

Article

Minimum-Cost Offloading for Collaborative Task Execution of MEC-Assisted Platooning

Xiayan Fan ^{1,2}, Taiping Cui ^{1,2,*} , Chunyan Cao ^{1,2}, Qianbin Chen ^{1,2}  and Kyung Sup Kwak ³

¹ School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; Fanxiayan@yeah.net (X.F.); Caocyan@yeah.net (C.C.); chenqb@cqupt.edu.cn (Q.C.)

² Chongqing Key Labs of Mobile Communications, Chongqing 400065, China

³ School of Electrical and Computer Engineering, Inha University, Incheon 402-751, Korea; kskwak@inha.ac.kr

* Correspondence: cuitp@cqupt.edu.cn; Tel.: +86-187-1628-5097

Received: 31 December 2018; Accepted: 14 February 2019; Published: 18 February 2019



Abstract: In this paper, we study the offloading decision of collaborative task execution between platoon and Mobile Edge Computing (MEC) server. The mobile application is represented by a series of fine-grained tasks that form a linear topology, each of which is either executed on a local vehicle, offloaded to other members of the platoon, or offloaded to a MEC server. The objective of the design is to minimize the cost of tasks offloading and meets the deadline of tasks execution. The cost minimized task decision problem is transformed into the shortest path problem, which is limited by the deadline of the tasks on a directed acyclic graph. The classical Lagrangian Relaxation-based Aggregated Cost (LARAC) algorithm is adopted to solve the problem approximately. Numerical analysis shows that the scheduling method of the tasks decision can be well applied to the platoon scenario and execute the tasks in cooperation with the MEC server. In addition, compared with task local execution, platoon execution and MEC server execution, the optimal offloading decision for collaborative task execution can significantly reduce the cost of task execution and meet deadlines.

Keywords: platooning; collaborative task execution; mobile edge computing; offloading decision

1. Introduction

With the explosive increase of vehicle terminals, new vehicular services such as 3D navigation, automatic driving and so on, which require the capability of supercomputing and mass storage, are developing rapidly. Recently, MEC technology has been proposed to solve the problem of global computing resource shortage caused by massive access of mobile devices in the fifth-generation mobile communication (5G) [1], which makes the transmission cost of mobile network lower and more efficient. MEC aims to reduce the pressure of vehicle terminals by offloading computing load to mobile edges. In the traditional wireless access network, the base station (BS) deployed on the edge of the mobile network carries out the services forwarding, but the BS does not actively analyze or respond to the user's request. With the introduction of MEC, these MEC devices deployed on the edge of the network provide information technology service environment and cloud computing capability in the wireless access network nearest to user terminals (included vehicle terminals), e.g., small cell BS, macro cell BS or access to the MEC server by wireless access points, and minimize the transmission delay.

Platooning is the first step to realize automatic driving. It is the most representative case of 5G. In the platoon, the distance and speed between vehicles are controlled by automatic control system through real-time updated kinematics data [2]. Vehicles in a platoon are on the same driveway, like a train, and the distance between vehicles is fixed and very short. The higher frequency of vehicle information exchange in the platoon, the faster the maneuvering response of its members, and the more

can avoid the unstable state of the platoon. This requires very high reliability and ultra-low delay data transmission, namely, vehicles should control the delay caused by the process of information processing and exchanging within 100 ms. On the other side, platooning is able to reduce fuel consumption and gas emission, as well as safe and efficient transportation in the context of intelligent transportation system (ITS) [3]. Generally, the platoon consists of two kinds of members: one is the leader (controller) and others are the members of the platoon (including the tail vehicle, the relay vehicle) [4], as shown in Figure 1. The longitudinal and lateral positions of each vehicle in the platoon are controlled by the collected data and vehicle status information (such as speed, acceleration, deceleration and location and so on). To reduce the possibility of the status of the platoon instability, these data should exchange frequently between vehicles in the platoon.

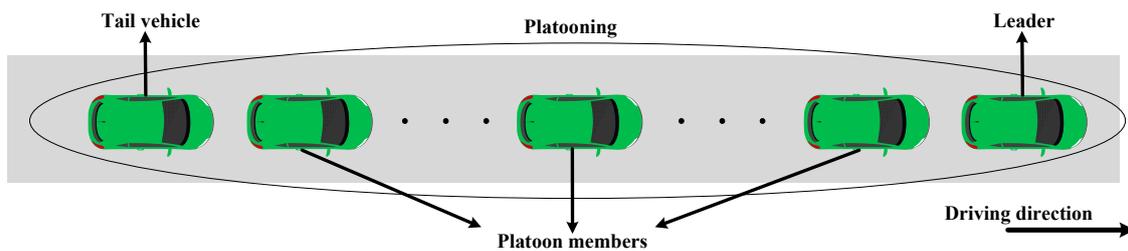


Figure 1. Platooning composition.

Most of the vehicle-related data are processed by the terminals that have been widely equipped on the vehicles [5]. More novel and attractive vehicle services have attracted more and more people to use them, such as 3D navigation, driverless, traffic information systems, voice processing and other vehicular networking applications. In the future, vehicles will be equipped with Augmented Reality (AR) applications that allow drivers to observe the surroundings of vehicles in windowless vehicles [6,7]. These types of vehicle applications are typical computing resource-hungry services, requiring high density computing resources and computing costs, and will be used to the platooning for better services. EyeDentify is a typical example [8], which matches the image based on target recognition algorithm. EyeDentify could be used as the security unlock for the vehicle users, and it includes a series of steps for feature extraction to transform the original image into a feature vector, and each of the step can be offloaded. Especially when running the delay-sensitive application on vehicle terminals, the vehicle users desire a fast response. For example, AR application combines computer-generated data with physical reality [9]. AR application contains five key components: the video source, the tracker, the mapper, the object recognizer and the renderer. First, the video sources must be collected by the local device. Second, the tracker, mapper and object recognizer are the computation-intensive components, which can be offloaded to the MEC server to reduce the computing time of the tasks and meet the task execution time requirements. Third, render must be executed locally and the calculated results need to be sent back to the local device for display. There are many relative applications for V2X services like Forward Collision Warning, Pre-crash Sensing Warning, V2X Road Safety Service via Infrastructure [10].

