RESEARCH Open Access

Joint optimization of network selection and task offloading for vehicular edge computing



Lujie Tang¹, Bing Tang^{1*}, Li Zhang¹, Feiyan Guo¹ and Haiwu He²

Abstract

Taking the mobile edge computing paradigm as an effective supplement to the vehicular networks can enable vehicles to obtain network resources and computing capability nearby, and meet the current large-scale increase in vehicular service requirements. However, the congestion of wireless networks and insufficient computing resources of edge servers caused by the strong mobility of vehicles and the offloading of a large number of tasks make it difficult to provide users with good quality of service. In existing work, the influence of network access point selection on task execution latency was often not considered. In this paper, a pre-allocation algorithm for vehicle tasks is proposed to solve the problem of service interruption caused by vehicle movement and the limited edge coverage. Then, a system model is utilized to comprehensively consider the vehicle movement characteristics, access point resource utilization, and edge server workloads, so as to characterize the overall latency of vehicle task offloading execution. Furthermore, an adaptive task offloading strategy for automatic and efficient network selection, task offloading decisions in vehicular edge computing is implemented. Experimental results show that the proposed method significantly improves the overall task execution performance and reduces the time overhead of task offloading.

Keywords: Mobile edge computing, Vehicular networks, Task offloading, Performance optimization

Introduction

Edge computing is an open platform which integrates core capabilities of computing, storage, and services at the edge of the network that near the source of sensor data. It provides intelligent services at the edge which meets the key needs of industry digitalization, real-time data streaming, data intelligence, security and privacy protection, etc. Edge computing has obtained great attentions from academia, industry, and government. Mobile edge computing (MEC) offers cloud capabilities and service environment for Internet of Things (IoT) applications at the edge of the mobile cellular network, such as 5G, which is regarded as a promising solution in which computing resources are pushed to the radio access network

(RAN) and services are provided near the devices [1]. In the MEC-based vehicular network environment, MEC servers with powerful computing and storage capabilities are installed at the edge of the vehicular networks, usually collocated with the roadside units (RSUs). The communications between the vehicles and RSUs is through small cell networks. Each edge server is collocated with an access point (AP) (e.g., 5G base station). Users can require the mobile edge services by selecting a nearby AP. The services for users are deployed at a nearby edge server rather than the remote cloud, so as to reduce the latency from end devices to the cloud-hosted service. Merging MEC with the dense deployment of 5G macro/micro base stations, makes possible a ultra-low latency access to cloud functionalities [2–4].

Full list of author information is available at the end of the article



^{*}Correspondence: btang@hnust.edu.cn

¹School of Computer Science and Engineering, Hunan University of Science and Technology, 411201 Xiangtan, China

Vehicular networks have gained huge popularity in recent years due to their wide range of applications. Many new types of vehicular services have emerged to provide good travel experience for users, such as autonomous driving, traffic safety and traffic monitoring. These services depend on computing-intensive workloads with time constraint. Although with the continuous advancement of technology, today's vehicles have obtained more capability than before, and many in-vehicle applications can be completed by the vehicle's local computing power. However, there are still some services that the vehicle cannot handle, such as autonomous driving technology, which establishes and recognizes the driving environment of the vehicle through radar, sensing and monitoring equipment on the vehicle, and then realizes the automatic control of the vehicle. Facing the complex and timevarying road network traffic environment, the characteristics of weak processing capacity and low storage capacity of vehicle on-board equipment will greatly restrict the real-time processing and effective storage of large-scale traffic information collected by vehicles, which will seriously affect the safety and reliability of autonomous driving. At the same time, some new applications such as road environment augmented reality (AR), traffic behavior intelligent guidance, voice-based human-vehicle dynamic interaction, etc., usually also require powerful computing capability and the support of massive data content, which are bringing users with rich and convenient driving experience, and also poses severe challenges to the computing and storage capabilities of smart vehicles. The introduction of MEC into vehicular networks is an effective way to solve the above problems. In the MEC-based vehicular networks environment, the computation tasks could be executed locally in the vehicles or offloaded to the nearby MEC servers, according to conditions and constraints [5, 6]. However, it is still faced with the following challenges:

First, the vehicles have significant movement characteristics, and their geographical positions change with time. If the vehicle runs a large amount of computation tasks, it usually passes through the coverage of multiple cells during task executions. During the movement of the vehicle, the distance between the vehicle and the edge server that serves the vehicle is dynamically changing. If the distance between them becomes too large, the quality of service (QoS) provided cannot be guaranteed, and user experiences will also decrease. Therefore, there is a need for a strategy to dynamically adjust task execution as vehicles moving [6].

Second, the vehicle maybe within the coverage of multiple available APs at the same time in a cell. If the user chooses the AP autonomously, it may cause resource competition and network congestion. Therefore, how to select

the appropriate network for each vehicle is also a problem to be considered.

Third, when the vehicle is performing network selection and task offloading, if the AP selected by the vehicle is close to the edge server where the task is executed, the access delay is only occurred by the connection to the AP. On the contrary, the task needs to be transmitted from the AP to the edge server far away, which introduces additional communication overhead. There are not any specific protocols that are used during the execution of the algorithms for the network selection and the task offloading. How to balance the access delay of the network access and the communication delay is a difficult problem to be solved.

In addition, if the data volume of the computation task is large and the coverage of the cell is relatively small, during the offloading process, the vehicle may have drove out of the previous cell before the task has completely offloaded, so that the interrupted task needs to be retransmitted, which increases communication overhead. Therefore, static task offloading strategy cannot solve this problem, and how to plan the task offloading in a dynamic manner is a problem that this paper focuses on.

In this paper, we propose an adaptive task offloading strategy in the MEC-based vehicular networks environment, which considers a scenario where the vehicle needs to pass through multiple cells during the offloading process for a large task. The impact factors we considered in the adaptive computation task offloading strategy include vehicle speed, cell coverage, data transmission rate, access point load in the cell, and MEC server workload. Considering these impact factors, an optimal plan is made for the next offloading strategy, so as to achieve that there will be no interruption of task execution when passing through multiple cells during the offloading process, and also try to achieve the minimized total delay in completing the tasks

In the offloading strategy proposed in this paper, first, the whole task is divided into many small task units (TUs) [7], and then, based on some restrictions, we obtain the following decisions: the AP to which the task is connected, and the edge server to which the task is offloaded, and the number of TUs allocated to the edge server, and the proportion of the corresponding number of TUs offloaded to the edge server to the total TUs allocated to the cell. It is an online strategy that whenever a vehicle enters a new cell, the above offloading strategy will be re-executed to obtain an optimal computation task offloading scheme for the new cell. Therefore, the strategy proposed can achieve a lower delay for task execution without interrupting the computation process.

