Raza *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:28
https://doi.org/10.1186/s13677-020-00175-w

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

An efficient task offloading scheme in vehicular edge computing

Salman Raza[1], Wei Liu[2*], Manzoor Ahmed[3], Muhammad Rizwan Anwar[1], Muhammad Ayzed Mirza[4], Qibo Sun[1] and Shangguang Wang[1]

Abstract

Vehicular edge computing (VEC) is a promising paradigm to offload resource-intensive tasks at the network edge. Owing to time-sensitive and computation-intensive vehicular applications and high mobility scenarios, cost-efficient task offloading in the vehicular environment is still a challenging problem. In this paper, we study the partial task offloading problem in vehicular edge computing in an urban scenario. Where the vehicle computes some part of a task locally, and offload the remaining task to a nearby vehicle and to VEC server subject to the maximum tolerable delay and vehicle's stay time. To make it cost-efficient, including the cost of the required communication and computing resources, we consider to fully exploit the vehicular available resources. We estimate the transmission rates for the vehicle to vehicle and vehicle to infrastructure communication based on practical assumptions. Moreover, we present a mobility-aware partial task offloading algorithm, taking into account the task allocation ratio among the three parts given by the communication environment conditions. Simulation results validate the efficient performance of the proposed scheme that not only enhances the exploitation of vehicular computation resources but also minimizes the overall system cost in comparison to baseline schemes.

Keywords: Vehicular edge computing, Task offloading, Mobility, Mobile edge computing, Vehicular networks

Introduction

As an enabling technology for the Internet of Vehicles, Vehicular edge computing (VEC) provides possible solutions to share the computation capabilities between vehicles. The continuous increase in mobile applications has caused an exponential growth in demand for high computational capability in wireless networks [1]. Vehicles are equipped with computing and storage resources to support an intelligent transport system and a wide variety of onboard infotainment services. It is predicted that by 2022, every self-driving car will have the computing capability to execute up to 106 dhrystone million instructions per second [2], which is ten times that of the existing laptops. However, vehicle's data demands and computation requirements are also increasing day by day caused by innovative safety and non-safety applications, i.e., augmented reality, virtual reality, and immersive & real-time interactive applications. To cope with all such evolving communication and computation demands of vehicular systems, mobile devices, and pedestrians, this is where the VEC system came into existence [3, 4]. Vehicular and infrastructural nodes, i.e., the roadside units (RSUs), can make their communication and computational resources available to the network.

In contrast to cloud infrastructure that may induce considerable overhead in terms of delay between the cloud and vehicles to offload the computation task [5, 6]. Since cloud resources are usually deployed at the remote end [7]. VEC solves this problem, as it brings these computational capabilities in close proximity and enables numerous vehicles to process their required tasks at the network edge [8]. The network latency can be minimized to a great extent while accessing edge computing resources by the proximity of mobile vehicles. This enables VEC to offer a

*Correspondence: liuw@bupt.edu.cn
[2]Software Engineering, Beijing University of Posts and Telecommunications, Beijing, China, 100876 Beijing, China
Full list of author information is available at the end of the article

prompt interactive response in the computation offloading service by deploying computing nodes or servers, to fulfill users' demands for delay-sensitive tasks [9]. Computation offloading is a promising way of transferring some computation-intensive activities to nearby servers [10, 11]. Moreover, VEC takes advantage of proximal vehicles, to reduce the edge servers' load. Hence, vehicles can also perform computational tasks for VEC servers.

In addition, vehicles' collaboration is enabled via infrastructure or infrastructure-independent communication using dedicated short-range communication (DSRC) or cellular vehicle to everything technology, i.e., vehicle to vehicle (V2V) communication. V2V communication facilitates both safety and non-safety applications [12]. In V2V communication, the vehicles share data that can define the internal state and environment of the vehicle to widen the perceptual horizon of the communication partner. Any relevant information gathered from vehicular onboard sensors can be forwarded to nearby vehicles [13]. Apart from sharing information, nearby vehicles can help in processing high computational tasks, which cannot be tackled alone by a computational resource-limited vehicle. The vehicles communicate directly with each other if they are within their communication range. Among the various scenarios, mobility and computation offloading that are mainly followed within the bound of the network edge, are based on the internet of vehicles [14]. However, still, vehicles' computation capability is limited to fully handle the existing and emerging low latency applications' computational demands.

Related work

VEC is becoming an eminent trend. Many researchers have contributed to figure out the challenges that VEC poses [15, 16]. For offloading the computation tasks to the network edge, the VEC emerged as a new paradigm to lighten the computation load of vehicles with limited resources and satisfies real-time responses to vehicles' task requests [17]. In [18], the authors presented a contract-based mechanism for the allocation of resources by exploiting mobile edge computing (MEC) servers' resources and fulfilling the offloading requisites of the tasks.

However, the computation and storage capacity of edge computing is still inadequate. Therefore, some hybrid schemes have been proposed, which integrate the advantages of both edge computing and vehicular networks. For instance, Hou et al. [16], analyzed the use of both moving and parked vehicles as computation and communication platforms to improve service quality. Ye et al. [19] presented a scheme to offload mobile devices and cloudlets to fog enabled bused at low energy and transmission costs. While Feng et al. [20] put forth a hybrid cloud computing infrastructure in vehicular networks, where tasks are offloaded to other vehicles in the vicinity or to the RSUs. Similarly, the authors in [21] opted for a Stackelberg game-theoretic scheme to develop a multilevel offloading framework and presented a hierarchically organized cloud-based VEC offloading scheme, in which a backup computing neighborhood is there to support the computing resources of MEC servers. Different from [16, 20], and [21], Lai et al. [22], proposed a three-tier vehicular network that includes the cloud layer, the fog layer, and the network layer. The authors developed cooperation and scheduling schemes to manage the vehicle nodes. Ren et al. [23], developed a partial compression offloading framework, where a small part of the data is computed locally on the vehicle while the remaining part of it is computed on the MEC server. This allows the exploitation of local and MEC computation resources efficiently. With the integration of V2V and vehicle to infrastructure (V2I) communication, the authors in [24] came up with a framework to offload a load of vehicles with a low signal-to-noise-and-interference ratio to be served by other vehicles with a greater quality link. The authors in [25] and [26], presented a predictive combination-mode and load-aware MEC offloading schemes, respectively, in which tasks are offloaded via V2V relay transmission to the MEC server or V2I uploading.

Bozorgchenani et al. [27], analyzed a partial offloading method. Where the amount of the task to be offloaded is estimated for reducing the outage probability considering the vehicles' mobility in an urban environment. The entire process of task offloading must be less than the stay time of the vehicles. However, we consider offloading the amount of task to a nearby vehicle, which can be transmitted within the stay time by meeting the maximum tolerable delay of a task. This will help to utilize the available resources of the vehicles while dividing the task accordingly and sharing the burden of VEC sever. In [28], the authors focused on the federated offloading in vehicular networks to reduce the total latency. The computation task is divided into three parts, i.e., to compute locally send to nearby vehicles, and on the VEC server. The authors assign the computing ratios to ensure and keep the whole task computed within the given latency deadline. However, this scheme does not fully use the vehicular resources as it prefers assigning most of the task part on the VEC server. Whereas, we proposed to exploit the vehicle resources as well as ease the load of the VEC server.

