Membrane Technology
ISSN (online): 1873-4049

Task Offloading and Collaborative Backhaul System Based on Multi-level Edge Computing in the Internet of Vehicles

Jin Lin, Zeqin Li, Ruofei Wang*, Ruyue Gong, Hongjing Wu
School of Computers Science, Neusoft Institute Guangdong, Foshan, China
**Corresponding Author.*

Abstract:
With the development of 5G and the Internet of Vehicles, diverse in-vehicle services continue to emerge. Computation-intensive and delay-sensitive in-vehicle tasks pose significant challenges to in-vehicle devices and represent one of the bottlenecks limiting the development of Internet of Vehicles technology. This paper proposes a Speed-Sensitive Offloading (SSO) and collaborative backhaul solution to address the problem of task offloading and result backhaul failure caused by vehicle movement, including a multi-level MEC architecture solution, speed sensitive task offloading and an MEC collaboration-based task return scheme (SSCOM). Through preliminary experimental verification, as the number of vehicles increases, the average task offloading time of all schemes shows an upward trend, but the SSCOM scheme has the smallest increase; compared with the schemes of random offloading, speed-prioritized and data-volume-prioritized offloading, the present scheme can significantly reduce the average task offloading time; collaborative backhaul can also solve the problem of result backhaul failure caused by vehicles driving out of the coverage area, etc., can improve the task backhaul success rate and MEC resource utilization rate by at least 5%.

Keywords: vehicle, offloading, backhaul, velocity sensing, MEC

INTRODUCTION

Currently, 5G and vehicular networking technologies are rapidly advancing, enabling mobile vehicles to interconnect with networks, artificial intelligence, and other intelligent terminal devices. At the same time, various in-vehicle services continue to emerge, such as online navigation, road condition recognition, audio-visual entertainment, and autonomous driving [1,2]. Although 5G communication technology has improved data transmission speeds, the high mobility and dynamism of vehicles pose significant challenges to in-vehicle devices for computationally intensive, latency-sensitive tasks such as task offloading and collaborative backhaul [3]. An effective solution is to offload computational tasks to cloud servers [4]. However, for delay-sensitive tasks, the delay caused by inter-cloud transfers is almost unacceptable. For example, the delay requirement for task response in autonomous driving is only 5-10ms [5]. In addition, a large number of in-vehicle tasks sending data to the cloud at the same time can also put a huge pressure on the bandwidth of the core network [6].

The emergence of Multi-Access Edge Computing (MEC) [7] provides the possibility to solve this challenge. MEC technology sinks the computing power of the cloud to the edge, and takes advantage of the edge's close proximity to the end-users to effectively reduce the transmission delays of tasks and optimize the user experience [8]. In the application scenario of Internet of Vehicles, MEC is deployed at the edge side of the network such as Road Side Unit (RSU). Therefore, users can offload computational tasks to MEC servers through Vehicle to Infrastructure (V2I) communication, thus relieving the computational pressure on in-vehicle devices, reducing the power consumption of in-vehicle devices, and the transmission delay of tasks [9]. On the other hand, RSUs can also obtain the position, speed and other information of vehicles within the coverage area through V2I communication [10], which provides the possibility for optimal scheduling of vehicle resources.

This paper proceeds to delve into the design and evaluation of a dynamically adaptive multi-level MEC framework. It initiates by addressing task offloading optimization under vehicle mobility constraints and introduces a speed-sensitive offloading strategy. Following this, a collaborative backhaul system is proposed to tackle issues related to task result returns when vehicles exit RSU coverage zones. The paper further presents an improved genetic algorithm tailored for task prioritization and scheduling, and evaluates its efficacy through rigorous experimentation. In conclusion, the paper delves into the intricacies of a pivotal real-time position updating mechanism integrated within the MEC servers. This mechanism is designed to significantly bolster the system's responsiveness and adaptability. It operates by dynamically synchronizing the geographic coordinates of vehicles with the multi-level MEC architecture, ensuring that the computational offloading and result backhaul processes are aligned with the vehicles' instantaneous positions. The seamless interaction with the MEC hierarchy is facilitated through a suite of algorithms and protocols that are adept at handling the velocity and mobility patterns of vehicles. These include, but are not limited to, geo-fencing algorithms for defining coverage areas, vehicle mobility prediction models that forecast short-term movement based on historical data, and real-time communication protocols that enable efficient information exchange between the vehicle and MEC nodes.

RELATED WORKS

The high mobility and dynamics of vehicular networks pose significant operational challenges, especially in the context of 5G and the Internet of Things (IoT). In these environments, the fast movement of vehicles leads to frequent switching and coverage transitions, which affects the stability and efficiency of task offloading and collaborative backhaul. Recent research has highlighted the importance of addressing these challenges. Sheng et al. [11] demonstrated that in edge sensor networks, tasks can be divided into subtasks for distributed processing. However, their work did not adequately consider the impact of vehicle mobility on the effectiveness of task offloading. Similarly, Chen et al. [12] and You and Huang [13] focused on the division of computational tasks but did not sufficiently address the challenge posed by the movement of vehicles. Zhou et al. [14] developed a model for task segmentation that ensures all subtasks can be fully offloaded to the Multi-access Edge Computing (MEC) infrastructure before the vehicle exits the current coverage area. This approach mitigates the issue of task offloading failure due to vehicular mobility. Cao et al. [15] proposed a scheme to divide onboard tasks into independent subtasks, reducing the volume of data to be offloaded to the MEC. Ji and Jiang [16] considered the dependencies between subtasks and used genetic algorithms to solve the multi-site cooperative computational offloading problem. This minimizes the time overhead associated with the offloading of individual subtasks. Our work builds upon these foundations by introducing a novel speed-sensitive task offloading scheme designed specifically for fleets of vehicles. Unlike previous studies, our focus is on the prioritization and scheduling of multiple task offloading scenarios to ensure successful offloading before vehicles leave the coverage area. Moreover, our approach incorporates a real-time position updating mechanism that leverages MEC collaboration to enhance the success rate of task return, even as vehicles move across different coverage areas. Sun et al. [17] proposed predicting the position of a vehicle at the time of task completion based on factors such as running speed and trajectory, and introduced two distinct offloading strategies. Their work highlights the need for predictive algorithms to manage the offloading process effectively. Our research complements this by providing a solution that not only predicts but actively manages the offloading process in real time, taking into account the dynamic nature of vehicular networks. Ning et al. [18] proposes a traffic control system deployed at the base station. Utilizing the feature that the coverage of the base station is larger than the RSU, it obtains the real-time position information of the vehicle and returns the results of the task processing to the vehicle. Li et al. [19] also adopts a centralized control approach to determine whether the processing results are returned successfully or not based on the signal-to-noise ratio between the vehicle and the RSU. However, the centralized control approach not only introduces additional transmission delay, but also reduces the flexibility of system deployment and becomes another form of “cloud” [20]. Therefore, this paper proposes a dynamically changing multi-level MEC architecture, which increases the flexibility of system deployment while improving the success rate of backhauling through MEC collaboration instead of cloud-side collaboration.