The main contributions of this paper are summarized as follows:

- We proposed a pre-allocation algorithm for vehicle tasks. It can comprehensively consider vehicle movement characteristics and the surrounding environment of the vehicle, and dynamically adjust the execution and offloading of tasks.
- We proposed the optimization of the selection of network access points, which reduces network congestion and resource competition caused by vehicles choosing network access points independently.
- We designed and implemented an adaptive offloading strategy, which can provide vehicles with automatic and efficient network access selection, task offloading and task migration decisions.

The rest of this paper is organized as follows. The second section summarizes related work. Then, we introduce our system model and solution, and proposed a pre-allocation algorithm and an adaptive offloading strategy, followed by simulation and result analysis. The final section summarizes the whole paper and discusses future work.

Related work

In recent years, the edge computing paradigm has attracted great attentions from academia and industry. It has the characteristics of fast processing speed and short response time, which could make cloud computing services closer to end users [8]. Edge computing has provided a powerful driving force for many key technologies such as 5G, IoT, AR, and vehicle-to-vehicle (V2V) communication, etc. There are three common edge computing models: Cloudlets [9], fog computing [10], and multi-access/mobile edge computing [11]. MEC, as a new architecture that moves service capabilities from the core network to the edge network, has caused extensive research by scholars [6, 12–14].

The key research topics in MEC include the placement of edge servers, computing migration and offloading, and edge caching, etc. Computation migration and offloading are about migration decisions and resource allocation. By migrating mobile device tasks/applications to servers in network for execution, it can enhance the computing capabilities of mobile devices and reduce time and energy consumption when running applications on mobile devices [1]. In recent years, several studies have addressed the mobile task offloading in the MEC scenario. The task offloading can be classified into several aspects according to optimization objectives, includes: delay or latency constrained offloading [15, 16], energy-efficient offloading [17, 18], energy-latency tradeoff for offloading [19, 20], cost-efficient offloading [21], etc. Generally speaking, the offloading is a multi-objective optimization

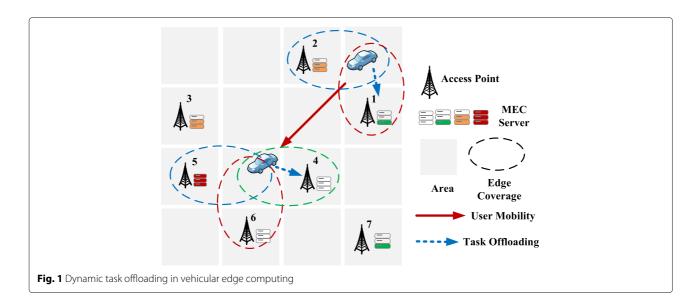
problem, which is usually solved by optimized or heuristic algorithms.

Task partitioning and task division are usually adopted in offloading. In [22], Wu et al. proposed a path-based offloading partitioning algorithm to determine which portions of the application tasks to run on mobile devices and which portions on cloud servers with different cost models in mobile environments. In [23], Kiani and Ansari proposed a task scheduling scheme designed for code partitioning over time and the hierarchical cloudlets in a mobile edge network. Similar work includes [24], which proposed a partial offloading technique for wireless mobile cloud computing. In [7], Wang et al. also divided the whole task into several small task units, taking into account the divisibility of task, and proposed a dynamic offloading in MEC-enabled vehicular networks, which is similar to our work. Compared to [7], our work considers multi-severs and APs for the decision, which have not mentioned by them. However, most of the existing work have not considered the practical constraints about the variable moving speed of vehicles.

Some studies about offloading and migration are focused on user mobility predictions in mobile edge networks. For example, the work in [25] is to formulate the mobility driven decision making problem for service migration using the framework of Markov Decision Process (MDP). By using the MDP model to predict, the work [25] makes decisions whether to migrate services. In [26], Alasmari et al. also proposed a MDP-based methodology to intelligently make decision for optimizing multiple objectives.

In [27], Sun et al. developed an energy-aware mobility management scheme to optimize the total delay due to both communication delay and computation delay under the long-term energy consumption constraint of the user, without requiring the future user mobility as a priory knowledge. In [28], Gao et al. proposed joint network selection and service placement for mobile edge computing. The authors considered the nonlinear network access latency, switching latency, and communication latency to minimize overall latency, and designed an online algorithm to reduce frequent switching cost and balance the access delay and communication delay. In [15], a contractbased offloading and computation resource allocation scheme was proposed to maximize the benefit of the MEC service provider with consideration on the vehicle mobility in cloud-enabled vehicular networks.

Resource sharing of access networks and edge servers are important issues recently studied, for example the graph-based cooperative scheduling proposed in [29], the matrix game approach proposed in [30], P2P-enabled decentralized edge servers approach in [21], etc. In [31], Sardellitti et al. proposed and solved the offloading problem of joint optimization of the radio resources and



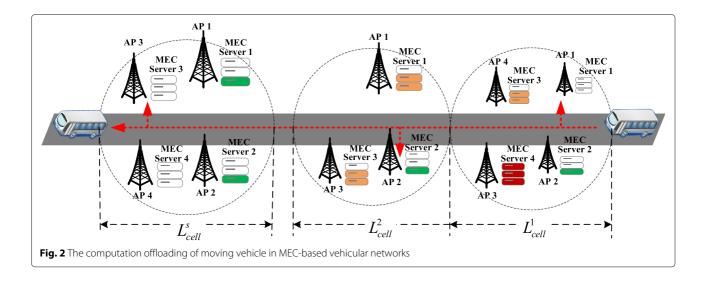
the computational resources in order to minimize the overall users' energy consumption, while meeting latency constraints. In [32], the authors proposed an adaptive sequential offloading game approach, where the mobile users sequentially make offloading decisions based on the current interference environment and available computation resources, and adjust the number of offloaded users adaptively. Poularakis et al. studied the joint optimization of service placement and request routing in MEC-enabled multi-cell networks with storage-computation-communication constraints [33].