In this paper, we focus on task offloading leveraging V2V (among vehicles) and V2I (between vehicle and VEC servers) communication. Since both the vehicles and VEC servers, are equipped with computation resources, therefore, considering such a hybrid network in a dense urban scenario can further boost the network's communication and computing capacity. As we have considered

partial task offloading, therefore, part of the computation is handled locally, while the remaining task is offloaded to nearby vehicles and the VEC server. We proposed a mobility-aware partial task offloading approach enabling the cost-efficient system. In our proposed scheme, we consider two types of vehicles, i.e., resource-hungry vehicles (RHV) and resource-rich vehicles (RRVs). As their name implies, RHV always tries to offload its task as having limited computing resources, while RRV has abundant resources and helps RHVs in computation. It is pertinent to mention here that the task offloading decisions are determined by each RHV. Multiple RRVs might be available to process each task of RHV. Thus, our proposed scheme helps in offloading the VEC server burden by exploiting the underutilized resources of RRVs. Considering the vehicular network dynamic nature, the wireless channel condition and network topology change hastily due to the incessant vehicles' mobility [29]. Furthermore, the computation workloads of available RRVs vary over time. Thus, taking into account the above factors, we then present a partial task offloading algorithm in which, apart from local computation, RHVs' assign priority to RRVs' selection while making a decision; and RRVs are assigned the maximum portion of a task as it comprises fewer communication and computation costs. This scheme fully utilizes the computation resources of RRVs and reduces the overall burden of the VEC server. Hence, the overall system's cost will also be minimized.

Motivation and contributions

Many works have been done on task offloading. The following schemes are limited in several aspects:

- In [19] the scheme is centralized where mobile users first send their tasks to RSUs, then RSU decides to compute according to its load or gives the task to resourceful vehicles for computation.
- The works [18, 19, 21, 25, 26] considered binary offloading, which do not guarantee the full utilization of vehicles' computational resources.
- The studies [25] and [26] are using vehicular communication resources but eventually put the computation load on the MEC server.
- In [27], the allocated portion of the task for the nearby vehicle depends upon the stay time of the vehicle to reduce the outage probability.
- In [28], the scheme does not fully utilize vehicular resources as it prefers the MEC server.

Our proposed work aims to fill the above-mentioned gaps. More specifically, the main contributions of this paper are listed below:

1. We model a task offloading scheme to minimize the overall offloading cost. This model is utilized to create a realistic vehicular environment to study the task offloading problem in a large-scale network. Where the task is computed partially at the source vehicle, and the maximum part of the remaining task is offloaded and computed first at the proximity vehicles and then at the relevant VEC server. This enables not only the exploitation of abundant vehicles' resources and reduction in the overburdened VEC server's load but also slashes the overall system cost.
2. We propose a mobility-aware partial task offloading algorithm in the VEC scenario. This allows each vehicle to select its nearby vehicles based on the best available resources with minimum cost. Moreover, we consider practical assumptions and estimate the transmission rates for V2V and V2I communication. Based on that the proportion of a task to be computed locally, on a nearby vehicle, and at the VEC, is calculated conditional to the maximum tolerable delay and vehicle's stay time.
3. We evaluate the influence of different parameters and vehicular environments on our mobility-aware partial task offloading scheme by comparing it with different strategies. We use extensive simulations to validate the effectiveness of our proposed solution.

The remaining parts of this paper are organized as follows. The "System model" section holds the system model and the problem's formal definition is discussed in the "Problem formulation" section. The proposed algorithm is presented in the "Mobility-Aware partial (MAP) task offloading algorithm" section. In "Results and discussions" section the implementation and evaluation of our MAP algorithm are presented. Finally, the "Conclusion" section concludes this paper.

System model

In this section, we first describe the network topology followed by the communication model's description. Then we present the computation model. All the notations used in the system model are listed in Table 1.

Network topology

Figure 1 shows our proposed mobility-aware partial task offloading in VEC. A unidirectional road is considered, where the RSUs are installed along the road like a typical case for vehicular networks. A VEC server is also installed with each RSU. We refer the vertical distance from the RSU to the road by e. Each RSU unit has a communication range, i.e., a radius of 200 meters. The set of vehicles having the task to be offloaded is defined as $N = \{1, , n\}$. Considering the heterogeneity of vehicles, each vehicle has a distinct set of computational resources. Vehicles can offload their tasks to the RSUs. Additionally,

Table 1 Frequently Used Notations

Term	Description
α	The portion of a task for local computation
β	The portion of a task for V2V computation
γ	The portion of a task for VEC computation
$t_{n,stay}^{V2I}$	The stay time of a vehicle RSU coverage
$t_{n,stay}^{V2V}$	The time of the V_n in the coverage of the V_i
T_n^{Local}	The total time of local computing
T_n^{V2V}	The total time of V2V computing
T_n^{VEC}	The total time of VEC computing
C_n^{Local}	The total cost of local computing
C_n^{V2V}	The total cost of V2V computing
C_n^{VEC}	The total cost of VEC computing
ψ_{V2V}	V2V transmission cost
ψ_{V2I}	V2I transmission cost
Φ_{local}	The cost for utilizing local computation.
Φ_{V2V}	The cost for utilizing V2V computation.
Φ_{VEC}	The cost for utilizing VEC computation.

if a complex computation task is handed over to RSU, then it will be computed on the VEC server. A central controller is installed in the network that monitors and manages the RSUs [18]. Many vehicles traverse under the coverage of each RSU, which we classify into two categories, i.e., RHV and RRV. As its name implies, RHV represents the vehicle that has a computation task to offload. Since the coverage radius of RSU r and the vertical distance from the RSU to road e is known, we can easily find the distance of vehicles traveling within the RSU coverage as:

$$s_n = 2\sqrt{r^2 - e^2}. \tag{1}$$

Accordingly, the stay time of the vehicle within RSU coverage is derived as:

$$t_{n,stay}^{V2I} = \frac{s_n}{v_n}, \tag{2}$$

where the parameter v_n denotes the speed of vehicle V_n.

Communication model

The communication model comprises of V2V communication and V2I communication, which are discussed as follows.

V2V communication

In V2V communication, the vehicles interact with each other according to the standard of DSRC [30]. The maximum communication range of V2V is expressed as C_{limit}. We choose an identically and independent distribution channel among vehicles. The path loss of V2V communication is determined as [24]:

$$L_n^{V2V} = 10^{-\frac{63.3+17.7log_{10}(d_{n,i})}{10}}, \tag{3}$$

where the $d_{n,i}$ defines the distance between V_n and V_i. It must satisfy the condition $0 \leq d_{n,i} \leq C_{limit}$.

In our scenario, the vehicles' speed may not be the same. Therefore, vehicles have a relative speed between them. We denote the speed between the V_n and V_i as $v_{n,i}$. The vehicle's bandwidth is specified as B_{V2V} and orthogonal frequency is usually chosen for V2V communication. Accordingly, the transmission rate between vehicle V_n and V_i is computed as:

$$R_{V2V}^{n,i} = B_{V2V} log_2 \left(1 + \frac{P_t L_n^{V2V} |h^2|}{N_0}\right). \tag{4}$$

Moreover, we need to evaluate the duration of time for which the vehicle V_n stays within the coverage of the vehicle V_i to avoid offloading failure when the vehicle V_i is not within the coverage area. The rest of the distance before leaving the coverage of the vehicle V_i at time t, can be expressed as follow [27].

$$d_{n,i}(t) = \sqrt{r_i^2 - (x_i(t) - x_n(t))^2} \pm (y_i(t) - y_n(t)), \tag{5}$$

where $\{x_n(t), y_n(t)\}$ and $\{(x_i(t), (y_i(t)\}$ are the position of the vehicle V_n and vehicle V_i, respectively, at time t. While r_i is the radius of the vehicle V_i coverage area. Accordingly, the time that the vehicle V_n remains in the coverage area of the vehicle V_i can be defined as:

$$t_{n,stay}^{V2V} = \frac{d_{n,i}(t)}{|\overrightarrow{v}_n - \overrightarrow{v}_i|}, \tag{6}$$

where $|\overrightarrow{v}_n - \overrightarrow{v}_i|$ is the vector speeds of the vehicles V_n and V_i in view of their relative direction. Thus, the uplink rate $R_{V2V}^{n,i}$ changes with time that can be defined as $R_{V2V}^{n,i}(t)$. Therefore, the average uplink rate between vehicle V_n and V_i is given as:

$$\overline{R_{V2V}^{n,i}} = \frac{\int_0^{t_{n,stay}^{V2V}} R_{V2V}^{n,i}(t)dt}{t_{n,stay}^{V2V}}. \tag{7}$$