In contrast to previous research, we focus on sequencing the fleet's tasks to ensure that offloading is completed in a timely manner before the vehicles leave the coverage area. This approach addresses the problem of failed task offloading due to vehicle movement. We introduce a collaborative backhaul system that leverages cooperation between MECs to improve the success rate of processing result returns. This addresses the problem of failed result returns when vehicles leave the RSU coverage area. We have improved the genetic algorithm specifically for task prioritisation and scheduling, which improves the responsiveness and flexibility of the system. Our solution ensures that vehicles can quickly offload tasks and receive results even in high-mobility situations, resulting in a safer and more responsive autonomous driving experience.

In conclusion, previous research has not fully addressed the operational challenges specific to in-vehicle networks, such as the need for real-time location updates, scalability, computational complexity, and security and privacy concerns. In this paper, we investigate speed-sensitive task offloading and co-return schemes in vehicular networks (VNETs) and introduce a dynamically evolving multi-layer MEC architecture that optimises the efficiency of task offloading and significantly improves the success rate of task return, thus providing a solution to the challenges posed by vehicular mobility in 5G and Internet of Things (IoT) environments.

SYSTEM ARCHITECTURE AND SCHEMES

Dynamically Changing Multi-level MEC Architecture

In vehicular networking, Road Side Units (RSUs) have a specific coverage area, and when a vehicle enters an RSU's coverage, it establishes a Vehicle-to-Infrastructure (V2I) connection. The Multi-access Edge Computing (MEC) server connected to the RSU provides computing resources to vehicles within the coverage area. As vehicles leave one RSU's coverage, they connect to the next RSU. RSUs also establish wired or wireless connections with other RSUs, creating a larger connection range than the V2I range. As shown in Figure 1, when a vehicle enters an RSU's coverage and generates a computational task, the MEC server for that RSU becomes the superior MEC, while other connected MEC servers become subordinate MECs. The superior MEC decides on task offloading, while subordinate MECs provide information on local link status, vehicle position, and computational capacity. Each MEC acts as both a superior and subordinate MEC, forming a dynamic hierarchical structure. This structure facilitates improved coordination and resource allocation. The superior MEC decides whether to offload a task to itself or to a subordinate MEC based on computational capacity, link quality, and vehicle velocity. It also coordinates with other superior MECs to ensure seamless task offloading and result return as vehicles move across different RSU coverage areas. The hierarchical structure optimizes task offloading and result return success rates, enabling informed decisions based on real-time information from subordinate MECs, adapting to vehicle mobility and optimizing resource utilization. Collaboration between superior and subordinate MECs ensures service continuity and minimizes average task offloading time by dynamically allocating tasks to the most suitable MEC based on network conditions and vehicle trajectory. Proactive task offloading by the superior MEC to subordinate MECs in anticipation of vehicle movement reduces the likelihood of offloading failure due to exiting RSU coverage.

In summary, the dynamically changing multi-level MEC architecture provides a flexible and scalable framework for managing computational tasks in vehicular networks, effectively handling high mobility challenges and ensuring seamless service delivery.

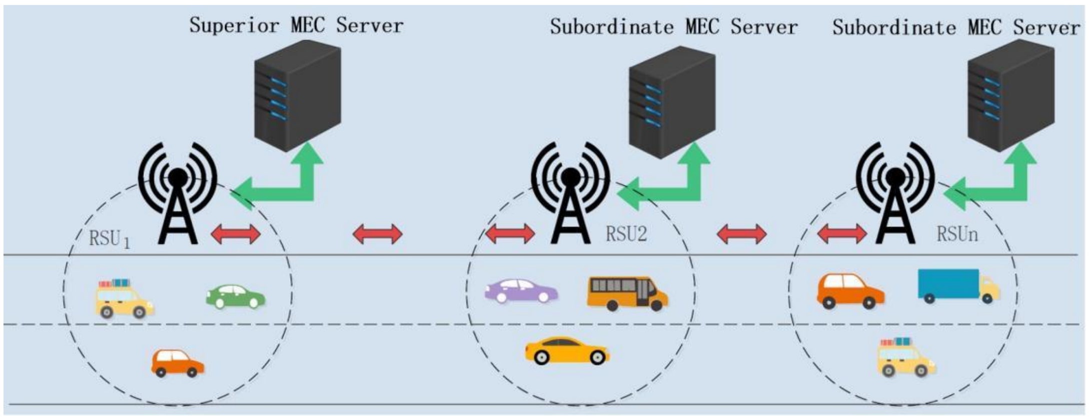

Figure 1. Superior MEC and a subordinate MEC

Multi-level MEC Architecture Sorts Offloading Tasks

At moment, there are multiple vehicles within the coverage area of a single RSU, and each vehicle may generate computational tasks. It is assumed that the communication mode between the RSU and the vehicles uses time-division multiplexing. Due to the limited link resources, the superior MEC must sort all the tasks generated at the moment, as shown in Figure 2.