Data security and privacy protection in the field of edge computing have also attracted the attention of many scholars [34, 35]. The integration of blockchain and edge computing is becoming an important concept that leverages their decentralized management and distributed service to meet the security, privacy protection, scalability and performance requirements in future networks and systems [36]. In [37], Gai et al. proposed a model permissioned blockchain edge model for smart grid network (PBEM-SGN) to address the two significant issues in smart grid, privacy protections, and energy security, by means of combining blockchain and edge computing techniques. In [38], the authors exploited consortium blockchain and smart contract technologies to achieve secure data storage and sharing in vehicular edge networks and proposed a reputation-based data sharing scheme to ensure high-quality data sharing among vehicles. In [39], physical-layer assisted privacy-preserving offloading schemes was proposed and two efficient algorithms are developed to address the corresponding optimization problems by exploiting the favorable structure of the privacy-preserving offloading problem in the delay optimal and the energy optimal scenarios. In [40], a hierarchical blockchain-enabled federated learning algorithm for knowledge sharing is proposed in IoVs. The hierarchical blockchain framework is able to not only improve the reliability and security of knowledge sharing, but also adapt to the large scale vehicular networks with various regional characteristics.

Compared with these work, the difference is that we consider a multi-cell MEC scenario, where the small cells are densely deployed and serve multiple mobile vehicles. We consider multi-server and multi-AP in multi-cell for the decision, and also consider the mobility of vehicles. The novelty in our offloading strategy is that, each access point is equipped with an edge server, multiple access points are included in a cell, and we study the joint optimization of access points selection and task offloading to decrease queuing delay and task execution with task division, due to the competition for limited computation resources.

System model and solution

As you can see, Fig. 1 shows the dynamic task offloading process of vehicles in the vehicular networks based on MEC. There are multiple available network access points around the vehicle. At the initial time, the vehicle is located in the upper-right corner, the TUs are offloaded to the edge server named Edge 1. Then, the vehicle moves to the next position along the red arrow. When this movement occurs, the coverage of the surrounding edge server will change. If the distance between the vehicle and the edge server is too large, the QoS provided will not be guaranteed, and the user experience will also decline. Therefore, it needs to offload the unfinished TUs to the new edge server to execute. As shown in this figure, Edge 4, Edge 5 and Edge 6 are three candidates, and finally the unfinished TUs are offloaded to the edge server named Edge 4 according to a decision.



We consider a straight road with successive small cells in MEC-enabled vehicular networks, and the cells are represented by a set $L=\{L_{cell}^1,L_{cell}^2,...,L_{cell}^s,...\}$, as you can see in Fig. 2. In order to facilitate understanding of system models and algorithms, we list main notations in Table 1. We assume that the cells are closely adjacent, and their coverage does not overlap each other, and it is also assumed that the communication coverage of the cell is relatively small, and its coverage is about 100m to 400m or less. The coverage radius of each cell is represented by a set $r = \{r_1, r_2, ..., r_s, ...\}$. There are multiple APs and edge servers with powerful computing and storage capabilities in each cell. It is assumed that there are m APs and n edge servers in the coverage area of the cell L_{cell}^s , which are denoted by $M = \{1, 2, 3, ..., m\}$ and $N = \{1, 2, 3, ..., n\}$, respectively. An edge server in the cell can serve multiple vehicles at the same time through CPU sharing, but the computing resources allocated to each vehicle are limited. As can be seen in Fig. 3, the objective of optimization decision is to optimally allocate tasks to m APs and n edge servers.

We suppose that the on-board device in the vehicle has a large computing tasks to be finished when it is moving on the road. Due to the heterogeneity of computing tasks, we denote the task for vehicle k as u_k , and denote the computation input data bits of task u_k as s_k , and w_k represents the CPU cycles required to process the task u_k , which can be calculated through $w_k = \omega s_k$, where the parameter ω depends on the computational complexity of the task u_k . We denote the computation capacity of on-board device as f_l^k . In addition, C_j is denoted as the total computing power of edge server j, and f_m^j denotes the computation capacity allocated to the vehicle by edge server j. For the task u_k , the execution time on local device is expressed by w_k/f_l^k , and the execution time on edge server j is w_k/f_m^j .

Taking into account the divisibility of task, we can divide the whole task into several small TUs. By dividing the task into small TUs, it can be accurately controlled according to the vehicle's speed and wireless network status. We can make a decision on each TU whether it should be processed locally or offloaded to the current-connected edge server. Even if there is an interruption during the task execution, there is no need to retransmit the whole computing task, and only the interrupted TU needs to be retransmitted. We assume that the size of each TU equals to I_0 bits. The total number of TUs of task u_k is ns_k , which is calculated by

$$ns_k = \left\lceil \frac{s_k}{I_o} \right\rceil \tag{1}$$

where \square is the ceil function.

As the vehicle passes through several cells, the whole task would be finished by several edge servers in successive cells. Thus, how many TUs should be offloaded to edge servers in each cell is the scheduling problem solved in this paper. The maximum number of TUs that can be completed in the cell is defined as N_k^s , and then the amount of data that need to be processed in the cell is given by:

$$D_k^s = N_k^s I_o. (2)$$

In addition, for the maximum number of TUs N_k^s , we need to calculate the number of TUs should be assigned and processed in edge servers according to the conditions, and the number of TUs processed in local. We define the optimal offloading ratio α_k^s which indicates the ratio of the number of TUs offloaded by vehicle k to the maximum number of TUs N_k^s in the cell L_{cell}^s .

Table 1 Notations

Symbol	Definition
L	The set of the cells
r	The set of cells coverage radius
u_k	The task for vehicle k
s_k	The computation input data bits of task u_k
W_k	The CPU cycles required to process the task u_k
f_l^k	The computation capacity of on-board device
C_j	The total computing power of edge server j
f_m^j	The computation capacity allocated to the vehicle by edge server j
lo	The size of each TU
$r_{i,k}$	The maximal uplink rate assigned to the vehicle k through AP i
T ^{AP} i,k	The waiting time of the vehicle k due to insufficient bandwidth of AP i
$T_{j,k}^Q$	The queuing delay of the task execution on edge server j
$T_{i,j}^{C}$	The communication delay from AP i to edge server j
N_k^s	The number of TUs that can be completed in the cell
$lpha_k^{s}$	The proportion of the number of TUs offloaded to the edge server
T ^{s,tra} i,k	The transmission latency of the offloading task that the vehicle $\it k$ needs to complete in the cell
$T_k^{s,loc}$	The time for processing the TUs locally in the vehicle k
$T_{j,k}^{s,pro}$	The time for processing the TUs offloaded by the vehicle k to the edge server j
T_k^s	The total latency of completing the allocated computing tasks in the cell

