail greater transmission delays. Optimizing the hybrid architecture through matching theory offers high scalability without centralized coordination.

V. Performance Evaluations

To validate the proposed HAMA in urban scenarios, simulations replicate a system model with a 3 MB input data size randomly chosen, involving 21 vehicles (two designated as servers during each time slot) and 30 evenly distributed RSUs along city roads. CPU cycles for completion are set at 200 cycles/bit, with RSUs and server vehicle CPU cycle frequencies at 6.0 GHz, 5.0 GHz, and 0.8 GHz. MEC server and server vehicle quotas are fixed at 12 MB, and background noise power is -140 dBm, using a constant 25 MHz wireless transmission bandwidth. Reference channel power ρ and Pathloss exponent θ are configured as -30 dBm and 3, respectively.

However, addressing the amendment challenging the assumption of no interference among vehicles communicating with RSUs is crucial. The assertion of no interference faces practical difficulties due to various factors, including the impracticality of allocating independent 25 MHz channels to each vehicle. The use of a 25 MHz bandwidth in simulations poses challenges in achieving fixed allocations. Dynamic channel allocation is considered as a potential solution, introducing non-trivial challenges related to channel assignment tasks, involving intricate operations associated with multi-carrier Orthogonal Frequency Division Multiplexing (OFDM), where users share subcarriers.

In different settings, the proposed HAMA has been compared with existing baseline method, i.e. the one-to-one matching algorithm [10], a DRL method [9] and local execution. The one-to-one matching algorithm is basic matching algorithm which is derived from stable marriage problem.

A. Impact of data size

Fig. 3 shows the variation of average latency with increasing data size. The latency of three algorithms increases with the increase of data volume, but HAMA performs better than the others, reducing the time by more than half. The reason is that the increase in data size leads to more tasks being offloaded to MEC servers and vehicles, achieving lower average latency. The latency of the proposed HAMA rises gradually as the size of the task data increases, indicating that the matching algorithm can ensure the correspondence between the task vehicle and the server.

B. The impact of server CPU cycle frequency

The increase in average time delay is shown in Fig. 4. Computing power is specified as frequency, and offloading decisions are closely related to computing power. It is more likely to offload to servers or carriers to reduce latency. The average latency of the two algorithms is reduced, and HAMA is better than the basic algorithm. The rate of time delay reduction gradually decreases due to the impact of computing power on task offloading decisions, which will be alleviated when sufficiently large.

C. Impact of number of vehicles (traffic intensity)

Fig. 5 illustrates an increase in delay correlating with higher quantities, as more vehicles require offloading and processing tasks. Limited resources contribute to heightened processing time, encompassing transmission, computation, and queuing delays. Notably, the HAMA exhibits a slower delay escalation compared to the basic algorithm, with an average increase of only 0.05 seconds. Thus, it is well-suited for scenarios with heavy vehicle volumes, exceeding 100 vehicles per kilometer in traffic intensity.

From the above comparisons and analysis, in urban road scenarios, the proposed HAMA addresses significant processing delays, showcasing its effectiveness in complex networks and dynamic environments. The theoretical advantages of HAMA include its ability to handle many-to-one matching efficiently, reduced computational complexity compared to DRL

methods, and improved latency over local execution strategies. HAMA's design leverages a hybrid centralized-distributed approach, which combines the strengths of both centralized coordination and distributed computation to achieve optimal task offloading decisions.

VI. Conclusion

In this paper, with resource allocation, a task offloading strategy in vehicular edge computing was proposed and optimized. A typical road scene vehicle network model is constructed based on contributions from both *V2V* and *V2I* topologies. We present a novel algorithm called HAMA, and empirically demonstrate its effectiveness through rigorous evaluation involving key parameters. The performance evaluation is expanded to a more intricate scenario, encompassing a multitude of urban road characteristics, surpassing the scope of existing solutions. There are numerous intersections to choose from, a significant task load to execute, and uncertainty regarding the traffic light system's impact. Experimental findings demonstrate that the proposed algorithm is effectively adaptable. The matching algorithm performs better than the greedy algorithm because of the mutually beneficial relationships among agents with preference lists, achieving a reduction in average time delay of up to 68% compared to local execution. For future research, centralized cloud scheduling risks signaling overhead and delays, prompting the need for decentralized or edge-based scheduling mechanisms to enhance overall performance. This raises concerns about the efficiency and responsiveness of the system, warranting further investigation into decentralized or edge-based scheduling mechanisms to mitigate the identified challenges and enhance the overall performance.