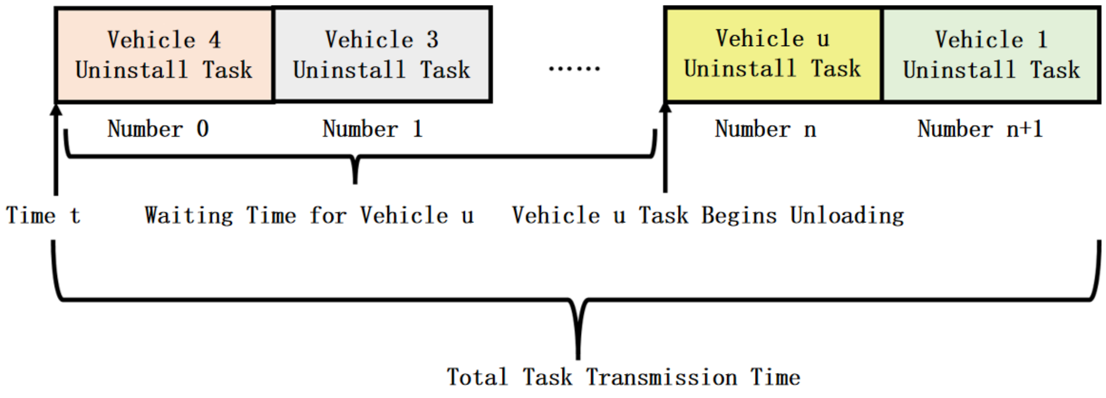

Figure 2. Sorting of tasks by superior MECs

The task with the serial number n must wait for the previous n-1 task to finish offloading before it can be offloaded. Since the vehicle is always on the move, the position of the vehicle corresponding to the task number n has changed when the previous n-1 task has finished offloading, therefore, the following effects may be caused:

1) the distance between the vehicle and the RSU changes: the change in distance will lead to the change in the link state between the vehicle and the RSU, resulting in the change of the task n offloading time, and ultimately leading to the change of the average task offloading time; 2) the vehicle moves out of the current RSU coverage area, and the average task offloading time changes.

Therefore, the superior MEC needs to consider how to sort all the tasks generated at a moment in time t to reduce the average task offloading time while ensuring that the vehicle does not drive out of the current coverage are.

SYSTEM SCHEMES

In the future intelligent transportation system (ITS), there will be a lot of negotiations between vehicles and vehicles (V2V) and vehicles and roads (V2I), which will generate a lot of task offloading and backhaul optimization problems. In the process of data transmission between vehicles, multilateral server vehicles, etc., many data services must be performed within a specified time, such as path planning services, information, entertainment and information transmission services. Therefore, the Internet of Vehicles needs to focus on how to reduce the latency of network data transmission. The use of edge computing resources, as well as the use of network function virtualization and software defense network technology, can greatly improve the efficiency and quality of end users to obtain services and reduce latency and energy consumption in data processing.

We define vehicles as autonomous agents that can act as temporary storage and computation resources for nearby vehicles, especially when the central server is overloaded or unavailable. Vehicles can perform temporary storage and computation tasks for other vehicles during peak traffic periods or in areas with limited MEC server coverage. Roadside Units (RSUs) act as communication nodes that collect information from the vehicles and forward it to the appropriate MEC servers, providing real-time updates about link status and computational capacity. As shown in Table 1, the differences between traditional and vehicular network operations are summarised. Strategically deployed Mobile Edge Computing (MEC) servers handle offloaded tasks and can operate in a centralised or decentralised manner, managing tasks over a large area or within a smaller geographic area, respectively. The task offloading mechanism utilises a genetic algorithm executed by the centralised MEC servers, which makes informed decisions on task offloading based on information provided by RSUs and vehicles.

Table 1. Differences between traditional and in-vehicle network operations

Traditional Network Operations	**Vehicular Network Operations**
Centralized servers provide all services	Centralized servers, decentralized nodes, and autonomous agents
Static network topology	Highly dynamic and mobile network topology
Limited peer-to-peer communication	Extensive peer-to-peer communication and cooperation
Fixed resource allocation	Adaptive resource allocation based on vehicle density and mobility
Predominantly static security measures	Dynamic security and privacy measures that adapt to the changing network

Speed Sensitive Task Offloading Based Scheme

Speed Sensitive Computing Offloading Model (SSCOM), a speed-aware task offloading scheme based on the classical offloading model, adds the consideration of vehicle speed and the constraints of the vehicle crossing the zone situation. The offloading model is as shown in Figure 3, this scheme uses the highway as the horizontal coordinate. the RSU is located on one side of the highway, and the perpendicular distance from the highway is y_0. Assuming that the coverage area of the RSU is L, the left boundary of the coverage area of the RSU is used as the starting point of the horizontal coordinate. Therefore, the coordinates of this RSU are $\left(\frac{L}{2}, y_0\right)$. Assume that all vehicles will not collide, the vertical coordinates are all 0, the speed of vehicle u is v_u, and all vehicles are traveling at a uniform speed within the coverage area of the RSU.

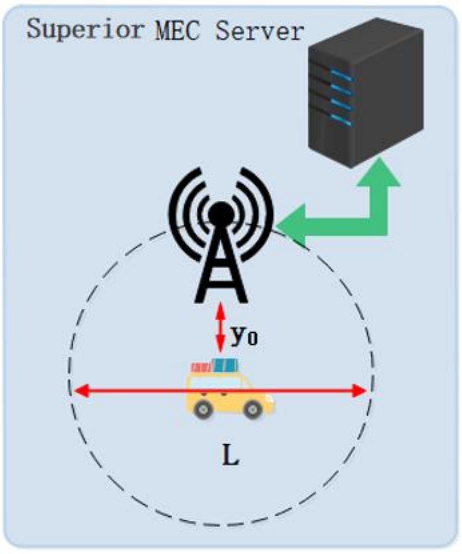

Figure 3. Internet of vehicles scenario

At moment t, vehicle u generates a computational task that needs to be offloaded. At this time, the horizontal distance between vehicle u and the left boundary of the RSU is $x_u(t)$ and the vertical distance is y_0. At the same moment, there are other vehicles that generate tasks, and their task offloading order is prioritized over the vehicle u. The set of these vehicles is $\mathrm{pre(u)}$. Therefore, the vehicle has to wait for a time of $\Delta_u = \sum_{m\in pre(u)} T_m$. The distance traveled by the vehicle during the waiting time is $v_u\Delta_u$. At this time, the vehicle's horizontal coordinate is $x_u(t+\Delta_u) = x_u(t) + v_u\Delta_u$ and the distance from the RSU can be expressed as:

$$r_u(t) = \sqrt{y_0^2 + (x_u(t+\Delta_u) - \tfrac{L}{2})^2} \tag{1}$$

At time $\mathrm{t} + \Delta_\mathrm{u}$, the data transmission rate between the vehicle and the RSU is:

$$R_u(t) = B\ log(1 + \frac{P_u h_u(t)}{\sigma_u^2}) \tag{2}$$

where B is the bandwidth of the link channel, P_u is the transmit power of vehicle u, σ_u^2 is the Gaussian white noise power, and h_u is the channel gain parameter. h_u is related to $r_u(t+\Delta_u)$:

$$h_u(t) = \frac{h^2}{r_u(t+\Delta_u)^\delta} \tag{3}$$

where h is the channel fading factor of the upload link and δ is the path loss factor. It is assumed that the position of vehicle u does not change during the offloading of its own task. Vehicle u consumes the time to offload the task of size D_u at moment $\mathrm{t} + \Delta_\mathrm{u}$ as:

$$T_u(t) = \frac{D_u}{R_u(t)} = \frac{D_u}{B\ log\left(1 + \frac{P_u \frac{h^2}{\sqrt{{y_0}^2 + \left(x_u(t) + v_u \sum_{m\in pre(u)} T_m - \frac{L}{2}\right)^2}^{\,\delta}}}{\sigma_u^2}\right)} \tag{4}$$

Assuming that the set of vehicles at time t is U and the number of tasks is N (each vehicle generates one task), the average task offloading time is:

$$\overline{T(t)} = \sum_u^U \frac{T_u(t)}{N} = \sum_u^U \frac{D_u}{BN\ log\left(1 + \frac{P_u \frac{h^2}{\sqrt{{y_0}^2 + \left(x_u(t) + v_u \sum_{m\in pre(u)} T_m - \frac{L}{2}\right)^2}^{\,\delta}}}{\sigma_u^2}\right)} \tag{5}$$

Assuming that the superior MEC offloads the tasks using the sorting method A, the following optimization objective is in place:

$$\min_{A} \overline{T(t)} \tag{6}$$

In prevent the vehicle from driving out of the coverage area of the RSU at the moment $t + \Delta_u$, the superior MEC also needs to consider the following constraints during the sequencing:

$$x_u(t + \Delta_u) \leq L \tag{7}$$

Assuming that there are a total of N tasks at time t, there are N! possible sorting methods. Therefore, in this paper, a genetic algorithm is used to solve the optimal offloading sorting scheme.

MEC Collaboration-Based Processing Result Return Scheme

During the time the MEC is processing the task, the vehicle may have already driven out of the RSU's coverage area. Even if the MEC is able to predict the vehicle's position from the vehicle's historical travel speed and return the result via inter-RSU communication, there is no guarantee that the vehicle can be found at the target MEC.

Therefore, based on the unpredictability of vehicle user behavior, this paper in troduces a real-time position update mechanism on MECs. A mapping table is maintained on each MEC. The table records the task currently being processed, and the RSU serial number of the task corresponding to the latest passing of the vehicle.

Assume that the vehicle has successfully offloaded the task to the superior MEC. The vehicle remains on an unpredictable driving trajectory while the superior MEC is processing the in-vehicle task. Therefore, the superior MEC divides the processing time of the task into multiple fixed-size time slots τ. At each time slot τ, the superior MEC sends a vehicle position update request message to the subordinate MEC, and keeps updating the serial number of the RSU in which the vehicle is located. When the task processing is finished, the superior MEC transmits the processing result back to the vehicle via inter-RSU communication based on the updated position information. The MEC acting as the superior MEC is itself a subordinate MEC to other MECs, so this MEC also responds to vehicle position update requests from other superior MECs. The superior MEC processing is shown in Figure 4.

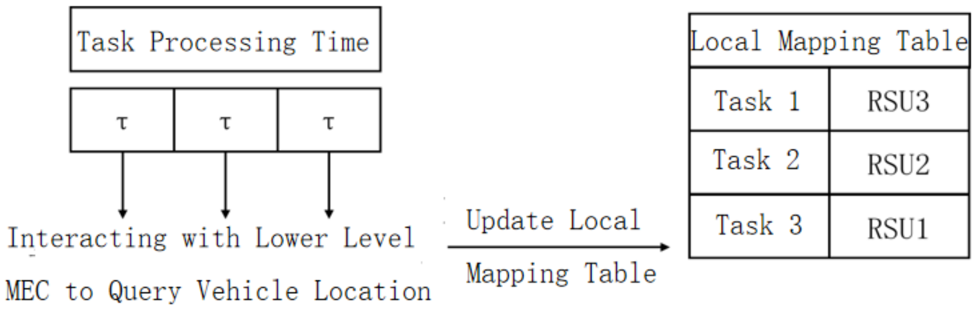
Task Processing Time
τ
τ
τ
Interacting with Lower Level MEC to Query Vehicle Location
Update Local Mapping Table
Local Mapping Table
Task 1
RSU3
Task 2
RSU2
Task 3
RSU1

Figure 4. Position of the vehicle corresponding to the superior MEC update task

Assume that the vehicle u offloads tasks Ku to the superior MEC. The superior MEC initiates position requests for the vehicle at fixed intervals. The data volume of the position request is very small, and thus imposes a negligible transmission delay. The superior MEC first checks whether the vehicle is in the coverage area of its own RSU. If it is, it updates the value in the mapping table; if it is not, it obtains the vehicle information from the subordinate MEC corresponding to the RSU through the corresponding RSU serial number in the mapping table. If the subordinate MEC also has no vehicle information, it indicates that the vehicle has driven out of the current range. However, since the subordinate MEC is the latest position of the vehicle during the last time. So it is easier to get the vehicle's position information by initiating a query from that lower.

Overcoming Algorithm Efficiency and Resource Management Challenges

The SSCOM solution addresses the challenges of algorithm efficiency and resource management in a dynamic vehicle environment. In order to reduce the computational intensity of the genetic algorithm used for task prioritisation and scheduling, parameters are optimised to achieve a balance between computational efficiency and solution quality. SSCOM dynamically assigns tasks to the most appropriate MEC servers based on network conditions and computational power, using a hierarchical MEC architecture where the upper-level MECs make informed decisions about task offloading, while lower-level MECs provide real-time information. This ensures efficient task execution and resource utilisation, minimising latency. The dynamically changing multi-level MEC architecture of the scheme facilitates seamless task offloading and result return, adapting to the high mobility and dynamic nature of in-vehicle networks. The genetic algorithm has a computational complexity of O(108), which makes it feasible to deploy in real vehicle networks considering the modest memory and CPU requirements. Thus, SSCOM ensures efficient task offloading and resource utilisation, making it a practical solution for vehicular networks.