Due to the limited size of the resource space, the available resources of the vehicle terminals cannot meet the service requirements, which will cause high application processing delay, and even lead to safety accidents and so on. In addition, the current centralized server deployment in cellular network is not sufficient to meet future service requirements, such as delay, reliability and availability, which are key parameters. End-to-end delay must be reduced to within the range of service requirements (non-security services: 100 ms to 1 s, security related services or delay-sensitive services: less than 20 ms [7]). For instance, in current network architecture, offloading the workload of the vehicle terminal to the cloud center will cause extremely high transmission delay and it is difficult to achieve real-time data transmission [11]. The tension between computing resource-hungry applications and vehicle terminals with limited computing resources poses a major challenge to the development of

mobile platforms [12]. In this paper, we consider a platoon with an MEC server to assist offload the workload of the vehicle terminal. Specifically, within the deadline of the application tasks, the resource purchase cost is employed to determine whether the tasks are executed locally, or offloaded to the other platoon members, or to the MEC server. We aim to find an optimal collaborative offloading decision among the members of the platoon and the MEC, and to execute the tasks with minimum resource purchase cost by the vehicle user within the deadline of the tasks. Mathematically, we model a minimum-cost task offloading problem as a constrained shortest path problem on a directed acyclic graph. Then we use the classical Lagrangian Relaxation-based Aggregated Cost (LARAC) algorithm to obtain an optimal solution of the constrained optimization problem. Finally, the simulation results of inputting the existing measurement data show that the offloading strategy is the optimal offloading strategy. Comparing the task execution locally or offloaded to the other members of the platoon, or offloaded the task to the MEC server, the cooperative task execution can complete the task within the task deadline at the minimum cost.

The rest of the paper is organized as follows. Section 2 reviews the relevant recent works. We give the system model in Section 3. In Section 4, delay constrained offloading decision is modeled as a constrained shortest path problem. Then we get an optimal strategy of offloading decision for collaborative task execution in Section 5. Section 6 shows the numerical analysis of offloading decision procedure, and Section 7 concludes the paper.

2. Related Work

There are a number of concepts that are closely related to MEC, for instance, mobile cloudlet systems, and MEC is considered to be the natural evolution of previous mobile cloud services (e.g., cloudlet) [13–15]. MEC provides a promising solution to the challenge for the development of mobile platforms and extends the capabilities of vehicle terminals, by providing additional computing, storage, and bandwidth resources in an on-demand manner [7].

MEC aims to further reduce delay, improve network operation efficiency, and promote service distribution capability, so as to improve the end-user experiences. Some of the proposed solutions can be applied to MEC scenarios. For example, an optimized structure for offloading a single mobile device to multiple MEC servers was proposed [16]. By jointly optimizing the task allocation decision and the Central Process Unit (CPU) frequency of the mobile device, the task execution delay and the energy consumption of the mobile device were minimized, but it only considered a single offloading architecture and could not achieve the multilayer tasks offloading. In addition, a game method could be used to solve the multi-user distributed computing offloading problem in multi-channel wireless interference environment, and provided users with computing resources by MEC at the edge of wireless access network [12]. However, the research did not consider the division of the application tasks, which were not flexible for execution of tasks. Some people considered offloading decision from the point of view of the servers, set the price for the unit resource provided to each vehicle user, and used the gain function of task offloaded to maximize the profit of the vehicular edge computing server [17], which increased the purchase cost of the vehicle user that was contrary to the interests of consumers. The authors of Reference [18] proposed an offloading algorithm minimized the completion time of the tasks. Some tasks were executed locally, and the other part were offloaded to the cloud. This algorithm could not be deployed directly to the realistic scenario, because reducing the completion time would result in too much processing cost. The authors of Reference [19] proposed a collaborative task execution strategy between the end-user and the cloud, which accomplished the tasks with minimum energy consumption within the application deadline. Its simulation results had proved that the collaborative task execution could significantly reduce the energy consumption on the mobile device, thus prolonged the lifetime of devices. The authors of Reference [20] used a directed graph to represent the code blocks and proposed a relatively genetic algorithm to determine how to offload the code blocks among multiple nodes. In Reference [21] the authors proposed an adaptive receding

horizon offloading strategy, which could minimize the estimating cost of offloading among multiple devices and meet the time constraints of the tasks.

At present, there are lack of researches on tasks offloading in platoon, most of which are about connectivity [22,23] and communication strategy [24] of the network in which the platoon is located, but no consideration has been given to how to offload the tasks in the platoon. Due to the stability and predictability of platoon on the road [25], platoon members can also provide stable computing resources to users for a certain period of time, so as to obtain better computing performance and save the cost.

In order to have a deeper understanding of the offloading decision strategy, a simple task model is considered. Each task is executed one after another, forming a linear topology [19]. The task model is transformed into a constrained shortest path problem in directed acyclic graph. We deal with this problem under the path loss model and use Lagrange aggregation cost algorithm to obtain an approximate solution of the constrained problem. The main contributions of this article are summarized as follows:

- The optimization problem of the combined mobile application of each vehicle terminal in a platoon and an MEC server is modeled as a constrained minimizing cost problem.
- By analyzing the characteristics of task model in a linear topology, combining the dynamic programming algorithm and the Lagrangian aggregation cost algorithm, the optimal task decision strategy is obtained under the condition that the deadline of the tasks is satisfied.
- The effectiveness of the proposed algorithm and strategy is verified by simulation. The simulation results show that, in comparison with task platoon execution and the MEC server execution, collaborative task execution can greatly reduce the cost of tasks offloading.

3. System Model

Figure 2 depicts a MEC server assisted platooning in our paper. The platoon controller controls and manages the whole platoon. When the platoon members communicate with the BS, they need to transmit the messages to the BS through the platoon controller, and the members in the platoon can directly communicate with each other [25,26]. Position of the controller in the platoon is not clearly defined in the current research. Most of the articles directly select the leader as platoon controller for the sake of simplification. In this paper, we do not discuss in detail how to choose the controller, which can be selected according to the signal intensity, platoon length and channel multiplexing.