V2I communication

Unlike the V2V communication that uses DSRC technology, we leverage LTE-A for V2I communication between vehicles and RSU [30]. The parameter $d_{n,rsu}$ is the distance between vehicle V_n and the center of coverage of the RSU. The path loss of the vehicle V_n and its proximal RSU can be represented as $d_{n,rsu}^{-\sigma}$ and the white Gaussian noise power as N_0. The factor σ is the path loss exponent [31]. Furthermore, the uplink channel is modeled as the Rayleigh fading channel denoted as h [28]. Hence, the

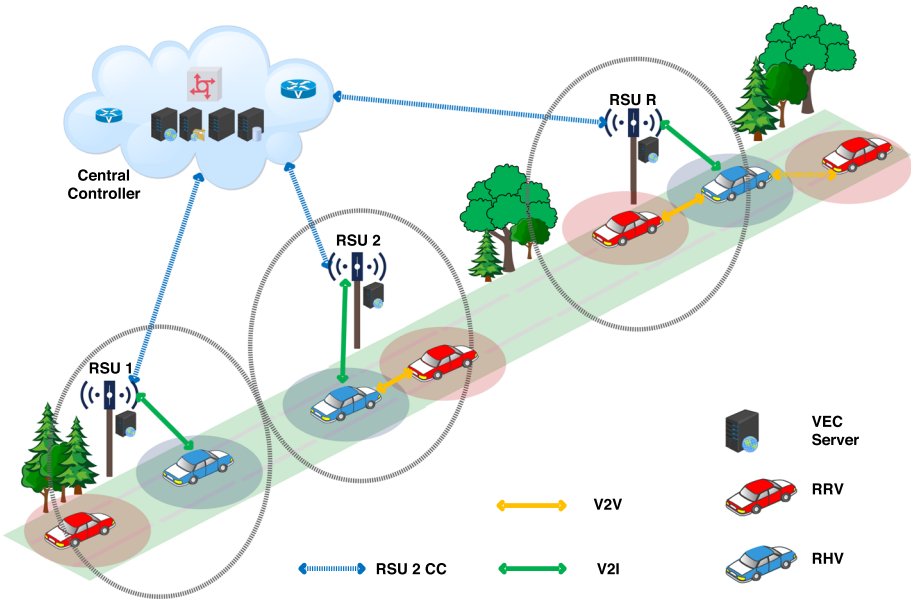

Fig. 1 Task offloading scenario in Vehicular Edge Computing

uplink data rate is defined as:

$$R_{V2I} = B_{V2I} log_2 \left(1 + \frac{P_t d_{n,rsu}^{-\sigma} |h^2|}{N_0}\right), \tag{8}$$

where the parameter B_{V2I} represents the uplink channel bandwidth, and P_t denotes the transmission power of the vehicle's onboard device.

In our scenario, vehicles travel at a constant speed. Therefore, the distance $d_{n,rsu}$ varies with time, which is given by

$$d_{n,rsu}(t) = \sqrt{e^2 + \left(\frac{s_n}{2} - v_n^{abs} t\right)^2}, \tag{9}$$

where v_n^{abs} is the speed of the vehicle V_n. Accordingly the uplink rate R_{V2I} varies with time as well that can be define as $R_{V2I}(t)$. The V2I average uplink rate is defined as:

$$\overline{R_{V2I}} = \frac{\int_0^{t_{n,stay}^{V2I}} R_{V2I}(t) dt}{t_{n,stay}^{V2I}}. \tag{10}$$

$\overline{R_{V2I}}$ is the V2I average uplink rate represented as the uplink rate of V_n offloading a task to the VEC server.

Computation model

We have assumed that the vehicle V_n has a computing task, described as $R_n = \{B_n, D_n, t_n^{max}\}$. Here B_n indicates the total number of required CPU cycles to carry out the task, D_n shows the task data size, which includes the input parameters and program code and the t_n^{max} indicates the maximum tolerable delay of the task R_n, which implies the time to complete the task should not exceed t_n^{max}. The task is divided into three parts: computed at the vehicle V_n locally, offloading the remaining task to nearby vehicles V_i by V2V communication, and the final remaining task is offloaded to the nearest VEC server for computation. The ratio of data to total task data D_n is denoted as α_n, β_n, and γ_n, respectively. The different ratio will influence the total latency and cost to finish the task. Since computation units are installed in vehicles, the tasks could be computed on the nearby RRV. The computation ability might change from vehicle to vehicle. Therefore, we specify the computation capacity of V_i as fV_i . In order to improve the utilization of computing resources, we present the V2V offloading method. In other words, the task of vehicle V_n might be offloaded to its nearby qualified vehicle. Moreover, the priority of each vehicle is to offload the part of the task to its nearby qualified vehicle as much as possible, according to available computing capacity, and the final remaining part to the VEC server.

To further elaborate on the computation model in detail. We introduce the local computing followed by nearby vehicle computing. and finally, the VEC computing is presented.

Local computing

When the source vehicle V_n chooses to perform task R_n locally, T_n^{Local} is defined as the local execution delay of the vehicle V_n, includes the local CPU processing delays. The fV_n is described as the computation capacity (i.e., CPU cycles per second) of V_n. Considering the heterogeneity of the vehicles, different vehicles might have different capacities for computation. The local execution delay of task R_n

is given as:

$$t_n^l = \frac{B_n}{fV_n}, \tag{11}$$

where α_n is the portion of the task D_n, which is computed locally as:

$$T_n^{Local} = \alpha_n * t_n^l. \tag{12}$$

The Φ_{local} is the cost for local computation. Taking into account the above mentioned time consumption, the total cost for local computing can be specified as:

$$C_n^{Local} = \Phi_{local} * T_n^{Local}. \tag{13}$$

Nearby vehicle computing

The selected RRV vehicle V_i processes the task and generates the output after fetching the input data from the Vehicle V_n. The computation intensity of the task mainly relies on the nature of applications. The V2V offloading latency consists of task execution and transmission time. The transmission time from V_n to the vehicle V_i is represented as $t_{up}^{V_i}$, which can be defined as:

$$t_{n,up}^{V_i} = \frac{D_n}{R_{V2V}^{n,i}}, \tag{14}$$

where the computation capacity of the nearby vehicle is defined as fV_i. The execution time of vehicle V_i can be defined as:

$$t_{n,ex}^{V_i} = \frac{B_n}{fV_i}. \tag{15}$$

We represent the total offloading latency (i.e., execution and transmission time) from V_n to V_i as T_n^{V2V} , which can be expressed as:

$$T_n^{V2V} = \beta_n * t_{n,up}^{V_i} + \beta_n * t_{n,ex}^{V_i}, \tag{16}$$

where β_n is the portion of the task D_n. We need to check all the vehicles computation resources then we select the qualified vehicle and its detail will follow in "Results and discussions" section. Each of the task has a unique memory and processing power besides has a specified cost of using per unit of time [32]. Therefore, by considering the aforementioned time consumption, the total cost of V2V computing can be defined as:

$$C_n^{V2V} = \left\{\psi_{V2V} * (\beta_n * t_{n,up}^{V_i})\right\} + \left\{\Phi_{V2V} * (\beta_n * t_{n,ex}^{V_i})\right\}, \tag{17}$$

where ψ_{V2V} is the transmission cost and Φ_{V2V} is the V2V computation cost.