EXPERIMENTS

Based on the above system scheme, this paper conducts simulation experiments on the offloading scheme. The optimization objective of the speed-aware task offloading scheme is to minimize the average task offloading time of the task under the constraint that all vehicles do not drive out of the RSU range. For the NP problem, this scheme is solved using a genetic algorithm. It is assumed that the number of genes N on the chromosome represents the number of momentary tasks. Each gene represents the vehicle number that generates the task. One arrangement of genes on the chromosome represents one possible arrangement of task offloading. The specific process is as follows:

Parameter Selectionn

Parameter Selection: The parameters of the genetic algorithm have a significant impact on the performance of the SSCOM scheme, particularly the mutation probability, replacement probability, and the number of genetic iterations. To achieve optimal performance, we employ the method of controlled variables to study the effects of these parameters on the average task offloading time.

a) Mutation Probability: The specific settings are the initial population size of 1000, the number of genes (vehicles generating tasks) is 100, the number of iterations is 1000, and the mutation probability varies over the range [0, 0.1]. As shown in Figure 5, the average task offloading time tends to decrease within the range [0, 0.04] and then increases with further increases in the mutation probability. Therefore, the mutation probability chosen for this scheme is 0.04 (Figure 5).

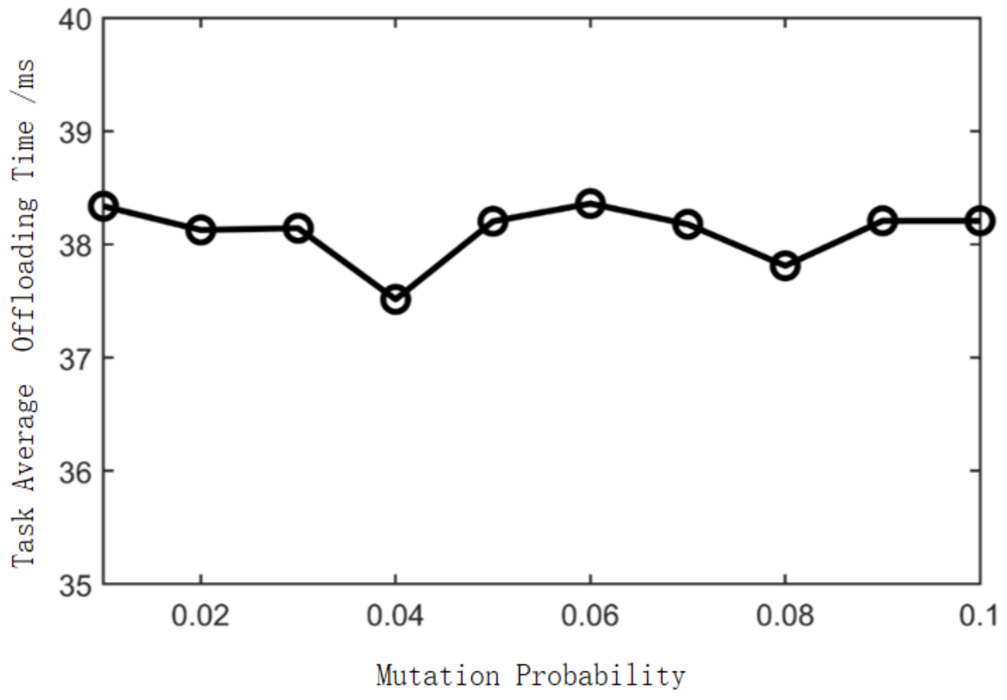

Figure 5. Mutation probability distribution

b) Replacement Probability: The specific settings are the initial population size of 1000, the number of genes is 100, the number of iterations is 1000, and the replacement probability varies over the range [0, 1]. As depicted in Figure 6, the replacement probability chosen for this scheme is 0.5, as it provides the optimal balance between introducing new chromosomes and maintaining diversity in the population.

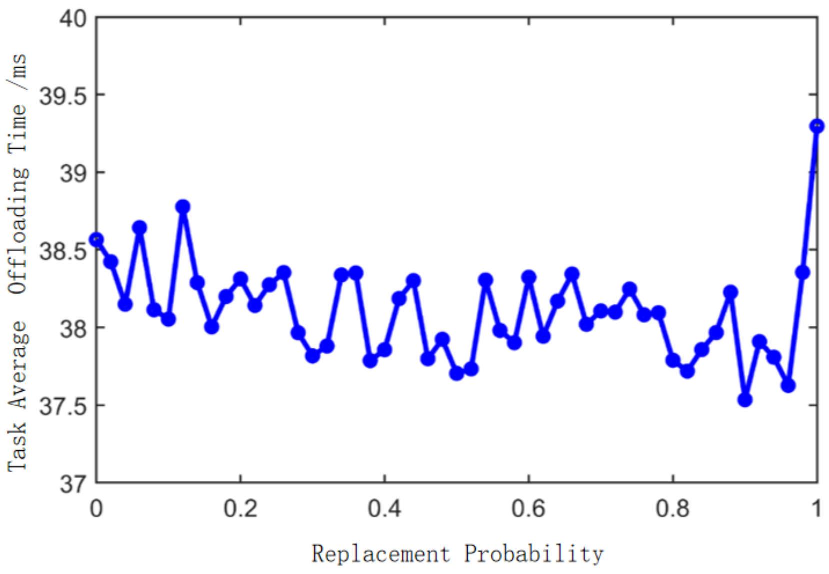

Figure 6. Replacement probability distribution

c) Number of Genetic Iterations: The specific settings were the initial population size set to 1000, the maximum number of iterations set to 1500, the number of genes set to 40, 80, and 120, respectively, the mutation probability is 0.04, and the replacement probability is 0.5. The number of iterations that yields the most stable performance without excessive computational overhead is selected. It is determined that the optimal number of iterations is 1000, balancing the trade-off between convergence speed and solution quality.

Test Process and Results

Program selection and setup

Based on the parameter selection experiments described in Section 5.1.1, we now conduct the main simulation experiments to evaluate the SSCOM scheme. The experimental parameters involved in this experiment are shown in Table 2.