We next introduce the system model of the optimal offloading decision for collaborative task execution, including task model, path loss model and task execution model.

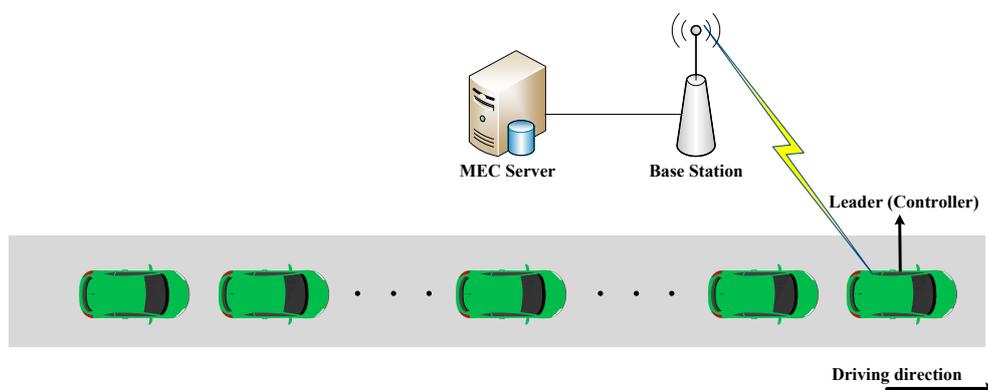


Figure 2. An MEC-assisted platooning scenario.

3.1. Task Model

Figure 3 illustrates a sequence of tasks with a linear topology, in the granularity of either a method or a module [19]. Each task is executed in sequence, the output of the previous task is the input of the

later task, and the whole application has a completion deadline T_d . Note that there are $n + 2$ tasks in an application. Define the demanded computation cycles of the task k as ω_k , and the input data size of task k as d_k , $k = 0, 1, 2, \dots, n + 1$. According to the paper [27], the computing resources of demand have the following relationship with the size of the input data, $\omega_k = \phi d_k$, where ϕ is the computational complexity of the task. The value of ϕ depends on the nature of the task, which is beyond the scope of this paper. The following assumptions are adopted to model the practical problem of tasks. First, the output of a task must be replicated to the other platoon members or the MEC server before the next task is executed. Second, the first and last task of the application must be executed in the same vehicle in the platooning.

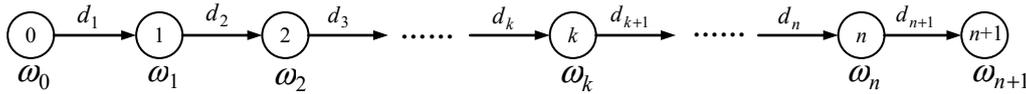


Figure 3. Task model in a linear topology.

3.2. Path Loss Model

In the paper, we consider task offloading for MEC-assisted platooning in a wireless cellular network. The communication mode of platoon members is Vehicle to Vehicle (V2V) communication, which is deployed according to IEEE 802.11p standard [28]. The communication mode between vehicle and BS is Vehicle to Infrastructure (V2I) communication, which is deployed according to LTE-V [29].

When one vehicle is served by another, the path loss $PL_v(l_{v_1, v_2})$ between the vehicle v_1 and vehicle v_2 is modeled by Reference [30]

$$PL_v(l_{v_1, v_2}) = 63.3 + 17.7 \log_{10}(l_{v_1, v_2}) \quad (1)$$

where l_{v_1, v_2} is distance in kilometers between two vehicles. Additionally, the path loss $PL_{MEC}(l_{q, u})$ between the leader and BS is modeled by Reference [31]

$$PL_{MEC}(l_{q, u}) = 128.1 + 37.5 \log_{10}(l_{q, u}) \quad (2)$$

where $l_{q, u}$ is the distance in kilometers between the leader and BS, $q = 1$, $u = 0$ or $q = 0$, $u = 1$. Here, for convenience of discussion, we do not consider fast fading and shadow fading in our wireless communication model. The carrier frequency used in V2I communication is 2 GHz and V2V communication is 5.9 GHz. Therefore, there is no interference between V2I and V2V communication. The distances between vehicles in the platoon and the distances between the platoon and BS are illustrated in Figure 4. The MEC server is numbered 0, the platoon leader vehicle is numbered 1, and the sequence number of vehicles following it increases in turn. l_{v_1, v_2} is the geographic distance from v_1 to v_2 , where v_1 and v_2 are the vehicle terminal numbers, $v_1, v_2 = 1, 2, \dots, m$ and $v_1 \neq v_2$.

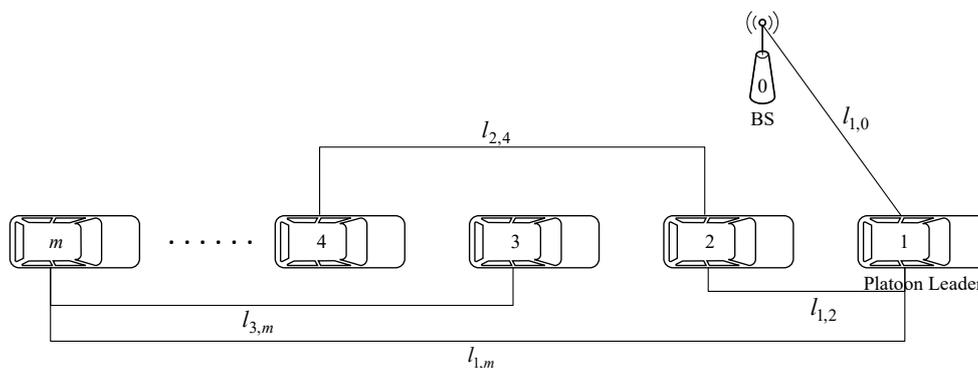


Figure 4. Illustration of the distance of platooning and base station (BS).