VEC computing

The VEC offloading latency comprises of three-parts: the latency to transmit the data to its nearest VEC server, ready time of task on the VEC server, and the execution time on the VEC server. Regarding the delay in transmitting the result back, we tend to neglect it following the footsteps of given references [31, 33]. The latency for transmitting the data to the VEC server can be given by,

$$t_{n,up}^{VEC} = \frac{D_n}{R_{V2I}}. \tag{18}$$

Vehicle V_n offloads the remaining part of the task to the nearest VEC server via the wireless link. During transmission, the vehicle V_n must be within the coverage area of connected RSU. Specifically, the transmission time from vehicle V_n to the VEC server $t_{n,up}^{VEC}$ must be shorter than the time that vehicle V_n is in coverage of its connected RSU. It can be defined as:

$$t_{n,up}^{VEC} \leq t_{n,stay}^{V2I}. \tag{19}$$

The computation capacity of the VEC server is denoted as f_m (i.e., CPU cycles per second). Therefore, the execution time $t_{n,ex}^{VEC}$ on the VEC server can be calculated as follow:

$$t_{n,ex}^{VEC} = \frac{B_n}{f_m}. \tag{20}$$

Further, we define the ready time of a task according to [34].

Definition 1 (Ready Time). *The ready time of a task can be expressed as the time at which all the predecessors of the task finished their execution. Therefore, the ready time of task R_n of Vehicle V_n in VEC computing is represented by RT_{n,R_n}^{VEC}.*

$$RT_{n,R_n}^{VEC} = \max_{k \in pred(R_n)} t_{n,ex,k}^{VEC}, \tag{21}$$

where pred(R_n) refers to a set of predecessors for task R_n. Therefore, $\max_{k \in pred(R_n)} t_{n,ex,k}^{VEC}$ is the time when the predecessors of task R_n that were offloaded to the VEC have completely performed their execution on the VEC server. Moreover, we can identify that the VEC can begin the execution of task R_n only after the task has been fully offloaded to VEC and all the predecessors of task R_n have completed their execution on the VEC.

The total latency for VEC offloading T_n^{VEC} is the sum of the uplink transmission time from vehicle V_n to VEC server $t_{n,up}^{VEC}$, ready time RT_{n,R_n}^{VEC} and execution time $t_{n,ex}^{VEC}$. Hence, the total latency for VEC offloading T_n^{VEC} can be defined as:

$$T_n^{VEC} = \gamma_n * t_{n,up}^{VEC} + \quad RT_{n,R}^{VEC} + \gamma_n * t_{n,ex}^{VEC}, \tag{22}$$

as mentioned above, many conditions and constraints will affect the latency T_n^{VEC}, for instance, the vehicle speed and the ratio γ_n. When the VEC server finishes the computing, the output results will be sent back to the vehicle V_n. We neglect the transmission time from the VEC server to V_n, since the amount of data as compared to input is very little [33].

The cost is evaluated by the utilization of the processor. The longer the utilization time is, the higher the cost is [35]. By considering the above time consumption, the total cost of VEC computing can be computed as:

$$C_n^{VEC} = \left\{\psi_{V2I} * \left(\gamma_n * t_{n,up}^{VEC}\right)\right\} + \left\{\Phi_{VEC} * \left(RT_{n,R_n}^{VEC} + (\gamma_n * t_{n,ex}^{VEC})\right)\right\}, \quad (23)$$

where γ_n is the portion of the task D_n, the ψ_{V2I} is the transmission cost, and the Φ_{VEC} is the cost of both ready time and execution time on the VEC server. Thus, the total cost to complete a task is denoted as C_n:

$$C_n = \left\{C_n^{Local} + C_n^{V2V} + C_n^{VEC}\right\}. \quad (24)$$

Moreover, the total cost of whole system can be derived as:

$$C_{Total} = \sum_{n=1}^{N} \left\{C_n^{Local} + C_n^{V2V} + C_n^{VEC}\right\}. \quad (25)$$

Problem formulation

In this section, we formulate the partial task offloading as an optimization problem. The aim is to minimize the total offloading cost satisfying constraints like the limits of maximum tolerable delay and computational capacity. The optimization problem is written as follows:

$$\mathbf{P1}: \min_{(\alpha_n,\beta_n,\gamma_n)} C_{Total}$$

$$s.t.\ \ \alpha_n + \beta_n + \gamma_n = 1 \quad (26a)$$

$$\max\{T_n^{Local}, T_n^{V2V}, T_n^{VEC}\} \le t_n^{max}, \quad (26b)$$

$$0 \le f_{V_i} \le F_n^{V_i}, 0 \le f_m \le F_n^{VEC}, \forall n \in N \quad (26c)$$

$$t_{n,up}^{VEC} \le t_{n,stay}^{V2I}, \quad (26d)$$

$$t_{n,up}^{V_i} \le t_{n,stay}^{V2V}. \quad (26e)$$

It is reiterated that our optimization goal is to minimize the total cost. Here, the constraint (26a) is the relationship among α_n, β_n, and γ_n. (26b) indicates that the time of local, nearby, and VEC server offloading should not exceed the maximum tolerable delay. (26c) shows that the computing resource assigned for the vehicle V_n cannot surpass the total resource $F_n^{V_i}$ of the nearby vehicle as well as VEC server, respectively. (26d) specifies that the task for the V2I part should be transmitted completely to the VEC server before the vehicle V_n runs out of the communication range. (26d) indicates that the task for the V2V part should be transmitted completely before the vehicle V_n runs out of the communication range of V_i.

Mobility-Aware partial (MAP) task offloading algorithm

In the V2V network, the global information of vehicles may not be available or cost too much. Besides, it is tough to obtain multi-hop information by a vehicle because of the maximum communication limit constraint as it also increases the complexity. Furthermore, the connection information among vehicles may change over time [36]. The V_n identifies the RHVs and RRVs through the beacons. Since beacons are the packets sent periodically in a broadcast by vehicles to notify about their type, speed, computation capacity, and state [37–39]. To obtain beacon messages from multi-hop vehicles in a dynamic environment is time-consuming. Since a vehicle can access multi-hop vehicle in a relay fashion, thus, the time it takes to receive the multi-hop vehicles' information might not be reliable over time. Moreover, frequent updates of beacon messages might overload the wireless channel, with a potential impact on communication reliability. Hence, with multi-hop, the appropriate quality of service would not be guaranteed [37]. Therefore, in our algorithm, each vehicle must keep the computation capacity of vehicles present in its one-hop communication range. The one-hop information is symbolized as $\Gamma_{v_n} = (f_{V_1}, f_{V_2}, f_{V_3}, ..., f_{V_j})$, which represents the computation capacity of all the vehicles available in the communication range of vehicle V_n. We further denote the Γ_{v_n} as a set of vehicles present in the communication range of V_n. As the one-hop information is kept locally, we follow the greedy algorithm to choose the best vehicle among all vehicles present in the communication range.

We calculate all the possible available resources of vehicles for offloading a task from the RHV V_n to the vehicles $V_i (V_i \in \Gamma v_n)$. Then, we select the vehicle with the least cost as the qualified vehicle among entire candidate vehicles present in its vicinity, where $C_n^{V_i}$ is the cost of all

Table 2 Simulation Parameters

Parameter	Value
No. of VEC servers (RSUs)	5
ψ_{V2V}	1 unit
ψ_{V2I}	1 unit
Φ_{V2V}	2.5 unit/s
Φ_{VEC}	10 units/s
Communication Range of vehicles C_{limit}	150m
Computation capacity of each vehicle	$[10^6, 2 \times 10^8]$c/s
Computation capacity of VEC server	$[8 \times 10^8]$c/s
Radius of RSU is r	200m
Task's size	[1,50] MB
Vertical distance from RSU to road is e	100m
Relative speed between vehicles	[10,20]m/s
The speed of vehicle V_n is v_n^{abs}	60km/h
V2I and V2V bandwidth B_{V2I} and B_{V2V}	1 MHz
White Guassian Noise power N_0	3×10^{-13}

candidate vehicles. As can be seen in Fig. 1, that the qualified vehicle is represented by the yellow line while the yellow dotted line represents the vehicle(s) present within V_n's communication range. The task will transmit from V_n to a qualified nearby vehicle. Thus, the qualified vehicle obtained by our algorithm is as follow:

$$V_{n,i} = \min\{C_n^{V_i}, V_i \in \Gamma v_n\}. \tag{27}$$

The vehicle V_n offload the β_n part to the qualified vehicle $V_{n,i}$. While in the transmission process, the vehicle V_n must stay in the coverage area of the vehicle V_i. Specifically, the transmission time $t_{n,up}^{V_i}$ of vehicle V_n to its nearby vehicle must be less than the stay time of the vehicle V_n in its communication range. We should examine whether the portion of the task β_n can be completely delivered before the vehicle runs out of the communication range. Therefore, the following constraints in Eq. (26e) must be satisfied.