Table 2. Parameters of simulation experiment

Experimental Parameters	Value
Link channel bandwidth B	10MHz
Vehicle transmit power P_u	500mW
Channel fading factor h	1
Path Loss Factor δ	4
RSU Coverage L	400m
RSU distance from y_0	10m
Gaussian white noise power σ_u^2	-100dB
Vehicle speed v_u	[16, 32]m/s
Number of vehicles N	[10, 150]
Task data size D_u	[0.1, 3.1]MB
Number of populations	1000
Maximum number of iterations	1000
Mutation probability	0.04
Replacement probability $y0$	0.5

Algorithm performance

To evaluate the performance of the SSCOM scheme, we vary the number of vehicles and observe the changes in the average task offloading time. As illustrated in Figure 7, when the number of vehicles is low, the difference between the schemes is minimal due to the limited number of tasks. However, as the number of vehicles increases, the SSCOM scheme achieves the optimal average task offloading time relative to the other schemes. For instance, at a high density of 120 vehicles, the average offloading time for the SSCOM scheme is 37 ms, which is 32% lower than the random offloading scheme (49 ms), 18% lower than the speed-first offloading scheme (45 ms), and 16% lower than the data-volume-first offloading scheme (43 ms).

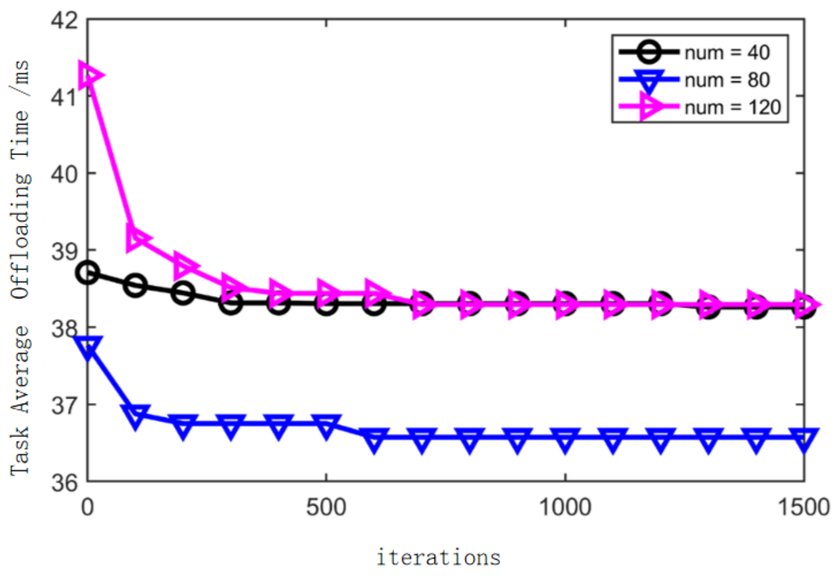

Figure 7. Effect of the number of iterations on the performance of the scheme

Computational resource requirements

The SSCOM scheme requires computational resources for the genetic algorithm to run, which includes memory for storing the population and fitness values, and CPU cycles for performing the genetic operations. The memory requirement is proportional to the population size and the number of genes, and the CPU requirement is proportional to the computational complexity of the algorithm. Given the parameters used in the SSCOM scheme (Table 2), the memory requirement is moderate, as the population size is 1000 and the number of genes is 100. The CPU requirement is also reasonable, given that modern computing hardware can efficiently handle the computational complexity of O(10 ^8).

Comparative analysis

To further understand the performance of the SSCOM scheme, we analyze the average vehicle traveling speed and the average data volume of tasks for each group of tasks in the case of a group of tasks with 10 vehicles when the number of vehicles is 100. As shown in Figure 8, the speed-first offloading scheme prioritizes tasks associated with faster vehicles, whereas Figure 9 demonstrates that the data volume priority offloading scheme favors tasks with larger data volumes. In contrast, the SSCOM scheme takes into account both the speed and data volume to optimize performance.