Considering that under ideal conditions all the vehicles in the platoon are of the same length and the same spacing so that the distance between the signal transmitter (v_1) and the receiver (v_2) in the platoon is the length of the vehicle plus the vehicle spacing. Then we get

$$l_{v_1,v_2} = \mu|v_1 - v_2| \quad (3)$$

where μ is the distance in kilometers between the adjacent transmitter and the receiver. $|v_1 - v_2|$ is the number of intervals. Specially, considering offloading tasks to the MEC server, and the distance in kilometers between the leader and the BS is set to η , we obtain

$$l_{1,0} = \eta. \quad (4)$$

When the task is offloaded to the MEC server, the change of distance between the leader and the BS has negligible effect on the path loss, so we get $l_{1,0} = l_{0,1}$.

3.3. Task Execution Model

In task execution, the models of platoon execution, MEC server execution, platoon data transmission, and MEC server data transmission are considered in our paper.

3.3.1. Platoon Execution

The application can be initiated by any member in the platoon. The platoon execution includes local execution and platoon members execution. Every member in the platoon has the opportunity to participate in the execution of the tasks. If the task k is offloaded to v_2 , the computing time of the task k is expressed by

$$t_k^{v_2} = \frac{\omega_k}{f_{v_2}} \quad (5)$$

and the cost of offloading the task k is

$$b_k^{v_2} = \alpha_v f_{v_2} \quad (6)$$

where f_{v_2} is the computation resource that the vehicle v_2 can provide and α_v is the computation resource cost of each unit provided by the platoon members. Suppose that the unit price of resources provided by each member is the same, f_{v_2} is constant during the task execution.

3.3.2. MEC Server Execution

If the task k is executed on the MEC server (numbered 0), the computing time of the task k is expressed by

$$t_k^0 = \frac{\omega_k}{f_0} \quad (7)$$

and the cost of offloading the task k is

$$b_k^0 = \alpha_{\text{MEC}} f_0 \quad (8)$$

where f_0 is the computation resource that the MEC server can provide and α_{MEC} is the computation resource cost of each unit provided by the MEC server. Note that the MEC server can provide more computation resources than platoon members, i.e., $f_0 > f_1, f_0 > f_2, \dots, f_0 > f_m$, that results less computation time for the task.

3.3.3. Platoon Data Transmission

If the task k is executed by the platoon members, the data needs to be offloaded to the destination vehicle terminal before it is executed. The data transmission time for offloading the task k from vehicle v_1 to vehicle v_2 is computed by

$$t_{v_1,v_2}^k = \frac{d_k}{R_{v_1,v_2}^k} \quad (9)$$

where R_{v_1, v_2}^k is the data transmission rate from vehicle v_1 to vehicle v_2 . We have

$$R_{v_1, v_2}^k = B_1 \log_2 \left(1 + \frac{\xi_v \text{PL}_v(l_{v_1, v_2})}{N_0} \right) \quad (10)$$

where ξ_v is the signal transmission power of the vehicle, N_0 is the noise power and B_1 is the bandwidth for V2V communication. If $v_1 = v_2$, $t_{v_1, v_2}^k = 0$.

3.3.4. MEC Server Data Transmission

If the task k needs to be offloaded and executed on the MEC server, the data transmission time for offloading the task k from the vehicle v_1 to the MEC server is computed by

$$t_{v_1, 0}^k = \frac{d_k}{R_{v_1, 1}^k} + \frac{d_k}{R_{1, 0}^k} \quad (11)$$

If the task k needs to be executed on a platoon member, the data transmission time for offloading the task k from the MEC server to the vehicle v_2 is represented by

$$t_{0, v_2}^k = \frac{d_k}{R_{1, v_2}^k} + \frac{d_k}{R_{0, 1}^k} \quad (12)$$

$R_{1, 0}^k$ and $R_{0, 1}^k$ indicate the transmission data rate from the leader to the MEC server and from the MEC server to the leader respectively. We have

$$R_{1, 0}^k = B_2 \log_2 \left(1 + \frac{\xi_v \text{PL}_{\text{MEC}}(l_{1, 0})}{N_0} \right) \quad (13)$$

and

$$R_{0, 1}^k = B_2 \log_2 \left(1 + \frac{\xi_{\text{MEC}} \text{PL}_{\text{MEC}}(l_{0, 1})}{N_0} \right) \quad (14)$$

where ξ_{MEC} is the transmission power of the MEC server and B_2 is the bandwidth for V2I communication. Because the platoon members cannot communicate directly with the BS, the data needs to be transmitted to the BS through the leader. The connection between BS and MEC is wired, and the time of wired transmission is ignored here.

4. Delay Constrained Offloading

The tasks of an application are represented by the directed acyclic graph $G = (V, C)$. $V = 0, 1, 2, \dots, m$ is the finite node set that represents the task offloading nodes, and C is the edge set that denotes the cost of task offloading. Figure 5 shows the task execution flow for collaborative offloading decision. S is the starting point of the application and D is the task destination terminal. For simplicity, take vehicle m as the requestor of the application as an example. An application contains $n + 2$ tasks. Except for the initial task and the last task, all the platoon members and the MEC server have the opportunity to execute the other n tasks.

Denote x and y are adjacent node numbers of an edge, $x, y \in V$ and $x \neq y$. Each edge corresponds to a task offloading decision and the weights of the edges $c_{x, y}^k$ are non-negative value, namely $c_{x, y}^k \geq 0$. This weight value stands for the cost of offloading the task and the time it takes to execute the task. Specifically, if the weight value is considered to be a cost price and the task k needs to be offloaded from x to y for execution, we obtain the weight $c_{x, y}^k = b_k^y$. Additionally, if the weight value is considered to be a time delay, we obtain the weight value $c_{x, y}^k = t_k^y + t_{x, y}^k$ that is the sum of the computation time and the transmission time of the task k . Therefore, two directed acyclic graphs with respect to the cost price and time delay can be obtained.