After the task has been computed by the vehicle $V_{n,i}$, the result will be forwarded to both the vehicle V_n and the nearest VEC server. Thus in case, the result cannot be received by V_n due to the communication range limitations and mobility, the VEC server with its wider coverage area will transfer the result back to the requesting vehicle. However, If there is no vehicle found having enough resources to bear the Vehicle V_n task then V_n will also offload the V2V part, i.e., β_n to a VEC server. The algorithm to choose the qualified nearby vehicle is as follow:

Algorithm 1: Choosing Qualified Nearby Vehicle

```
INPUT: one-hop computation information (f_{V_1}, f_{V_2}, f_{V_3}, ..., f_{V_j})
OUTPUT: Qualified vehicle V_{n,i}
for n = 1; n <= N; n++ do
    for i = 1; i <= j; i++ do
        Γ_{v_n} = (f_{V_1}, f_{V_2}, f_{V_3}, ..., f_{V_j})
        From Eq. (27) we can get.
        V_{n,i} ← min{C_n^{V_i}, V_i ∈ Γv_n}
    end
    return V_{n,i}
end
```

Ratio estimation for partial task offloading

The time to transmit the portion of a task of vehicle V_n must satisfy the constraint of the stay time of a selected vehicle. We consider to offload the portion of the task by estimating the offloading time and the velocity of the vehicles as well as meeting the maximum tolerable delay. Therefore, by exploiting the Eqs. (6) and (16), we can formulate as:

$$\beta 1_n * \frac{D_n}{R_{V2V}^{n,i}} + \beta 1_n * \frac{B_n}{fV_i} \leq t_n^{max},$$
$$\beta 1_n \leq \frac{t_n^{max}}{\{\frac{D_n}{R_{V2V}^{n,i}} + \frac{B_n}{fV_i}\}}, \tag{28}$$

that allows to find the value of $\beta 1_n$ parameter according to maximum tolerable delay t_n^{max}. In addition, the ratio of $\beta 2_n$ according to stay time can be calculated as:

$$\beta 2_n * \frac{D_n}{R_{V2V}^{n,i}} \leq \frac{d_{n,i}(t)}{|\overrightarrow{v}_n - \overrightarrow{v}_i|},$$
$$\beta 2_n \leq \frac{d_{n,i}(t)}{|\overrightarrow{v}_n - \overrightarrow{v}_i| * \{\frac{D_n}{R_{V2V}^{n,i}}\}}, \tag{29}$$

the above equations set an upper limit on the portion of task to be offloaded. Moreover, Algorithm 2 estimates the values of α_n, β_n, and γ_n for all vehicles.

The entire process of mobility-aware partial task offloading algorithm is described in Algorithm 3.

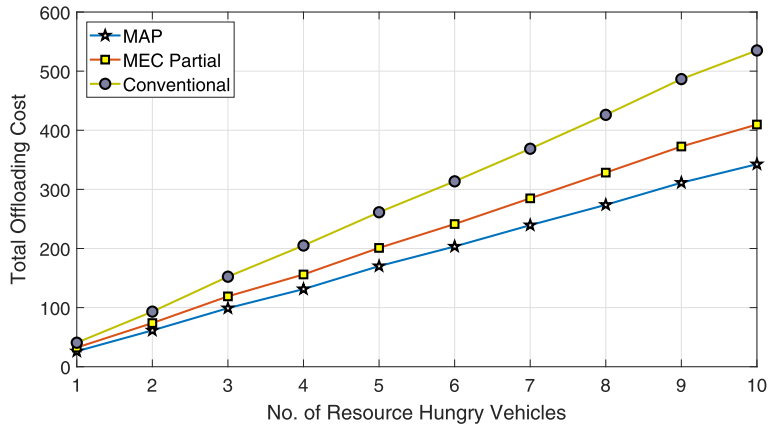

Fig. 2 The total offloading cost in terms of varying RHVs, when RRVs=10

Algorithm 2: To estimate the offloading ratios

1 **INPUT:** R_n, $V_{n,i}$
2 **OUTPUT:** $\alpha_n, \beta_n, \gamma_n$
3 **for** $n = 1; n <= N; n++$ **do**
4
$$\alpha 1_n \leq \frac{t_n^{max}}{\{\frac{B_n}{fV_n}\}} \tag{30}$$
5 $\alpha_n \leftarrow \frac{\alpha 1_n}{D_n}$
6 $\beta_n \leftarrow \min\{\frac{\beta 1_n}{D_n}, \frac{\beta 2_n}{D_n}\}$
7 $\gamma_n \leftarrow 1 - (\alpha_n + \beta_n)$
8 **end**
9 **return** $\alpha_n, \beta_n, \gamma_n$

In Algorithm 3, each RHV offloads its task locally, to the qualified vehicle, and VEC server according to the portion of α_n, β_n, and γ_n, respectively. This process may continue until the maximum tolerable delay is reached. Here, Lines 5-8 are used to compute the α_n locally while computing the cost. Lines 9–20 are used to compute the β_n offloaded to the qualified vehicle $V_{n,i}$ that is having a minimum cost. If the vehicle remains in the coverage area before the job is done, then it returns the output directly to the RHV, otherwise, it handovers the output to the nearest VEC server. Lines 21–30 are used to compute the γ_n portion on the nearest VEC server. If the computation is finished within the stay time then the VEC server immediately transmits the output to V_n. However, if it is not the case then it forwards the output to the VEC server where the vehicle is currently present. Line 33 is used to represent the total offloading cost of the whole system.

Results and discussions

In this section, we analyze our proposed mobility-aware partial task offloading scheme. We consider five RSUs, each having a VEC server located alongside a unidirectional road in an urban mobility road traffic scenario. We also assume that vehicles follow random distribution on the road. In our simulation, we consider the computing speed of each vehicle in the range $[10^6, 2 \times 10^8]$ cycles/s. We set the computational speed of VEC server as $F_n^{VEC} = 8 \times 10^8$cycles/s [40]. The vehicle speed V_n is $v_n^{abs} = 60km/hour$ [41]. The relative speed among vehicles is set in the range of $[10, 20]\, m/s$. The vertical distance from RSU to the road is set as $e = 100m$. The communication radius of the RSU coverage area is taken as $r = 200m$. In addition, the radius of V2V communication C_{limit} is set to $150m$ [28]. Similarly, the White Gaussian noise power $N0 = 3 \times 10^{-13}$, V2I and V2V communication bandwidth $B_{V2I} = B_{V2V}$ = 1MHz, the V2I path loss exponent $\sigma = 2$, and the transmit power of onboard unit $P_t = 1.3W$ [33]. As the qualified vehicle (RRV) acts as a mini server for the requested vehicle (RHV). Therefore, we set the communication and computation cost for the vehicle, according to the ratio of the total computational capacity of the VEC server. The detailed setting of simulation parameters is listed in Table 2.

In order to show the efficiency of our proposed approach (designated as MAP), we compared its performance with conventional partial offloading (represented as conventional) and MEC partial offloading techniques (i.e., MEC Partial) [27]. In the conventional partial offloading scheme, the task is computed locally and on the VEC server without the support of other vehicles. While the MEC Partial gets offloading support from both VEC server and nearby vehicles along with local computing. However, this approach determines the vehicle according to the stay time to minimize the outage probability.