Access point selection

When the vehicle is moving in a cell, it may be within the coverage of multiple available APs. If the vehicle chooses AP autonomously, it may cause resource competition and network congestion. Therefore, how to select the appropriate AP for each vehicle is a problem we need to consider. The vehicle needs to select an AP from the surrounding candidates to transmit tasks. Assuming that the maximum uplink transmission rate R_{max} is limited by the bandwidth of AP. Thus, the maximal uplink rate assigned to the vehicle k through AP i is denoted as $r_{i,k}$ which can also be derived from Shannon's formula:

$$r_{i,k} = Blog_2 \left(1 + \frac{p_k h_{i,k}}{\sigma^2} \right) \tag{3}$$

where $h_{i,k}$ represents the channel gain between the vehicle k and AP i, p_k represents the transmission power of vehicle k, B represents the channel bandwidth, and σ^2 represents the transmission noise. Assuming that the maximum transmission rate remaining in the AP i is ΔR_i , if the vehicle k needs to offload task to the edge server through the AP i, $r_{i,k} \leq \Delta R_i$ should be satisfied.

When the remaining transmission rate of AP is insufficient, we need to wait for the completion of the previous transmission to release bandwidth resources to accommodate new task transmission. We assume that the waiting

time of the vehicle k due to insufficient bandwidth of AP is denoted as $T_{i,k}^{AP}$.

Task offloading

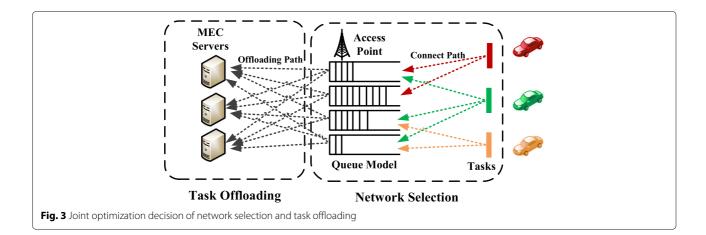
In our model, the computing task of the vehicle does not have to be offloaded to the edge server close to AP. This will reduce the load on some hot edge servers to achieve load balancing of the whole system. At the same time, it also helps to reduce the queuing delay of computing tasks executed on the edge server and improve QoS. Correspondingly, an additional communication delay is introduced due to the distance factor. We use $T_{j,k}^Q$ to denote the queuing delay of the task execution on edge server j, that is, the sum of the estimated execution time of all tasks in the queue of edge server j and the execution time of the task. $T_{i,j}^C$ indicates the communication delay from AP i to edge server j, which is given by:

$$T_{i,j}^C = \beta dis_{i,j} \tag{4}$$

where $dis_{i,j}$ represents the distance between AP i and edge server j. If the computing task of the vehicle is offloaded to the edge server near the connected AP, we have $T_{i,j}^C = 0$.

Pre-allocation algorithm in cell

Because the data size of task u_k is large and the coverage of the cell is relatively small, the whole task is separated into



TUs and would be offloaded to several edge servers in successive cells. For the goal of task completion in time, the number of computing tasks completed in each cell should be as many as possible. It is necessary to calculate the maximal number of TUs N_k^s that the vehicle can complete in the cell based on the available computing and network resources in the cell, including the maximum number of TUs that can be finished in vehicle local and the maximum number of TUs that can be completed by edge servers. It is assumed that the vehicle k is going to enter the coverage of the cell L_{cell}^s and the time that the vehicle stays in the cell (the travel time of the vehicle in the cell) can be calculated by

$$T_{stay}^{k,s} = \frac{r_s}{v_k^s} \tag{5}$$

where v_s^k represents the speed of the vehicle k in the cell. Based on the time the vehicle stays in the cell, we can calculate the maximum number of TUs processed in local through:

$$T_{stay}^{k,s} = \frac{\omega N_{loc,k}^{max,s} I_o}{f_I^k}.$$
 (6)

The number of TUs that the vehicle offloads to the edge server for processing is also limited, and $N_{off,k}^{max,s}$ stands for the maximal number of TUs processed in the cell. The value of $N_{off,k}^{max,s}$ depends on the time staying in the cell, the channel conditions in the cell and the computing capacity of edge servers. We use the average uplink rate of all APs in the cell to represent the data transmission rate in the cell:

$$r^{s} = \frac{1}{m} \sum_{i=1}^{m} Blog_2 \left(1 + \frac{p_k h_{i,k}}{\sigma^2} \right). \tag{7}$$

The average computing capacity of all edge servers in the cell is calculated by Eq. (8), and the average value is used to process offloaded tasks in the cell.

$$f_m^s = \frac{1}{n} \sum_{j=1}^n f_m^j \tag{8}$$

In order to ensure that there is no interruption during the task execution, the total time spend for the offloading task execution in edge servers must not exceed the maximum time that the vehicle stays in the cell, then we have

$$\frac{N_{off,k}^{max,s}I_o}{r^s} + \frac{\omega N_{off,k}^{max,s}I_o}{f_{s}^s} \le T_{stay}^{k,s}.$$
 (9)

Therefore, the total number of TUs that can be calculated at most in the cell includes the maximum number of TUs that the on-board device can perform (denoted as $N_{loc,k}^{max,s}$) and the maximum number of TUs that offloaded to the edge server for processing (denoted as $N_{off,k}^{max,s}$), which is expressed by

$$N_k^{max,s} = N_{loc,k}^{max,s} + N_{off,k}^{max,s}.$$
 (10)

At the same time, in order to adapt to the environment and system changes, we set a task adjustment factor θ ($\theta \in [0,1]$), which is inversely proportional to the quality of the wireless channel and the workload in the cell. The worse the wireless channel quality in the cell, or the heavier the traffic, or the larger the system workload, the smaller the value of θ . Therefore, we obtain

$$N_k^s = \left| \theta_s N_k^{max,s} \right| \tag{11}$$

where $\lfloor \rfloor$ is the floor function. Thus, the pre-allocation of tasks for each cell can be obtained, which is expressed as $N_k = \{N_k^1, N_k^2, N_k^3, ...\}$.

Adaptive task offloading strategy

The proposed adaptive task offloading consists of two stages. In the first stage, the number of TUs N_k^s assigned to the cell that needs to be completed has been solved. The

Table 2 Parameter configuration in simulation scenario

Parameter	Value
The coverage diameter of the cell (m)	[100,400]
Number of edge servers in a cell	[2,5]
Computing power of an edge server (MIPS)	[20,60]
Computing power allocated to a vehicle (MIPS)	[10,15]
On-board computing power of vehicle (MIPS)	[1,5]
Data size for each task (MB)	[50,300]
The size of task unit (MB)	0.1
Task complexity (Million Instructions/MB)	[0.1,10]
Number of access point in a cell	[2,5]

next stage aims to solve the proportion of task offloaded to edge servers for execution in the cell.