References

[1] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Cached model-as-a-resource: Provisioning large language model agents for edge intelligence in space-air-ground integrated networks," *arXiv preprint arXiv:2403.05826*, 2024.

[2] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE Transactions on Vehicular Technology*, pp. 5087–5099, May 2019.

[3] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, pp. 2041–2057, Feb. 2022.

[4] H. Liu, W. Huang, D. I. Kim, S. Sun, Y. Zeng, and S. Feng, "Towards efficient task offloading with dependency guarantees in vehicular edge networks through distributed deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, 2024.

[5] Y. Dong, S. Guo, Q. Wang, S. Yu, and Y. Yang, "Content caching-enhanced computation offloading in mobile edge service networks," *IEEE Transactions on Vehicular Technology*, pp. 872–886, Jan. 2022.

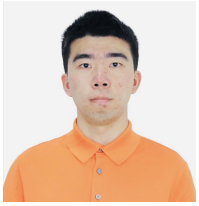
[6] J. Kang, J. Wen, D. Ye, B. Lai, T. Wu, Z. Xiong, J. Nie, D. Niyato, Y. Zhang, and S. Xie, "Blockchain-empowered federated learning for healthcare metaverses: User-centric incentive mechanism with optimal data freshness," *IEEE Transactions on Cognitive Communications and Networking*, 2023.

[7] B. Ko, K. Liu, S. H. Son, and K. J. Park, "Rsu-assisted adaptive scheduling for vehicle-to-vehicle data sharing in bidirectional road scenarios," *IEEE Transactions on Intelligent Transportation Systems*, pp. 977–989, Feb. 2021.

[8] A. Arfaoui, A. Kribeche, and S. M. Senouci, "Cooperative mimo for adaptive physical layer security in wban," *IEEE International Conference on Communications (ICC)*, 2020.

[9] M. Z. Alam and A. Jamalipour, "Multi-agent drl-based hungarian algorithm (madrlha) for task offloading in multi-access edge computing internet of vehicles (iovs)," *IEEE Transactions on Wireless Communications*, pp. 7641–7652, Sept. 2022.

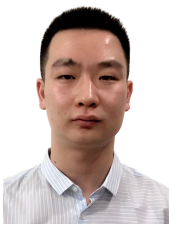
[10] X. Wang, J. Wang, X. Zhang, X. Chen, and P. Zhou, "Joint task offloading and payment determination for mobile edge computing: A stable matching based approach," *IEEE Transactions on Vehicular Technology*, pp. 12 148–12 161, Oct. 2020.



Hongyang Xie is an undergraduate researcher in the School of Artificial Intelligence, Southeast University, Nanjing, China. He was a research intern at the Department of Electronic Engineering, Tsinghua University in the summer of 2023. His currently research interests are in the area of stochastic optimization and resource allocation in the wireless networks.

Haoqiang Liu is a postdoctoral researcher in the Department of Electronic Engineering, Tsinghua University, Beijing, China. He received Master's degree and Ph.D. degree in communication and information system from Beihang University in 2017 and 2022. He is currently doing research on mobile edge computing, computation offloading, wireless network optimization, and reinforcement learning.

Huiming Chen is currently a post-doctoral research fellow at Department of Computer Science, City University of Hong Kong, and an associate professor with the School of intelligence Science and Technology, University of Science and Technology Beijing (USTB). He received the Ph.D degree in the Department of Electrical and Electronic Engineering, the University of Hong Kong (HKU) in 2020. His current research interests mainly lies in data-driven complex network analysis, knowledge graph with applications in urban computing.



Shaohan Feng received the B.S. degree from the School of Mathematics and Systems Science, Beihang University, Beijing, China, in 2014, the M.S. degree from the School of Mathematical Sciences, Zhejiang University, Hangzhou, China, in 2016, and the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2020. His current research interests include resource management and risk management in computer networks and wireless communications.

Zhaobin Wei is a Ph.D student at the Department of Electrical Engineering, Sichuan University, Chengdu, China. He received the M.S. degree in electrical engineering from China Three Gorges University, Yichang, China, in 2021. He is currently doing research on computation offloading, wireless network optimization, and reinforcement learning.

Yonghong Zeng received the B.S. degree from Peking University, M.S. degree and Ph.D. degree from National University of Defense Technology, China. He has been working in the Institute for Infocomm Research, A*STAR, since 2004, and currently is a senior principal scientist and group leader. He is working on integrated sensing and communication, B5G/6G communications, ultra-reliable low-latency communication, vehicular radar and communication, and real time localization system.