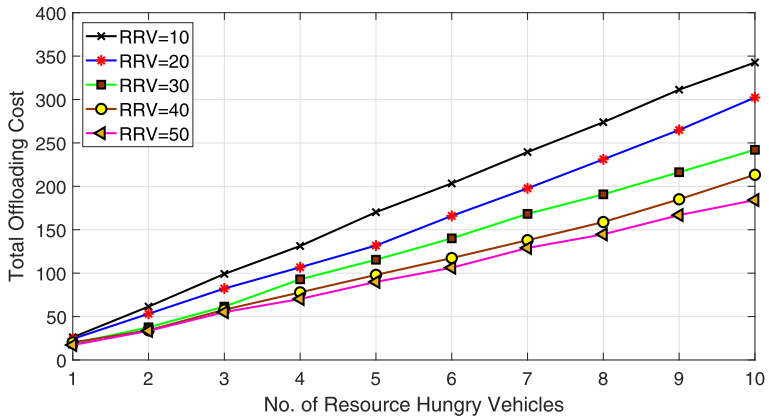

Fig. 3 The total offloading cost versus varying RHVs with fixed No. of RRVs

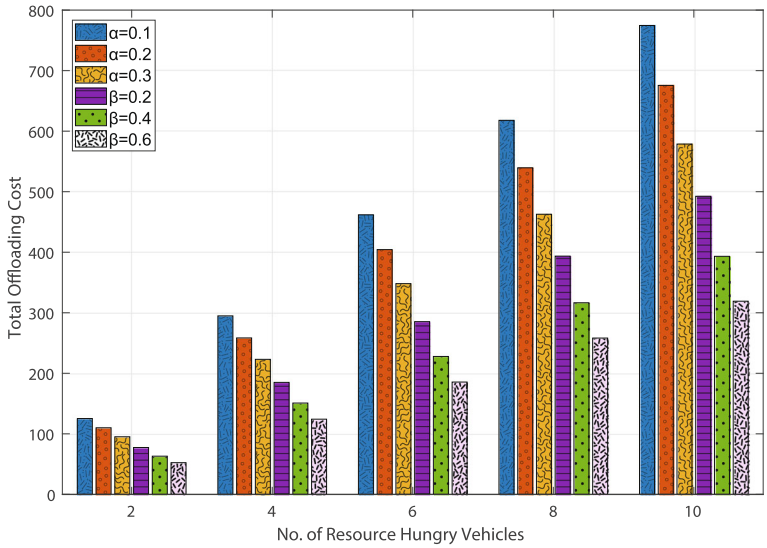

Fig. 4 The total offloading cost with fixed values of α_n and β_n, when RRVs=10

Figure 2 represents the total computation offloading costs regarding vehicle density on the road. We make a comparison of our proposed MAP scheme with two benchmark schemes, i.e., Conventional and MEC partial offloading schemes, respectively. From Fig. 2, it is observed that the performance of the MAP is best in terms of saving the total offloading cost than the compared schemes, especially when the vehicle's density is high. Nevertheless, in the low density of vehicles scenario, the difference between the costs of all three schemes are minor. Additionally, a load of computation on each VEC server is low. A great percentage of the offloaded tasks on the VEC servers could be computed within the required time while the vehicles accessing the RSUs. Since the queue size is small, which affects the stay time of the task on the VEC server, hence, lowers the cost. On the other hand, in the case of high density, the burden on the VEC server increases, which may also increase the stay time of the task on the VEC server as well as the cost. Due to communication and computation, the overall costs of the conventional offloading scheme rises fast as the density of the vehicles grows. Moreover, in MEC partial offloading, part of the transmission is offloaded to the vehicle, which has the least cost as compared to the other vehicles in the vicinity at that time. Also, the portion of the task must be uploaded and executed within the stay time of that vehicle. Whereas in our scheme, the portion of the V2V task β_n must be uploaded within the

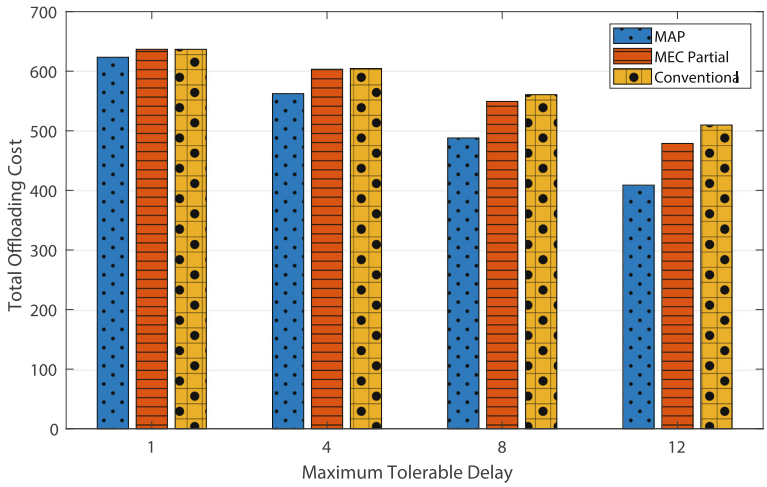

Fig. 5 The total offloading cost versus maximum tolerable delay t_{max}, when RHVs=10 & RRVs=10

Algorithm 3: MAP Task Offloading Algorithm

INPUT: $V_{n,i}, \alpha_n, \beta_n, \gamma_n$
OUTPUT: C_{Total}
for $n = 1; n <= N; n++$ **do**
  while t_n^{max} **do**
    goto Algorithm 2 to fetch α_n
    By exploiting Eqs. (11-13)
    We can get and denote the local computing cost as: C_n^{Local}
    goto Algorithm 1 to fetch $V_{n,i}$
    while $t_{n,up}^{V_i} \leq t_{n,stay}^{V2V}$ **do**
      By using Eq. (14)
      goto Algorithm 2 to fetch β_n
      Transmit the β_n on $V_{n,i}$
    end
    By using Eqs. (15-17)
    We can get and denote the V2V computing cost as: C_n^{V2V}
    if $T_n^{V2V} \leq t_{n,stay}^{V2V}$ **then**
      Give output directly to V_n
    else
      Put the output on nearest VEC server.
    end
    while $t_{n,up}^{VEC} \leq t_{n,stay}^{V2I}$ **do**
      goto Algorithm 2 to fetch γ_n
      By applying Eqs. (18,20-23)
      We can get and denote the VEC computing cost as: C_n^{VEC}
    end
    if $T_n^{VEC} \leq t_{n,stay}^{V2I}$ **then**
      Give output directly to V_n
    else
      Give output to VEC server where the V_n is currently present.
    end
  end
  $C_n \leftarrow C_n^{Local} + C_n^{V2V} + C_n^{VEC}$
  $C_{Total} \leftarrow C_{Total} + C_n$
end

stay time to other vehicles and the duration it takes from uploading to execution must be within the maximum tolerable delay. Therefore, the value of β_n will be greater, thus reduces more cost and utilizes vehicles' resources. Our proposed scheme notably reduces the computation and communication cost of the system by fully exploiting the underutilized vehicular resources up to their full capacity.

Figure 3 indicates a decrease in the total offloading cost of the system with an increase in the number of RRVs. From Fig. 3, we observe that when the number of RRVs is considered fixed such as 10, 20, 30, 40, and 50, and keep varying RHVs, the total offloading cost starts declining with the increase in RRVs. This is mainly due to the fact that RHVs may have more opportunities in selecting the best nearby vehicle, which incurs less cost and more benefit. Thus, these result comparisons reveal that in partial task offloading the increase in RRVs significantly influences the overall system performance.