If a vehicle offloads tasks to a nearby edge server, the time cost consists of four parts: 1) the time for the vehicle to establish a wireless connection with an AP in the cell and upload the required data for the task to be processed, 2) the communication delay between the AP and the selected edge server, 3) the time the vehicle waits for the edge server to complete task queue, 4) the time when the result data is transferred back to the vehicle.

We define the decision variable named offloading ratio which is denoted as $\alpha_k^s \in [0,1]$, which means the proportion of the number of TUs offloaded to the edge server to the total number of TUs that need to be completed in the cell. We easily know that when $\alpha_k^s = 0$, TUs are all processed in the vehicle k locally; while when $\alpha_k^s = 1$, TUs that the vehicle k needed to complete in the cell are all offloaded to the edge server.

Since the maximal uplink rate assigned to the vehicle k through AP i in the cell is $r_{i,k}$, the transmission latency of

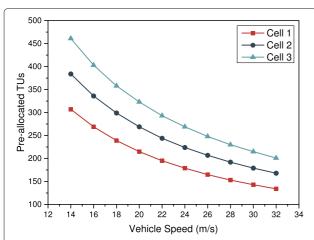


Fig. 4 The relationship between the vehicle's speed and the number of pre-allocated TUs in each cell

all offloading tasks that the vehicle k needs to complete in the cell can be given by:

$$T_{i,k}^{s,tra} = \frac{\alpha_k^s N_k^s I_o}{r_{i,k}}. (12)$$

We also define an AP selection decision variable $x_{i,k} = \{0,1\}$, and $x_{i,k} = 1$ indicates that the vehicle k is connected to the AP i for data transmission; otherwise, $x_{i,k} = 0$ means there is no connection between them. Therefore, the AP selection strategy can be expressed as $x_k = (x_{1,k}, x_{2,k}, ..., x_{m,k})$, restricted by $\sum_{i \in M} x_{i,k} = 1, \forall i \in M$.

If the accessing network is congested and there is a long queue for AP to process, the AP waiting latency will occur, which is denoted as $T_{i,k}^{AP}$. After the edge server completes the task, it will return the result to the corresponding vehicle. Generally speaking, the data volume of the result is very small, so the transmission time of the result from the edge server to the vehicle can be ignored.

The time for processing the TUs locally in the vehicle *k* can be expressed as follows:

$$T_k^{s,loc} = \frac{\omega \left(1 - \alpha_k^s\right) N_k^s I_o}{f_i^k}.$$
 (13)

The time for processing the TUs offloaded by the vehicle k to the edge server j while moving in the cell can be expressed as follows:

$$T_{j,k}^{s,pro} = \frac{\omega \alpha_k^s N_k^s I_o}{f_m^j}.$$
 (14)

We define edge server selection decision variables $y_{j,k} = \{0,1\}$, and $y_{j,k} = 0$ indicates that TUs of the vehicle k are not offloaded to edge server j; otherwise, $y_{j,k} = 1$ means TUs are offloaded to edge server j. So, the edge server selection strategy can be expressed as $y_k = (y_{i,k}, y_{2,k}, ..., y_{n,k})$ with the restriction of $\sum_{j \in N} y_{j,k} = 1$, $\forall j \in N$

Therefore, considering the queuing latency of the AP, the queuing latency of the edge server, the communication

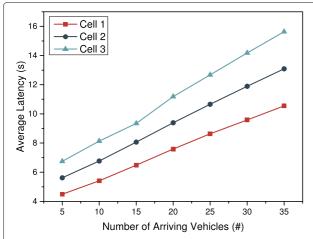


Fig. 5 The relationship between the number of arriving vehicles and the average latency of completing the pre-allocation TUs in a single cell

latency between the edge server and the AP, the offloading time to edge server *j* through AP *i*, and the execution time in edge server *j*, the total latency can be given by:

$$T_k^{s,off} = T_{i,k}^{AP} + T_{i,k}^{s,tra} + T_{i,j}^C + T_{j,k}^Q + T_{j,k}^{s,pro}.$$
 (15)

As mentioned above, the total latency of completing the allocated computing tasks in the cell is denoted as T_k^s , which can be calculated by:

$$T_k^s = max \left\{ T_k^{s,loc}, T_k^{s,off} \right\}. \tag{16}$$

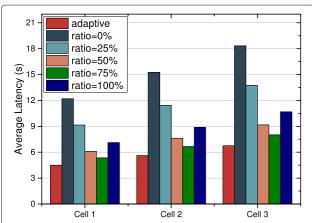


Fig. 6 The relationship between average latency to complete the pre-allocated tasks and task offloading ratio

Thus, the latency-minimization problem can be formulated as:

$$\begin{aligned} \min T_{k}^{s} &= \min \sum_{i \in M} \sum_{j \in N} \max \left\{ T_{k}^{s,off}, T_{k}^{s,loc} \right\} \\ &= \min \sum_{i \in M} \sum_{j \in N} \max \left\{ \left(x_{i,k} T_{i,k}^{AP} + x_{i,k} T_{i,k}^{s,tra} + x_{i,k} Y_{i,k}^{s,tra} + x_{i,k} Y_{i$$

By solving the optimization problem, we can derive the optimal AP, the optimal offloading edge server, the optimal offloading ratio, and determine how to offload TUs in each cell to avoid the interruption. Then, the minimum time required to complete the whole task is given by:

$$T_k = \min \sum_s T_k^s. \tag{18}$$

Solution

Now, we present the solution to the above optimization problems. It can be seen from Eq. (17) that the problem is a mixed integer programming (MIP) problem, which is to solve AP selection, task offloading, and task offloading ratio. In the solving process, we assume that the AP selection strategy is $x_k = x_k^*$ and the task offloading strategy is $y_k = y_k^*$, then the original problem becomes a convex optimization problem about α_k^s , and the optimization could be transformed into a function as follows:

$$\begin{split} f(\alpha_{k}^{s}) &= \min \sum_{i \in M} \sum_{j \in N} \max \left\{ \left(x_{i,k}^{*} T_{i,k}^{AP} + x_{i,k}^{*} T_{i,k}^{s,tra} + \right. \right. \\ &\left. x_{i,k}^{*} y_{j,k}^{*} T_{i,j}^{C} + y_{j,k}^{*} T_{j,k}^{Q} + y_{j,k}^{*} T_{j,k}^{s,pro} \right), T_{k}^{s,loc} \right\} \\ &= \min \sum_{i \in M} \sum_{j \in N} \max \left\{ \left(x_{i,k}^{*} T_{i,k}^{AP} + x_{i,k}^{*} \frac{\alpha_{k}^{s} N_{k}^{s} I_{o}}{r_{i,k}} + \right. \\ &\left. x_{i,k}^{*} y_{j,k}^{*} T_{i,j}^{C} + y_{j,k}^{*} T_{j,k}^{Q} + y_{j,k}^{*} \frac{\omega \alpha_{k}^{s} N_{k}^{s} I_{o}}{f_{m}^{j}} \right), \frac{\omega (1 - \alpha_{k}^{s}) N_{k}^{s} I_{o}}{f_{l}^{t}} \right\} \end{split}$$

$$(19)$$

We define

$$g(\alpha_k^s) = x_{i,k}^* T_{i,k}^{AP} + x_{i,k}^* \frac{\alpha_k^s N_k^s I_o}{r_{i,k}} + x_{i,k}^* \mathcal{Y}_{j,k}^* T_{i,j}^C$$

$$+ y_{j,k}^* T_{j,k}^Q + y_{j,k}^* \frac{\omega \alpha_k^s N_k^s I_o}{f_{j,n}^d}$$
(20)

$$u\left(\alpha_{k}^{s}\right) = \frac{\omega\left(1 - \alpha_{k}^{s}\right) N_{k}^{s} I_{o}}{f_{l}^{k}} \tag{21}$$

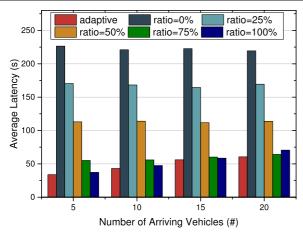


Fig. 7 The relationship between average task execution delay and task offloading ratio under different vehicle congestion conditions

Then, we have

$$f(\alpha_k^s) = \begin{cases} \min \sum_{i \in M} \sum_{j \in N} u(\alpha_k^s) \ g(\alpha_k^s) \le u(\alpha_k^s) \\ \min \sum_{i \in M} \sum_{j \in N} g(\alpha_k^s) \ g(\alpha_k^s) > u(\alpha_k^s) \end{cases}$$
(22)

It can be seen from Eq. (22) that $g\left(\alpha_k^s\right)$ and $u\left(\alpha_k^s\right)$ are linear functions of α_k^s . $g\left(\alpha_k^s\right)$ is a monotone increasing function of α_k^s , and $u\left(\alpha_k^s\right)$ is a monotone decreasing function of α_k^s . Thus, we can easily find that the optimal task offloading ratio $\alpha_k^{s,bt}$ to minimize task execution time, when the AP selection strategy is $x_k = x_k^*$ and the task offloading strategy is $y_k = y_k^*$. Since the number of APs and edge servers in a cell is limited, we can traversal all the combinations of APs and edge servers to obtain the optimal task offloading ratio in this case. Finally, through the comparison of different combinations, the optimal solution is obtained.

Simulation and result analysis

In this section, we introduce simulation scenarios, including parameter settings. Then, we analyze the impact of several important parameters and discuss the performance of the proposed scheduling scheme through simulation results.

Simulation scenarios

The simulation experiment in this paper is conducted using an edge scheduler written in Java, which can simulate the vehicle entering a series of closely adjunct cells (the coverage of the cells does not overlap each other). During the experiment, we assume vehicles entering 7 successive cells L^1_{cell} , L^2_{cell} , ... L^7_{cell} , and the diameter coverage is 100m, 120m, 150m, 230m, 200m, 250m, 310m, respectively. The number of arriving vehicles is [1,40], and the speed range is [12,34]m/s. Then, we perform the

task pre-allocation algorithm, comprehensively considering the vehicle speed, the range of each cell, the communication capability of the network access points and the computing power of the edge servers in the cell, etc., to predict the amount of tasks that the vehicle can execute in each cell. After the vehicle enters each cell, the adaptive offloading strategy is invoked, considering the load status of each network access point and MEC server in the current cell, to find the optimal network access point, offloading edge server and the optimal offloading ratio. Simulation experiment parameters are detailed in the following Table 2. The value for these parameters (except the size of task unit) is a random value in an interval. The computing power of edge server and vehicle is measured in million instructions per second (MIPS).

Result of performance evaluation

Figure 4 shows the relationship between the vehicle's speed and the number of pre-allocated TUs in three cells L^1_{cell} , L^2_{cell} , L^3_{cell} in the head. The number of pre-allocated TUs in different cells is different, because the communication and computing capacity in each cell is different. It can also be seen from Fig. 4 that as the vehicle speed increases, the number of pre-allocated TUs in each cell decrease accordingly. This is because the increase in vehicle speed will reduce the time vehicles stay in the cell, which will cause task offloading time becomes smaller. Therefore, the number of pre-allocated TUs in each cell is not static, it changes dynamically with the change of vehicle speed.

Figure 5 shows the relationship between different number of arriving vehicles and the average latency of completing the pre-allocated TUs in three cells L^1_{cell} , L^2_{cell} , L^3_{cell} in the head. The average data size of the vehicle task S_k is set to 150MB, and the average vehicle speed is 20m/s. In the process of task offloading execution, the vehicle may

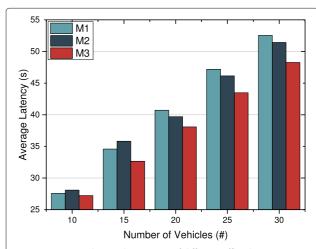


Fig. 8 Average task completion time of different offloading strategies under different traffic status

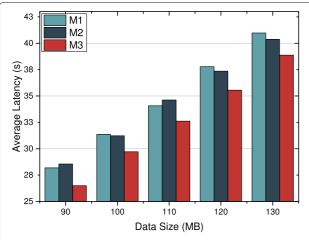


Fig. 9 Average task completion time of different offloading strategies under different task data sizes

pass through multiple cells. It can be seen in Fig. 5 that as traffic becomes heavy, the latency of completing preallocated TUs in the cell also increases. This is because with the increase in the number of vehicles entering the cells, the pressure of the APs and edge servers in the cell is also increasing, resulting in the delay of task offloading and the task queue in edge servers, which increases the total task competing latency.