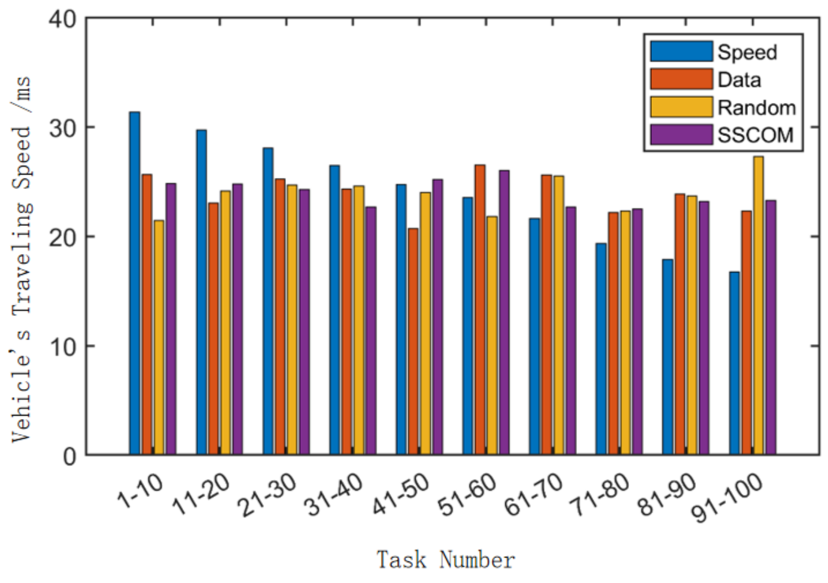

Figure 8. The relationship between offloading scheme and vehicle speed

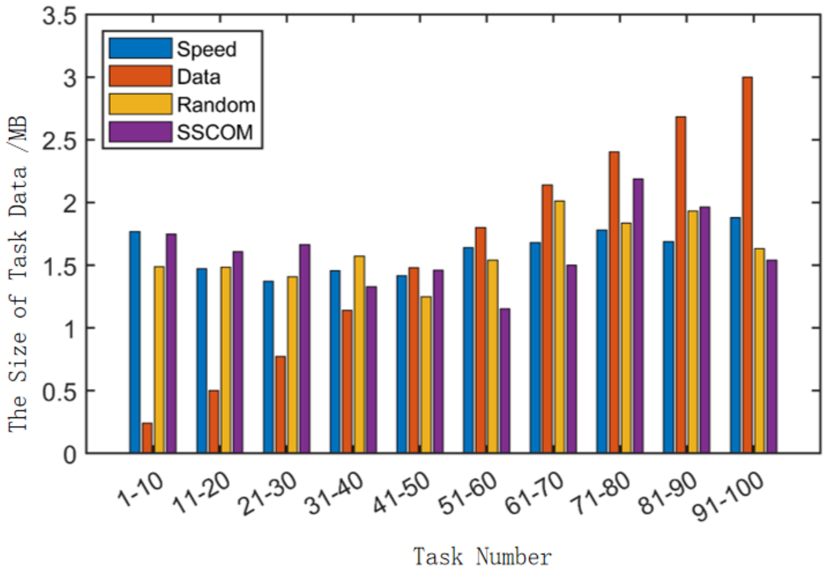

Figure 9. The relationship between offloading scheme and task data volume

Testing results

We conducted experiments with varying vehicle densities to examine the SSCOM scheme's behavior under different traffic conditions. The density ranged from low (20 vehicles) to high (120 vehicles) in the RSU coverage area. As shown in Figure 10, the SSCOM scheme consistently performed well, maintaining a lower average task offloading time compared to alternative schemes, even as the number of vehicles increased. Specifically, at a high density of 120 vehicles, the average offloading time for the SSCOM scheme was 37 ms, which is 32% lower than the random offloading scheme (49 ms), 18% lower than the speed-first offloading scheme (45 ms), and 16% lower than the data-volume-first offloading scheme (43 ms).

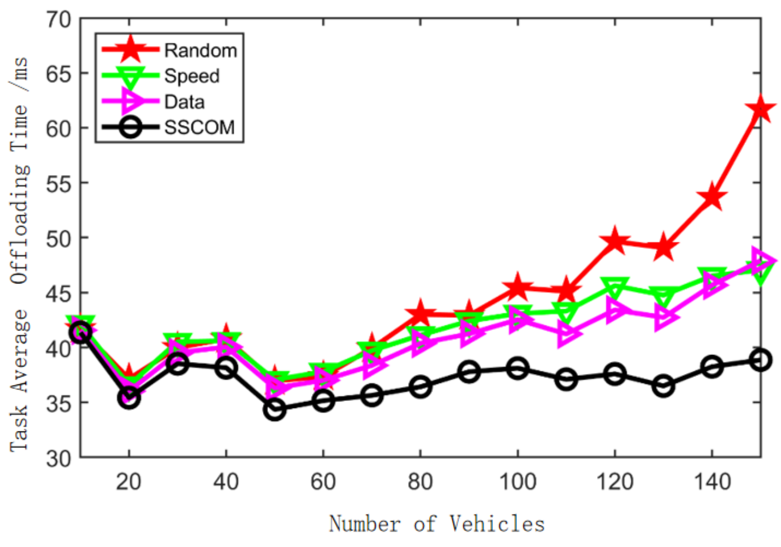

Figure 10. Comparison of various offloading schemes

Analysis and Extension

Detailed analysis of the genetic algorithm

The genetic algorithm used in the SSCOM scheme is tailored to the specific needs of vehicular networks. The optimization objective is to minimize the average task offloading time of the task under the constraint that all vehicles do not drive out of the RSU range. As mentioned earlier, the number of genes on the chromosome represents the number of momentary tasks. Each gene represents the vehicle number that generates the task. One arrangement of genes on the chromosome represents one possible arrangement of task offloading. The genetic algorithm operates with a population size of 1000, a maximum number of iterations of 1000, and a mutation probability of 0.04. These parameters were selected based on the parameterization experiments described in Section 5.1.1.

Algorithm complexity analysis

The SSCOM scheme utilizes a genetic algorithm for task prioritization and scheduling. The complexity of the genetic algorithm is influenced by the number of genes, the population size, and the number of iterations. The computational complexity of the genetic algorithm can be estimated as $O(N * P * I)$, where N is the number of genes, P is the population size, and I is the number of iterations. For the SSCOM scheme, the number of genes (N) is set to 100, the population size (P) is set to 1000, and the number of iterations (I) is set to 1000. Therefore, the computational complexity of the genetic algorithm in the SSCOM scheme is approximately $O(10^8)$.

Scalability and practicality

The SSCOM scheme is designed to be scalable and practical. The genetic algorithm used for task prioritization and scheduling has a computational complexity of O(108), which is feasible for deployment in realistic vehicular networks. The memory and CPU requirements are moderate and can be met by current computing hardware. The SSCOM scheme ensures efficient task offloading and resource utilization, making it a practical solution for vehicular networks.

OPERATIONAL ISSUES

To address the operational complexities inherent in vehicular networks, we concentrate on five key business challenges: task offloading at varying speeds, collaborative backhaul during coverage range transitions, real-time location updates, scalability and computational complexity, and security and privacy. These challenges are critical to the effective operation of vehicular networks and require comprehensive solutions.

Security and Privacy

To ensure data security and privacy, we propose implementing centralized solutions that incorporate secure data transfer protocols, specifically Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), to encrypt data both in transit and at rest. Robust access control mechanisms are essential to regulate who can offload tasks and receive results. Authentication mechanisms are included to verify the identities of vehicles and users, preventing unauthorized access. We also recommend encrypting and protecting both static and dynamic data using end-to-end encryption for data transmission and strong encryption algorithms, such as the Advanced Encryption Standard (AES) or Elliptic Curve Cryptography (ECC), for storing data. Mutual authentication protocols should be implemented between superior and subordinate MEC (Mobile Edge Computing) nodes to ensure that only legitimate MEC nodes can communicate, and that data is returned only to vehicle-related MEC nodes.

Realistic Validation

For realistic validation, we propose a comprehensive experimental setup that includes both simulations and field trials. Extensive simulations using real-world vehicular network scenarios will evaluate the performance of our proposed scheme. Field trials, conducted in collaboration with industry partners, will assess the practicality and effectiveness of the solution under realistic conditions.

Real-Time Location Updates

To integrate real-time location update algorithms into a multi-tier MEC architecture, we propose a scalable and flexible approach involving deploying real-time location services on MEC servers. These services update vehicle locations as they move within the network, ensuring that all relevant MEC servers are synchronized. This maintains the operational integrity of the vehicular network and supports the efficient management of tasks and resources.