Figure 4 illustrates the total offloading cost with an increase in the number of RRVs. We evaluate our scheme by examining the impact of both α_n and β_n in the total offloading cost of the system. From Fig. 4, we observe that as much as the α_n contributes to the offloading process, the cost decreases accordingly. Similarly, with the increase in β_n, the vehicles' computational resources can be exploited more effectively. Since the remaining portion for the VEC server remains less as the values of α_n and β_n increases, thus it becomes cost-effective. Our scheme results further corroborate the numerical analysis.

Figure 5 shows the total offloading cost versus the maximum tolerable delay. Here the task data size is fixed to 25 MB while the RHVs and RRVs are set to 10, respectively. To observe the role of tolerable delay, we take different values of maximum tolerable delay. Considering practical assumption, at any given time the VEC servers are at the heterogeneous level of computation load. If the vehicles choose a conventional scheme, the complex computational tasks may take a longer time as well as the higher cost to complete their job. Thus, a larger part of the task will be shifted to nearby RSU, eventually leading to increased system cost. Moreover, if the vehicles adopt MEC partial offloading scheme, the vehicle still consumes the VEC computation resources by putting more load. Besides, the tasks that require prompt response would be offloaded, if the vehicle is still in the communication range of the other vehicle until the job is fully completed. Among all the other schemes, we note that the MAP always gets the best performance since more portion of a task can be distributed to nearby vehicles. In Fig. 5, at maximum tolerable delay= 12, compared to the conventional and MEC partial schemes, the total offloading cost is saved by 19% and 15%, respectively. Moreover, our proposed scheme becomes more efficient when the density of vehicles increases. From Fig. 5, it can easily be observed that the MAP scheme is cost-effective under any given tolerable delay.

Figure 6 indicates the relationship between the task data size D_n and the total offloading cost, where the total number of RHV and RRV are set to 10, respectively. From Fig. 6, we observe that as the size of the task increases on the x-axis, the curves of all three schemes give an upward trend, which proves that the size of the task has a direct impact on the total offloading cost. Our proposed scheme achieves the best results as it shows the slowest growth trend. The slope of the conventional curve is greater than

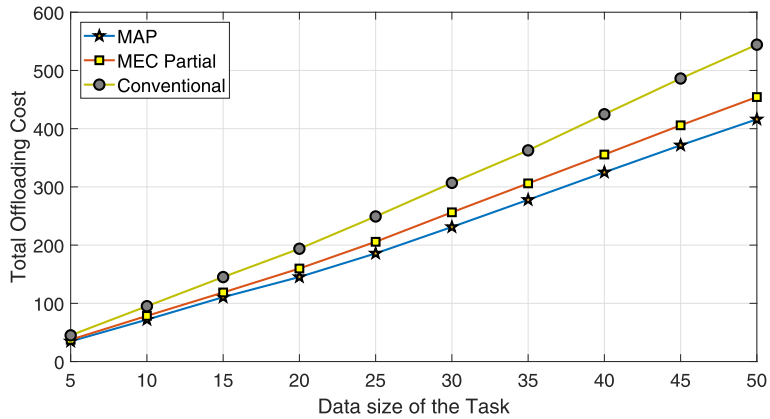

Fig. 6 The total offloading cost to data size D_n, when RHVs=10 & RRVs=10

that of the other two schemes, showing that the total offloading cost of the system grows rapidly. This indicates that the larger the data volume of the computing task, the major portion will be allotted to the VEC server, thus increasing the total system cost. While in our proposed MAP task offloading scheme, the transmission and computation load on the VEC server will be released as well as avoids network congestion.

Figure 7 represents a comparison between the total offloading cost with varying RHVs velocity while fixing the speed of RRVS to 60 km/h. In terms of the impact of speed, we note from Fig. 7 that low offloading cost incurs when the RHVs speed is close to the RRVs speed. Since RHVs have stable and longer stay time, consequently, a greater portion of the task is transmitted to RRVs. On the other hand, when the speed of RHVs is less or greater than 60 km/h, it affects the V2V connection time as the RHVs quickly move out of the communication range of RRVs. In that case, a larger part of the task is shifted to the VEC server, which increases offloading cost. We can observe that the proposed scheme outperforms other benchmark schemes.

Conclusion

In this paper, we proposed a mobility-aware partial task offloading algorithm to minimize the total offloading cost. To make it cost-efficient, the vehicle's available resources are exploited. In this scheme, the task is divided into three parts. We further determined the allocation ratio among these parts according to the vehicles' mobility.

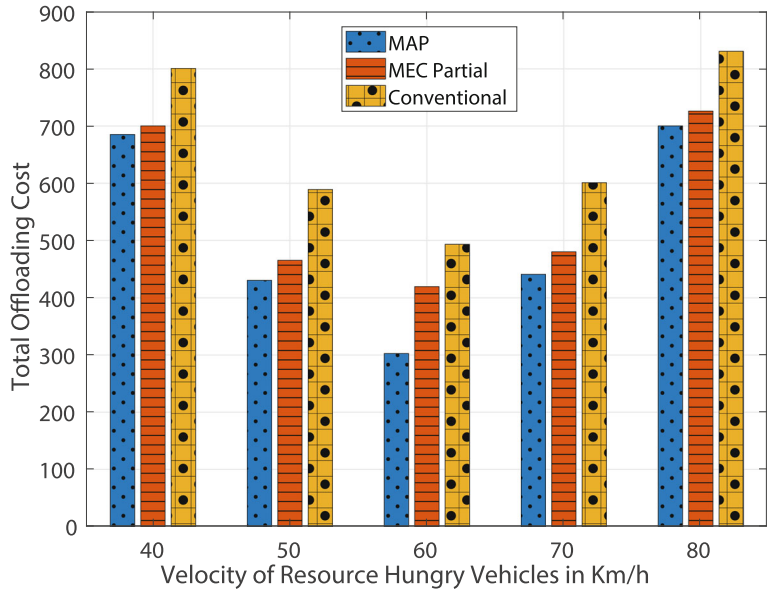

Fig. 7 The total offloading cost versus varying RHVs velocity, when velocity of the RRVs=60km/h

Moreover, we estimate the transmission rates for V2V and V2I communication in the light of practical assumptions. Simulation results demonstrate that nearby vehicle communication and computation resources not only reduced the cost but also offload the burden of the VEC servers, especially which are deployed in a dense urban environment. Extensive simulation results demonstrated our proposed scheme's effectiveness against the compared schemes. Although the results provided in this work significantly contribute to the state-of-the-art, yet they can be improved in many ways. One of the major challenges in partial task offloading for vehicles is mobility, which greatly affects the V2V and V2I communication. In this regard, our work would be extended to highway scenarios, and also the task offloading can be improved by incorporating mmWave communications or considering 5G New Radio. These challenging yet interesting extensions are left for our future work.

Acknowledgements
Not applicable.

Authors' contributions
Conceptualization, Salman Raza; Methodology, Salman Raza and Manzoor Ahmed; Resources, Salman Raza; Validation, Muhammad Rizwan Anwar and Muhammad Ayzed Mirza; Visualization, Muhammad Rizwan Anwar and Muhammad Ayzed Mirza; Writing – original draft, Salman Raza and Manzoor Ahmed; Writing – review & editing, Wei Liu, Qibo Sun and Shangguang Wang. The author(s) read and approved the final manuscript.

Funding
This work was supported by the National Key R&D Program of China (2018YFB1402801), and Funds for Creative Research Groups of China (61921003).

Availability of data and materials
The random numbers are generated to check validity. Therefore, no supporting dataset is available.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China, 100876 Beijing, China. [2]Software Engineering, Beijing University of Posts and Telecommunications, Beijing, China, 100876 Beijing, China. [3]College of Computer Science and Technology, Qingdao University, Qingdao, China, 266071 Qingdao, China. [4]School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China, 100876 Beijing, China.