Figure 6 shows when the average vehicle speed is 20 m/s, the average vehicle task size S_k is 150MB, and the vehicle passes through three cells L^1_{cell} , L^2_{cell} , L^3_{cell} , the task execution latency in the three cells in the case of different task offloading ratio. Figure 7 shows the relationship between average task execution delay and task offloading ratio under different vehicle congestion conditions when the average vehicle speed is 20 m/s and the average vehicle task size S_k is 150 MB. Through comparison we found our adaptive method is superior to all other schemes (offloading ratio is 0%, 25%, 50%, 75%, 100%), and can determine the optimal task offloading ratio to achieve minimizing latency.

In order to validate the performance of the adaptive task offloading strategy proposed in this paper, we compare our proposed strategy (M3 strategy) with the other two strategies: M1 strategy and M2 strategy.

- (1) **M1 strategy**: After task pre-allocation in the cell has been completed, when the vehicle selects the AP in the cell, the AP with the smallest waiting latency is selected to connect, and the task is offloaded to the nearby edge server, without task migrations.
- (2) **M2 strategy**: After task pre-allocation in the cell has been completed, when the vehicle selects the edge server for offloading, the edge server with the smallest queuing latency is selected for offloading, and the AP near the selected edge server is connected.

(3) **M3 strategy**: our proposed adaptive task offloading strategy.

Figure 8 shows the performance comparison of three different offloading strategies (the adaptive offloading strategy proposed in this paper and M1/M2 strategy) under different traffic situations. Figure 9 represents performance comparison under different vehicle task data sizes. Comparison results indicate that our proposed adaptive offloading strategy outperforms others.

Figure 10 presents the impact of different offloading strategies on the average task completion time under different vehicle speeds, when the number of arriving vehicles is 20 and the average vehicle task size S_k is 150MB. Through observation we can find that the strategy we proposed brings less task execution delay than others. At the same time, we can see in the figure that as the vehicle speed increases, the average task completion delay experiences a process of falling first and then rising. This is because if the vehicle speed is slow, the vehicle stays in a cell for a long time, which will cause a certain load pressure on the edge server in the cell. When the vehicle speed is relatively fast, the vehicle is traveling between different cells, which will cause frequent pre-assigned task uploads and a long waiting time.

Conclusion

This paper discusses and studies the problem of task offloading in vehicular edge computing environment. In order to solve the problem of service interruption and low QoS caused by the strong mobility of vehicles, a TUs pre-allocation algorithm in the cell has been proposed. In existing work, the influence of network access point selection on task execution latency has been often ignored. Since the access network and edge servers are

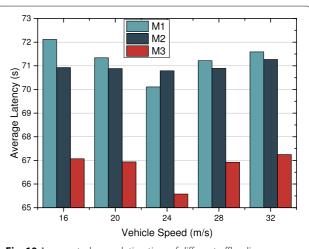


Fig. 10 Average task completion time of different offloading strategies under different vehicle speed

often to be overloaded, the commonly used task offloading method cannot guarantee the user's QoS. In this paper, we study the joint optimization of network selection and task offloading, and propose an adaptive task offloading strategy. The simulation results have proved that the proposed adaptive offloading strategy has a good performance improvement in terms of task latency and response performance of the system.

In this paper, the scenario we consider is a one-way straight road with no intersection and slow speed changes, but the actual road scene is very complicated. The vehicle may also accelerate, decelerate, stop, etc. In future work, we will consider more complex road and vehicle movements, and establish a more accurate system model, so that our proposed pre-allocation algorithm and adaptive offloading strategy can adapt to more complex road environments.

Abbreviations

MEC: Mobile edge computing; IoT: Internet of things; RAN: Radio access network; RSUs: Roadside units; AP: Access point; AR: Augmented reality; TUs: Task units; QoS: Quality of service; V2V: Vehicle-to-vehicle; MDP: Markov decision process; MIP: Mixed integer programming; MIPS: Million instructions per second

Acknowledgements

The authors would like to thank all anonymous reviewers for their invaluable comments.

Authors' contributions

This paper is completed under the supervision of author Bing Tang. Lujie Tang wrote the paper. Feiyan Guo is responsible for the technical architecture design, Lujie Tang is responsible for the experiment, and Li Zhang is responsible for the images. The grammar of the paper was reviewed and modified by Haiwu He. Finally, Bing Tang gives some modification suggestions. All authors have read and approved the final manuscript.

Funding

This work was supported by the National Natural Science Foundation of China under grant no. 61872138 and 61602169, the National Key R&D Program of China under grant no. 2018YFB1402800, and the Natural Science Foundation of Hunan Province under grant no. 2018JJ2135, as well as the Scientific Research Fund of Hunan Provincial Education Department under grant no. 18A186.

Availability of data and materials

The data used to support the findings of this study are available from the corresponding author upon request.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

¹ School of Computer Science and Engineering, Hunan University of Science and Technology, 411201 Xiangtan, China. ² Shandong Computer Science Center (National Supercomputer Center in Jinan), 250014 Jinan, China.

Received: 31 July 2020 Accepted: 23 February 2021 Published online: 09 March 2021

References

 Mach P, Becvar Z (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun Surv Tutor 19(3):1628–1656