CONCLUSION

This paper has presented a dynamically changing multi-level MEC architecture aimed at minimizing average task offloading times without compromising on-service continuity for vehicles traversing RSU coverage areas. By integrating vehicle speed and crossing scenarios into the classical offloading model, our proposed Speed Sensitive Collaborative Offloading and Backhaul (SSCOM) scheme has demonstrated a superior capability to minimize offloading times and enhance the task return success rate by at least 5%, outperforming conventional methods. Its practical significance lies in facilitating seamless and efficient service delivery in IoT environments, where timely data processing and reliable information feedback are critical. The SSCOM scheme ensures that vehicles can swiftly offload tasks and receive processed results even amidst high mobility, contributing to safer and more responsive autonomous driving experiences.

However, this study also has its limitations. Firstly, the parameter settings and scenario design in the simulation experiments are relatively simplified; future work should further validate the effectiveness of the proposed schemes in more complex real-world environments. Secondly, this paper's schemes primarily focus on the performance of task offloading and result return, with less in-depth discussion on security and privacy concerns during the task offloading process. In future work, we intend to incorporate security mechanisms to ensure the safe transmission of data and the protection of privacy. Looking ahead, we anticipate that research in the field of vehicular networking will progress towards more intelligent, service-oriented, and secure development. With the proliferation of 5G technology and the enhancement of edge computing capabilities, vehicular networks will be able to offer richer and more efficient services.

ACKNOWLEDGMENT

My work was supported by the self-funded science and technology Innovation Project of Foshan City in 2022 (ID:2220001005675), Guangdong Provincial Department of Education Youth Innovation Talent Project (ID:2021KQNC- X123), Guangdong Software and Information Service Engineering Technology Research Center (ID:2022E07) and "Climbing" Program (pdjh2022b0607) from Guangdong Province.

REFERENCES

[1] C. Chen, C. Wang, T. Qiu, M. Atiquzzaman, and D. O. Wu, "Caching in vehicular named data networking: Architecture, schemes and future directions," IEEE Communications Surveys and Tutorials, vol. 22, no. 4, pp. 2378-2407, 2020.

[2] S. Talal, W. S. M. Yousef, and B. Al-Fuhaidi, "Computation offloading algorithms in vehicular edge computing environment: A survey," in 2021 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE), pp. 1-6, 2021.

[3] G. Liu, F. Dai, B. Huang, Z. Qiang, S. Wang, et al., "A collaborative computation and dependency-aware task offloading method for vehicular edge computing: a reinforcement learning approach," Journal of Cloud Computing, vol. 11, no. 1, p. 68, 2022.

[4] S. Raza, W. Liu, M. Ahmed, M. R. Anwar, M. A. Mirza, et al., "An efficient task offloading scheme in vehicular edge computing," Journal of Cloud Computing, vol. 9, pp. 1-14, 2020.

[5] L. Han, M. Dong, and Song, "Service-based virtual network function placement algorithm in vehicle networking," High-tech Communication, vol. 31, pp. 341-349, 2021.

[6] X. Huang, L. He, X. Chen, G. Liu, and F. Li, "A more refined mobile edge cache replacement scheme for adaptive video streaming with mutual cooperation in multi-mec servers," in 2020 IEEE International Conference on Multimedia and Expo (ICME), pp. 1-6, 2020.

[7] Z. Wu and D. Yan, "Deep reinforcement learning-based computation offloading for 5g vehicle aware multi-access edge computing network," China Communications, vol. 18, no. 11, pp. 26-41, 2021.

[8] R. Dhanare, K. K. Nagwanshi, S. Varma, and S. Pathak, "The future of internetof vehicle: Challenges and applications," in 2021 International Conference on Computational Performance Evaluation (ComPE), pp. 023-026, 2021.

[9] H. Zhang, Z. Wang, and K. Liu, "V2x offloading and resource allocation in sdn-assisted mecbased vehicular networks," China Communications, vol. 17, no. 5, pp. 266-283, 2020.

[10] A. Talpur and M. Gurusamy, "Drld-sp: A deep-reinforcement-learning-based dynamic service placement in edge-enabled internet of vehicles," IEEE Internet of Things Journal, vol. 9, no. 8, pp. 6239-6251, 2022.

[11] Z. Sheng, C. Mahapatra, V. C. M. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 114-126, 2018.

[12] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," IEEE/ACM Transactions on Networking, vol. 26, no. 4, pp. 1619-1632, 2018.

[13] C. You and K. Huang, "Exploiting non-causal cpu-state information for energy-efficient mobile cooperative computing," IEEE Transactions on Wireless Communications, vol. 17, no. 6, pp. 4104-4117, 2018.

[14] J. Zhou, D. Tian, Y. Wang, Z. Sheng, X. Duan, et al., “Reliability-optimal cooperative communication and computing in connected vehicle systems,” IEEE Transactions on Mobile Computing, vol. 19, no. 5, pp. 1216-1232, 2019.

[15] J. Cao, L. Yang, and J. Cao, “Revisiting computation partitioning in future 5g-based edge computing environments,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2427-2438, 2019.

[16] Z. H. Ji and L. Y. Jiang, “A multi-site collaborative unloading algorithm based on genetic algorithm,” Computer Engineering and Science, vol. 43, pp. 426-434, 3 2021.

[17] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, “Adaptive learning-based task offloading for vehicular edge computing systems,” IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3061-3074, 2019.

[18] Z. Ning, K. Zhang, X. Wang, M. S. Obaidat, L. Guo, et al., “Joint computing and caching in 5g envisioned internet of vehicles: A deep reinforcement learning-based traffic control system,” IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 8, pp. 5201-5212, 2020.

[19] M. Li, J. Gao, L. Zhao, and X. Shen, “Deep reinforcement learning for collaborative edge computing in vehicular networks,” IEEE Transactions on Cognitive Communications and Networking, vol. 6, no. 4, pp. 1122-1135, 2020.

[20] X. P. W. Ye and C. X. R. Yang, “Multi-agent reinforcement learning edge-cloud collaborative unloading for vehicle networking,” Computer Engineering, vol. 47, pp. 13-20, 4 2021