Received: 7 January 2020 Accepted: 13 May 2020
Published online: 02 June 2020

References

1. Abolfazli S, Sanaei Z, Ahmed E, Gani A, Buyya R (2013) Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. IEEE Commun Surv Tutor 16(1):337–368
2. Technology and Computing Requirements for Self-Driving Cars. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/automotive-autonomous-driving-vision-paper.pdf
3. Bitam S, Mellouk A, Zeadally S (2015) Vanet-cloud: a generic cloud computing model for vehicular ad hoc networks. IEEE Wirel Commun 22(1):96–102
4. Jang I, Choo S, Kim M, Pack S, Dan G (2017) The software-defined vehicular cloud: A new level of sharing the road. IEEE Veh Technol Mag 12(2):78–88
5. Taleb T, Dutta S, Ksentini A, Iqbal M, Flinck H (2017) Mobile edge computing potential in making cities smarter. IEEE Commun Mag 55(3)
6. You C, Huang K, Chae H, Kim B-H (2016) Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans Wirel Commun 16(3):1397–1411
7. Lin B, Zhu F, Zhang J, Chen J, Chen X, Xiong NN, Mauri JL (2019) a time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. IEEE Trans Ind Inform 15(7):4254–4265
8. Huang X, Yu R, Kang J, Zhang Y (2017) Distributed reputation management for secure and efficient vehicular edge computing and networks. IEEE Access 5:25408–25420
9. Wang S, Zhang X, Zhang Y, Wang L, Yang J, Wang W (2017) A survey on mobile edge networks: Convergence of computing, caching and communications. IEEE Access 5:6757–6779
10. Chen X, Chen J, Liu B, Ma Y, Zhang Y, Zhong H (2019) androidoff: Offloading android application based on cost estimation. J Syst Softw 158:110418
11. Chen X, Chen S, Ma Y, Liu B, Zhang Y, Huang G (2019) an adaptive offloading framework for android applications in mobile edge computing. Sci China Inf Sci 62(8):82102
12. Ahmed M, Li Y, Waqas M, Sheraz M, Jin D, Han Z (2018) A survey on socially aware device-to-device communications. IEEE Commun Surv Tutor 20(3):2169–2197
13. Waqas M, Niu Y, Li Y, Ahmed M, Jin D, Chen S, Han Z (2019) Mobility-aware device-to-device communications: Principles, practice and challenges. IEEE Commun Surv Tutor. https://doi.org/10.1109/comst.2019.2923708
14. Oteafy SM, Hassanein HS (2018) Iot in the fog: A roadmap for data-centric iot development. IEEE Commun Mag 56(3):157–163
15. Anawar MR, Wang S, Azam Zia M, Jadoon AK, Akram U, Raza S (2018) Fog computing: An overview of big iot data analytics. Wirel Commun Mob Comput 2018
16. Hou X, Li Y, Chen M, Wu D, Jin D, Chen S (2016) Vehicular fog computing: A viewpoint of vehicles as the infrastructures. IEEE Trans Veh Technol 65(6):3860–3873
17. 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. https://doi.org/10.1155/2019/3159762
18. Zhang K, Mao Y, Leng S, Vinel A, Zhang Y (2016) Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks. In: Proceeding of the 8th International Workshop on Resilient Networks Design and Modeling. IEEE. pp 288–294. https://doi.org/10.1109/rndm.2016.7608300
19. Ye D, Wu M, Tang S, Yu R (2016) Scalable fog computing with service offloading in bus networks. In: 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud). IEEE. pp 247–251. https://doi.org/10.1109/cscloud.2016.34
20. Feng J, Liu Z, Wu C, Ji Y (2019) Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling. IEEE Veh Technol Mag 14(1):28–36
21. Zhang K, Mao Y, Leng S, Maharjan S, Zhang Y (2017) Optimal delay constrained offloading for vehicular edge computing networks. In: Proceeding of the International Conference on Communications. IEEE. pp 1–6. https://doi.org/10.1109/icc.2017.7997360
22. Lai Y, Yang F, Zhang L, Lin Z (2018) Distributed public vehicle system based on fog nodes and vehicular sensing. IEEE Access 6:22011–22024
23. Ren J, Yu G, Cai Y, He Y, Qu F (2017) Partial offloading for latency minimization in mobile-edge computing. In: Proceeding of the Global Communications Conference. IEEE. pp 1–6. https://doi.org/10.1109/glocom.2017.8254550
24. Luoto P, Bennis M, Pirinen P, Samarakoon S, Horneman K, Latva-Aho M (2017) Vehicle clustering for improving enhanced lte-v2x network performance. In: Proceeding of the European Conference on Networks and Communications. IEEE. pp 1–5. https://doi.org/10.1109/eucnc.2017.7980735
25. Zhang K, Mao Y, Leng S, He Y, Zhang Y (2017) Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. IEEE Veh Technol Mag 12(2):36–44

26. Li L, Zhou H, Xiong SX, Yang J, Mao Y (2019) Compound model of task arrivals and load-aware offloading for vehicular mobile edge computing networks. IEEE Access 7:26631–26640
27. Bozorgchenani A, Tarchi D, Corazza GE (2018) Mobile edge computing partial offloading techniques for mobile urban scenarios. In: Proceeding of the Global Communications Conference. IEEE. pp 1–6. https://doi.org/10.1109/glocom.2018.8647240
28. Wang H, Li X, Ji H, Zhang H (2018) Federated offloading scheme to minimize latency in mec-enabled vehicular networks. In: Proceeding of the Globecom Workshops. IEEE. pp 1–6. https://doi.org/10.1109/glocomw.2018.8644315
29. Cheng X, Wang C-X, Ai B, Aggoune H (2013) Envelope level crossing rate and average fade duration of nonisotropic vehicle-to-vehicle ricean fading channels. IEEE Trans Intell Transp Syst 15(1):62–72
30. 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
31. Wang Y, Sheng M, Wang X, Wang L, Li J (2016) Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Trans Commun 64(10):4268–4282
32. Aminizadeh L, Yousefi S (2014) Cost minimization scheduling for deadline constrained applications on vehicular cloud infrastructure. In: Proceeding of the International Conference on Computer and Knowledge Engineering. IEEE. pp 358–363. https://doi.org/10.1109/iccke.2014.6993446
33. Mazza D, Tarchi D, Corazza GE (2014) A partial offloading technique for wireless mobile cloud computing in smart cities. In: Proceeding of the European Conference on Networks and Communications. IEEE. pp 1–5. https://doi.org/10.1109/eucnc.2014.6882623
34. Guo S, Liu J, Yang Y, Xiao B, Li Z (2018) Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. IEEE Trans Mob Comput 18(2):319–333
35. Fan Y, Zhai L, Wang H (2019) Cost-efficient dependent task offloading for multiusers. IEEE Access 7:115843–115856
36. Lu Z, Sun X, La Porta T (2016) Cooperative data offloading in opportunistic mobile networks. In: Proceeding of the Annual INFOCOM Conference on Computer Communications. IEEE. pp 1–9. https://doi.org/10.1109/infocom.2016.7524494
37. Bazzi A, Masini BM, Zanella A, Thibault I (2017) on the performance of ieee 802.11 p and lte-v2v for the cooperative awareness of connected vehicles. IEEE Trans Veh Technol 66(11):10419–10432
38. Feng J, Liu Z, Wu C, Ji Y (2018) mobile edge computing for the internet of vehicles: Offloading framework and job scheduling. IEEE Veh Technol Mag 14(1):28–36
39. Shah SS, Ali M, Malik AW, Khan MA, Ravana SD (2019) vfog: A vehicle-assisted computing framework for delay-sensitive applications in smart cities. IEEE Access 7:34900–34909
40. Munoz O, Pascual-Iserte A, Vidal J (2014) Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. IEEE Trans Veh Technol 64(10):4738–4755
41. Chen S, Hu J, Shi Y, Peng Y, Fang J, Zhao R, Zhao L (2017) Vehicle-to-everything (v2x) services supported by lte-based systems and 5g. IEEE Commun Stand Mag 1(2):70–76

Publisher's Note