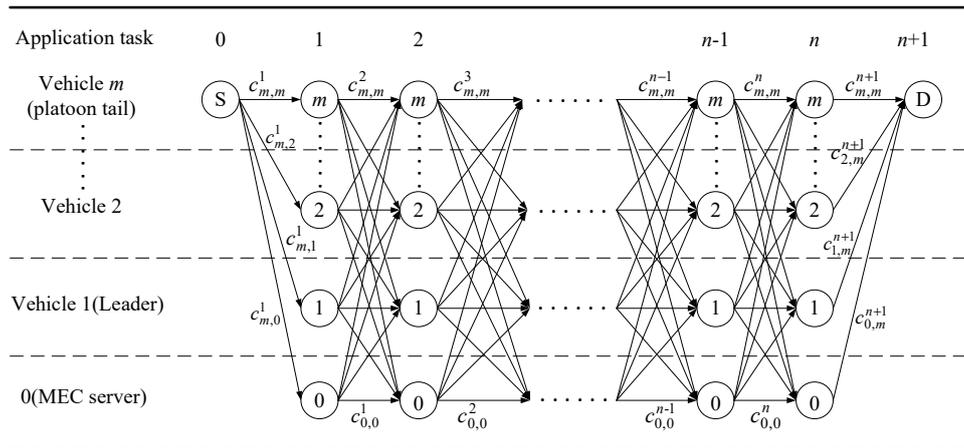


Figure 5. The task execution flow for collaborative offloading decision.

Under this framework, we then transform the optimal task offloading decision into a shortest path problem to find the path with minimum cost between nodes S and D . It is constrained by the deadline time (T_d) of the tasks, and the time delay of the path should be less than or equal to T_d . If the time delay of a path p satisfies the constrained conditions, p is an appropriate path. A path p^* is an optimal path with the minimum cost among all appropriate paths. We mathematically formulate the problem as a constrained shortest path problem by

$$\begin{aligned} \min_{p \in P} \quad & b(p) = \sum_{k=1}^n b_k^y \\ \text{s.t.} \quad & \text{(C1)} : d(p) = \sum_{k=1}^{n+1} (t_{x,y}^k + t_k^y) \leq T_d \\ & \text{(C2)} : f_{v_1} < f_0, f_{v_2} < f_0 \end{aligned} \quad (15)$$

where p is the path from node S to node D , and P is the set of all possible paths, $p \in P$. (C1) is to ensure that the time consumed in executing the tasks is within the deadline. (C2) indicates that any platoon member provides less computational resources than the MEC server. Since each task has $m + 1$ offloading decision options, there are $(m + 1)^n$ possible options for the offloading strategy.

5. Optimal Offloading Decision for Collaborative Task Execution

In this section, we derive an optimal solution for Equation (15) and develop an optimal offloading decision for collaborative task execution. The constrained optimization problem in Equation (15) has been proved to be NP-complete [32].

5.1. Optimal Offloading Based on LARAC

It has been previously proposed that the constrained shortest path problem can be solved by LARAC algorithm [33]. We first denote a LARAC function

$$L(\lambda) = \min_{p \in P} [b_\lambda(p)] - \lambda T_d \quad (16)$$

where $b_\lambda(p) = b(p) + \lambda d(p)$ and λ is the Lagrangian multiplier. By using the Lagrangian duality principle, we can prove $L(\lambda) \leq b(p^*)$.

Next, we employ Algorithm 1 to find the path with smallest b_λ between S and D . In Algorithm 1, the *PathAlgorithm* is a process (e.g., Bellman–Ford algorithm [34]) to find the shortest path between S and D with the cost C . The details of the *PathAlgorithm* adopted in this paper will be explained in Section 5.2. If we can find the minimum-cost path under all constraints, this path will be the offloading strategy, otherwise update p_d and p_b repeatedly to find the optimal λ .

Algorithm 1. Find minimum-cost path of b_λ for collaborative task execution.

```

1. Input:
2.  $S, D, T_d$ 
3.  $p_b \leftarrow PathAlgorithm(S, D, b)$ 
4. If  $d(p_b) \leq T_d$  then
5. return  $p_b$ 
6. end if
7.  $p_d \leftarrow PathAlgorithm(S, D, d)$ 
8. If  $d(p_d) > T_d$  then
9. return "There is no solution"
10. end if
11. while true do
12.  $\lambda \leftarrow \frac{b(p_b) - b(p_d)}{d(p_d) - d(p_b)}$ 
13.  $p_\lambda \leftarrow PathAlgorithm(S, D, b_\lambda)$ 
14. If  $b_\lambda(p_\lambda) = b_\lambda(p_b)$  then
15. return  $p_d$ 
16. else
17. if  $d(p_\lambda) \leq T_d$  then
18.  $p_d \leftarrow p_\lambda$ 
19. else
20.  $p_b \leftarrow p_\lambda$ 
21. end If
22. end If
23. end while
24. Output:  $p_\lambda^*$ 

```

The computational complexity of the dynamic programming algorithm depends on the number of nodes, namely $O(|V|)$. In addition, as shown in Reference [35], the Lagrangian multiplier of optimum is obtained after $O(|N| \log^2 |N|)$ iterations, so the overall computational complexity of the Algorithm 1 is $O(|V| |N| \log^2 |N|)$, namely $O(n^2 \log^2 n)$.

Although the algorithm cannot guarantee to find the optimal path, it can obtain a lower bound of the optimal solution. Moreover, its running time is shown to be polynomial [33].

5.2. Dynamic Programming Algorithm

In order to apply the Algorithm 1 to the optimal task offloading decision, we need to find the shortest path according to the task execution cost, execution time and aggregation cost. Specifically, we treat all tasks as a multistep process with chain structure, namely a multistep decision process.

The state transition process for the optimal offloading decision of collaborative task execution is depicted in Figure 6. State 0 and state $n + 1$ represent the start and end of the whole application execution respectively.