- Tran TX, Hajisami A, Pandey P, Pompili D (2017) Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. IEEE Commun Mag 55(4):54–61
- 3. Ksentini A, Frangoudis PA (2020) Toward slicing-enabled multi-access edge computing in 5G. IEEE Netw 34(2):99–105
- Schwab J, Hill A, Jararweh Y (2020) Edge computing ecosystem support for 5G applications optimization. In: Pillai P, Lv Q (eds). Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications. ACM, New York. p 103. https://doi.org/10.1145/3376897. 3379166
- Boukerche A, Grande RED (2018) Vehicular cloud computing: Architectures, applications, and mobility. Comput. Networks 135:171–189
- Raza S, Wang S, Ahmed M, Anwar MR (2019) A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions. Wirel Commun Mob Comput 2019:3159762. https://doi.org/ 10.1155/2019/3159762
- Wang H, Li X, Ji H, Zhang H (2018) Dynamic offloading scheduling scheme for MEC-enabled vehicular networks. In: 2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops). IEEE, New York. pp 206–210. https://doi.org/10.1109/ICCChinaW.2018.8674508
- 8. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Internet Things J 3(5):637–646
- Shaukat U, Ahmed E, Anwar Z, Xia F (2016) Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges. J Netw Comput Appl 62:18–40
- Stojmenovic I, Wen S (2014) The fog computing paradigm: Scenarios and security issues. In: Proceedings of the 2014 Federated Conference on Computer Science and Information Systems Vol. 2. pp 1–8. https://doi. org/10.15439/2014F503
- Ahmed E, Rehmani MH (2017) Mobile edge computing: Opportunities, solutions, and challenges. Futur Gener Comput Syst 70:59–63
- 12. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: The communication perspective. IEEE Commun Surv Tutor 19(4):2322–2358
- Liu Y, Peng M, Shou G, Chen Y, Chen S (2020) Toward edge intelligence: Multiaccess edge computing for 5G and internet of things. IEEE Internet Things J 7(8):6722–6747
- Wan S, Li X, Xue Y, Lin W, Xu X (2020) Efficient computation offloading for internet of vehicles in edge computing-assisted 5G networks. J Supercomput 76(4):2518–2547
- Zhang K, Mao Y, Leng S, Vinel AV, Zhang Y (2016) Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks. In: 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM). IEEE, New York. pp 288–294
- Zhang K, Mao Y, Leng S, Maharjan S, Zhang Y (2017) Optimal delay constrained offloading for vehicular edge computing networks. In: 2017 IEEE International Conference on Communications (ICC). IEEE, New York. pp 1–6. https://doi.org/10.1109/ICC.2017.7997360
- Zhang K, Mao Y, Leng S, Zhao Q, Li L, Peng X, Pan L, Maharjan S, Zhang Y (2016) Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access 4:5896–5907
- Hao Y, Chen M, Hu L, Hossain MS, Ghoneim A (2018) Energy efficient task caching and offloading for mobile edge computing. IEEE Access 6:11365–11373
- Zhang J, Hu X, Ning Z, Ngai ECH, Zhou L, Wei J, Cheng J, Hu B (2018) Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks. IEEE Internet Things J 5(4):2633–2645
- Tran TX, Pompili D (2019) Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Trans Veh Technol 68(1):856–868
- Tang W, Zhao X, Rafique W, Qi L, Dou W, Ni Q (2019) An offloading method using decentralized P2P-enabled mobile edge servers in edge computing. J Syst Archit 94:1–13
- Wu H, Wolter K (2015) Software aging in mobile devices: Partial computation offloading as a solution. In: 2015 IEEE International Symposium on Software Reliability Engineering Workshops. IEEE Computer Society, New York. pp 125–131. https://doi.org/10.1109/ ISSREW.2015.7392057
- 23. Kiani A, Ansari N (2018) Optimal code partitioning over time and hierarchical cloudlets. IEEE Commun Lett 22(1):181–184

- Mazza D, Tarchi D, Corazza GE (2014) A partial offloading technique for wireless mobile cloud computing in smart cities. In: European Conference on Networks and Communications. IEEE, New York. pp 1–5. https://doi. org/10.1109/EuCNC.2014.6882623
- Wang S, Urgaonkar R, Zafer M, He T, Chan KS, Leung KK (2015) Dynamic service migration in mobile edge-clouds. In: Proceedings of the 14th IFIP Networking Conference. IEEE Computer Society, New York. pp 1–9. https://doi.org/10.1109/IFIPNetworking.2015.7145316
- Alasmari KR, II RCG, Alam M (2018) Mobile edge offloading using Markov decision processes. In: 2018 International Conference on Edge Computing (EDGE). Springer, Switzerland. pp 80–90. https://doi.org/10. 1007/978-3-319-94340-4 6
- Sun Y, Zhou S, Xu J (2017) EMM: energy-aware mobility management for mobile edge computing in ultra dense networks. IEEE J Sel Areas Commun 35(11):2637–2646
- Gao B, Zhou Z, Liu F, Xu F (2019) Winning at the starting line: Joint network selection and service placement for mobile edge computing. In: 2019 IEEE Conference on Computer Communications. IEEE, New York. pp 1459–1467. https://doi.org/10.1109/INFOCOM.2019.8737543
- Zheng K, Liu F, Zheng Q, Xiang W, Wang W (2013) A graph-based cooperative scheduling scheme for vehicular networks. IEEE Trans Veh Technol 62(4):1450–1458
- Yu R, Ding J, Huang X, Zhou M, Gjessing S, Zhang Y (2016) Optimal resource sharing in 5G-enabled vehicular networks: A matrix game approach. IEEE Trans Veh Technol 65(10):7844–7856
- Sardellitti S, Scutari G, Barbarossa S (2015) Joint optimization of radio and computational resources for multicell mobile-edge computing. IEEE Trans Signal Inf Process over Netw 1(2):89–103
- Deng M, Tian H, Lyu X (2016) Adaptive sequential offloading game for multi-cell mobile edge computing. In: 23rd International Conference on Telecommunications. IEEE, New York. pp 1–5. https://doi.org/10.1109/ICT. 2016.7500395
- Poularakis K, Llorca J, Tulino AM, Taylor I, Tassiulas L (2019) Joint service placement and request routing in multi-cell mobile edge computing networks. In: 2019 IEEE Conference on Computer Communications. IEEE, New York. pp 10–18. https://doi.org/10.1109/INFOCOM.2019.8737385
- Zhang J, Chen B, Zhao Y, Cheng X, Hu F (2018) Data security and privacy-preserving in edge computing paradigm: Survey and open issues. IEEE Access 6:18209–18237
- Qu X, Hu Q, Wang S (2020) Privacy-preserving model training architecture for intelligent edge computing. Comput Commun 162:94–101
- Yang R, Yu FR, Si P, Yang Z, Zhang Y (2019) Integrated blockchain and edge computing systems: A survey, some research issues and challenges. IEEE Commun Surv Tutor 21(2):1508–1532
- Gai K, Wu Y, Zhu L, Xu L, Zhang Y (2019) Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. IEEE Internet Things J 6(5):7992–8004
- Kang J, Yu R, Huang X, Wu M, Maharjan S, Xie S, Zhang Y (2019)
 Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet Things J 6(3):4660–4670
- He X, Jin R, Dai H (2019) Physical-layer assisted privacy-preserving offloading in mobile-edge computing. In: 2019 IEEE International Conference on Communications. IEEE, New York. pp 1–6. https://doi.org/ 10.1109/ICC.2019.8761166
- Chai H, Leng S, Chen Y, Zhang K (2020) A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles. IEEE Trans Intell Transp Syst:1–12. https://doi.org/10.1109/TITS. 2020.3002712

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ▶ springeropen.com