We denote r_k as the location identifier of the task k . For instance, $r_k = 1$ indicates that the task k is executed on Vehicle 1 in the platoon. r_k keeps tracking the position of the application tasks, with 1 to m indicating the platoon members and 0 for the MEC server, as shown in the following

$$r_k = \begin{cases} 0, & \text{MEC server;} \\ 1, & \text{Vehicle 1;} \\ 2, & \text{Vehicle 2;} \\ \dots & \\ m, & \text{Vehicle } m. \end{cases} \quad (17)$$

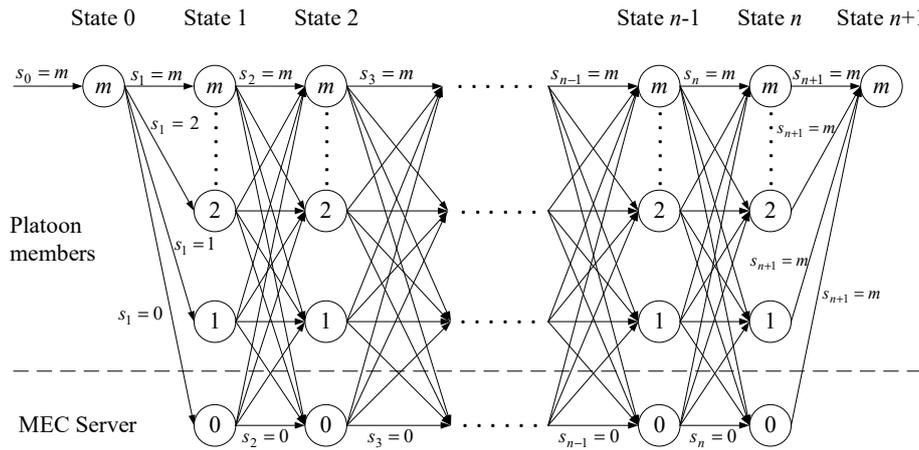


Figure 6. Task state transition process.

Task $n + 1$ is the output result of the application. Since the output result needs to be sent back to the starting point after the completion of task n , it can be obtained that $r_0 = m$ and $r_{n+1} = m$. The output result of task $n + 1$ does not need to be calculated, and the user does not need to purchase the computing resource, so we have b_{n+1}^m .

s_k is the decision variable for state k , indicating the choice of which vehicle the task k should be offloaded to, shown as follows

$$s_k = \begin{cases} 0, & \text{MEC server execution;} \\ 1, & \text{Vehicle 1 execution;} \\ 2, & \text{Vehicle 2 execution;} \\ \dots & \\ m, & \text{Vehicle } m \text{ execution.} \end{cases} \quad (18)$$

s_0 is the task initiation decision and s_{n+1} denotes the last decision of the application, that is $s_0 = s_{n+1} = m$. The aim is to find an optimal offloading decision strategy set $S^* = \{s_0, s_1, \dots, s_{n+1}\}$.

We next try to find the minimum cost, time delay, and aggregation cost in Figure 6, based on the established iterative equations, respectively.

First, we denote $G_{k-1}(r_{k-1})$ as the minimum cost from task $k - 1$ to task $n + 1$. $G_0(r_0)$ is the minimum cost for all tasks of the application, given $G_{n+1}(r_{n+1}) = 0$. On this basis, we establish an iterative equation of the latter term with the minimum cost. Knowing $G_k(r_k)$ at state k for location r_k , we obtain every decision at state $k - 1$ so that the cost from state $k - 1$ to state $n + 1$ is minimized. The backward iteration equation with minimum cost is represented by

$$\begin{aligned} G_{k-1}(r_{k-1}) &= \min_{s_k} [b_k(r_{k-1}, s_k) + G_k(r_k)] \\ G_{n+1}(r_{n+1}) &= 0 \end{aligned} \quad (19)$$

where $b_k(r_{k-1}, s_k)$ refers to the cost to be paid for making the offloading decision s_k after task $k - 1$ at position r_{k-1} is completed. Both $G_k(r_k)$ and $b_k(r_{k-1}, s_k)$ are known. The latter one can be obtained using (6) and (8) when the task is offloaded, that is, $b_k(r_{k-1}, s_k) = b_k^y$ and $y = s_k$. The system state starts from the state n , and the value of the $G_n(r_n)$ can be computed from the initial condition $G_{n+1}(r_{n+1})$ given in the state $n + 1$. Repeat this process for numerical iteration, and we find the optimal objective function value, optimal decision, and optimal path for the entire multistep decision problem in reverse order.

Second, let $H_{k-1}(r_{k-1})$ denote the minimum completion time from task $k - 1$ to task $n + 1$. $H_0(r_0)$ is the minimum task completion time for all tasks of the application, given $H_{n+1}(r_{n+1}) = 0$. Knowing $H_k(r_k)$ at state k for location r_k , we obtain every decision at state $k - 1$ so that the time delay from state

$k - 1$ to state $n + 1$ is minimized. The backward iteration equation with minimum task completion time is expressed by

$$\begin{aligned} H_{k-1}(r_{k-1}) &= \min_{s_k} [t_k(r_{k-1}, s_k) + H_k(r_k)] \\ H_{n+1}(r_{n+1}) &= 0 \end{aligned} \quad (20)$$

where $t_k(r_{k-1}, s_k)$ refers to the completion time of task k being offloaded to s_k after task $k - 1$ at position r_{k-1} is executed. Both $H_k(r_k)$ and $t_k(r_{k-1}, s_k)$ are known. The latter one can be obtained using (5), (7), (9) and (11) when the task is offloaded, that is, $t_k(r_{k-1}, s_k) = t_{x,y}^k + t_k^y$ and $y = s_k$.

Third, let $J_{k-1}(r_{k-1})$ be the minimum aggregated cost from task $k - 1$ to task $n + 1$. $J_0(r_0)$ is the minimum aggregated cost for all tasks of the application, given $J_{n+1}(r_{n+1}) = 0$. Knowing $H_k(r_k)$ at state k for location r_k , we obtain every decision at state $k - 1$ so that the aggregated cost from state $k - 1$ to state $n + 1$ is minimized. The backward iteration equation with minimum task aggregated cost is represented by

$$\begin{aligned} J_{k-1}(r_{k-1}) &= \min_{s_k} [b_k(r_{k-1}, s_k) + \lambda t_k(r_{k-1}, s_k) + J_k(r_k)] \\ J_{n+1}(r_{n+1}) &= 0 \end{aligned} \quad (21)$$

We use iterative Equations (19)–(21) to implement the processes of *PathAlgorithm(S, D, b)*, *PathAlgorithm(S, D, d)* and *PathAlgorithm(S, D, b_λ)* to find the minimum cost, time delay and aggregated cost, respectively. Finally, the minimum-cost offloading decision strategy with time delay constraints is obtained in Algorithm 1.

The proposed approach is designed for the applications considered. This is not the only scenario that can be used. Other scenarios, such as an individual vehicle or a cluster of vehicles in the cellular network, also can use the proposed method.

6. Numerical Analysis

In this section, the performance of the proposed task offloading decision strategies for an MEC-assisted platoon is evaluated.

6.1. Application Profile

We evaluate the efficiency of our algorithm in a cellular network composed of a BS with an MEC server. The distance between the leader and BS is $\eta = 0.5$ km. The distance between the adjacent vehicular transmitter and the receiver is $\mu = 0.008$ km. The platoon consists of nine members (i.e., $m = 9$) and the distance between vehicles is fixed in the platoon. Assume that the antenna position of each vehicle is the same. Assume that the antenna position of each vehicle is the same. The communication mode is deployed in accordance with the relevant rules of IEEE 802.11p standard and LTE-V. The bandwidth and transmission power for V2V and V2I communications in the simulation are set according to the reference papers [31,36]. The parameters of computation resource are set as follows: $f_0 = 3000$ MHz, $f_1 = 100$ MHz, $f_2 = 650$ MHz, $f_3 = 600$ MHz, $f_4 = 620$ MHz, $f_5 = 700$ MHz, $f_6 = 800$ MHz, $f_7 = 660$ MHz, $f_8 = 1000$ MHz, $f_9 = 550$ MHz. The other parameters in the simulation are summarized in Table 1.

Table 1. Simulation parameters.

Parameters	Value
The number of platoon members	9
Bandwidth for V2V communication	$B_1 = 20$ MHz
Bandwidth for V2I communication	$B_2 = 100$ MHz
The noise power	$N_0 = -174$ dBm/Hz
The transmission power of vehicle	$\xi_v = 23$ dBm
The transmission power of BS	$\xi_{MEC} = 46$ dBm
The distance between the adjacent transmitter and the receiver	$\mu = 0.008$ km
The distance between the leader and BS	$\eta = 0.5$
the computation resource cost of vehicle	$\alpha_v = 1$

6.2. Minimum-Cost Decision Strategy

We consider a mobile application that consists of 12 tasks. The deadline of the application is 0.3 s, the computation resource cost of MEC server is $\alpha_{\text{MEC}} = 0.9$ and the offloading decisions in detail are shown in Figure 7. In these four figures, the red line represents the tasks offloading decision S^* , which is the shortest path with the lowest cost and within the deadline. The NeededCycles means the computing workload. The task context includes two parts: the NeededCycles, whose unit is cycle, and the DataSize, whose unit is Kilobyte (kb). Figure 7 indicates different offloading strategies based on the proposed algorithm for tasks with different computing workloads and data sizes. Specifically, in Figure 7a, the amount of computation cycles required for each task of the application is relatively small and transmission data size of the tasks is relatively large. We get $S^* = \{9, 6, 6, 9, 9, 6, 6, 6, 6, 6, 6, 9\}$, because the transmission data rate between vehicles is much faster than that between leader and BS, and the large data size of the tasks limits the task execution on the MEC server. If the task is to be offloaded to the MEC server, it will lead to a large transmission delay. In Figure 7b, $S^* = \{9, 8, 0, 8, 0, 0, 8, 8, 0, 8, 9\}$. Note that the size of the transmitted data is relatively small, and the required computation cycles for each task increase. In order to satisfy the deadline constraint of the application, a task with high required computation cycles should be offloaded to the MEC server. In Figure 7c, $S^* = \{9, 4, 4, 7, 7, 7, 1, 8, 8, 1, 8, 9\}$. Each task requires a relatively small amount of computation cycles. To minimize the cost, the MEC server with higher computing power does not need to provide computational assistance to the task execution. In Figure 7d, $S^* = \{9, 8, 8, 0, 0, 0, 0, 8, 8, 8, 9\}$. Note that last four tasks are executed in the platoon to avoid high data transmission delays from the BS to the leader. These four cases show that the MEC server and the BS can cooperate to complete the application tasks.

Figure 7 indicates that the computing workloads and the data size affect the offloading decisions. A task with low computing cycles and high size of data tends to be executed in the platoon instead of MEC server, because the impact of task transmission time is greater than that of computing time, and these kinds of tasks are more likely to be offloaded to the nodes with fewer computing resources for lower cost. On the contrary, a task with high computing cycles and low size of data tends to be executed at the nodes with more computing resources in the platoon or at the MEC server decided by the cost and deadline. Note that the tasks with high computing cycles and low size of data is offloaded to the MEC server, which do not always meet the requirements of system. For instance, if the penultimate task with high computing cycles and low size of data was offloaded to the MEC server to execute, and the last task with 0 computing cycle and high size of data must be executed at the platoon member. That will cause extremely large transmission time when the data of last task is transmitted back to the member from the BS so that execution time of all tasks may exceed the deadline. Our algorithm succeeds in avoiding this. In summary, the LARAC algorithm can offload the tasks reasonably, and the task execution deadline can be satisfied.

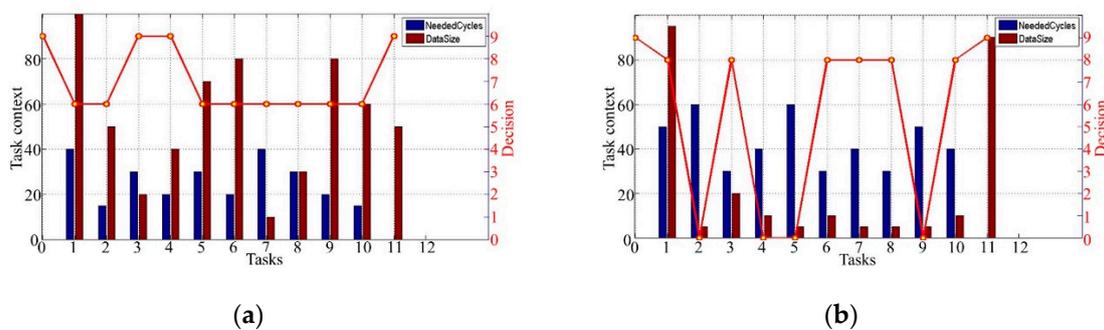


Figure 7. Cont.

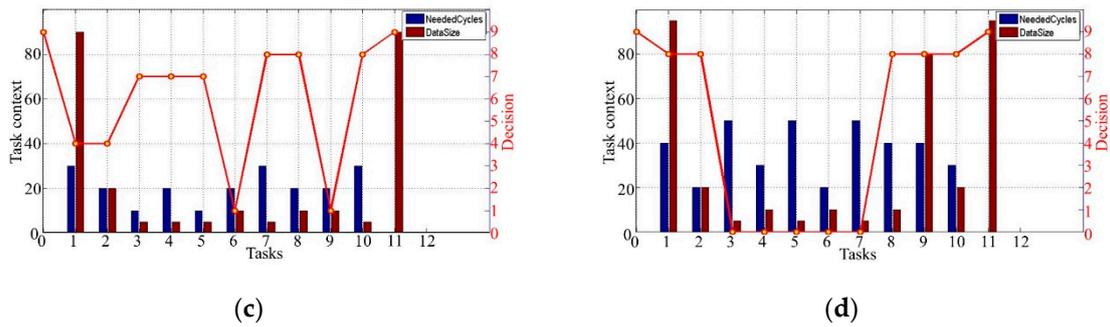


Figure 7. Task offloading decisions. (a) low computing cycles with high size of data; (b) high computing cycles with low size of data; (c) low computing cycles with low size of data; (d) high computing cycles with high and low size of data.

We next evaluate the effect of α_{MEC} on the tasks offloading decisions. The NeededCycles and the DataSize of these four subfigures in Figure 8 are the same. In Figure 8a, the cost of offloading the tasks to the MEC server is greater than that of any member of the platoon. To meet the deadline of the application, Tasks 6 and 7 must be offloaded to the MEC server for shorter execution time. In Figure 8b, we can see that, besides Tasks 6 and 7, Tasks 2, 8, and 9 are offloaded to the MEC server to execute. Similarly, in Figure 8c,d, we can see that more tasks are offloaded to the MEC server compared with that in Figure 8a. This is because, as α_{MEC} getting smaller, more of the tasks are offloaded to the MEC server for saving the cost. The proposed algorithm in this paper considers the tasks offloading strategy with minimum cost. As α_{MEC} becomes smaller, the cost of offloading tasks to the MEC server decreases gradually, so that more tasks will be offloaded to the MEC server.

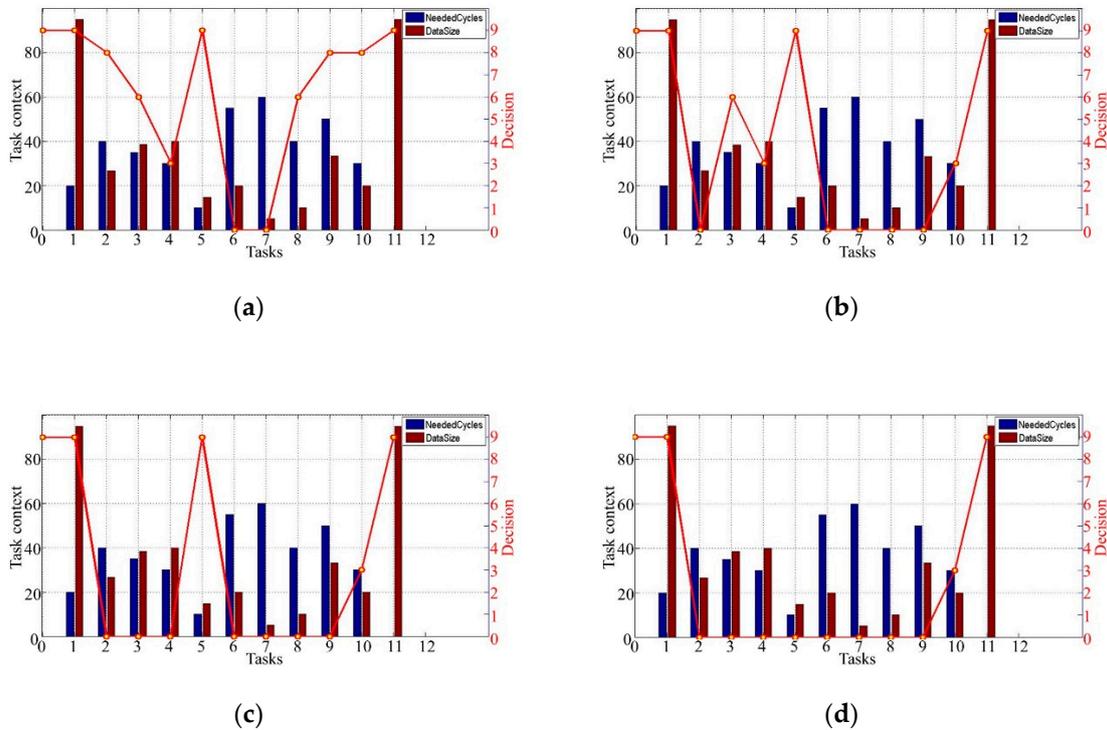


Figure 8. Task offloading decisions. (a) $\alpha_{MEC} = 0.4$; (b) $\alpha_{MEC} = 0.3$; (c) $\alpha_{MEC} = 0.2$; (d) $\alpha_{MEC} = 0.1$.

6.3. Comparison of Execution Patterns

Figure 9 shows the cost comparison of three execution modes (platoon execution, MEC execution and proposed collaborative execution) and the Constrained Bellman–Ford (CBF) algorithm [37],

$\alpha_{\text{MEC}} = 0.4$. The CBF algorithm could also be adopted to solve this problem, but its computational complexity is $O(n^3)$, which is larger than $O(n^2 \log^2 n)$ in this paper. Additionally, the CBF algorithm gets the optimal solution, while the LARAC algorithm gets the approximate solution. The data sizes and the demanded computation cycles are as follows [19]: $\{d_k\} = \{0\ 100\ 40\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 1\ 100\}$ kb, Mcycle. First, compared with MEC server execution, the proposed collaborative execution strategy can greatly reduce the cost. Second, collaborative task execution is more flexible than MEC server execution. Due to the low transmission rate between the leader and the BS, the high transmission delay will be caused by the MEC server execution when the data size is large. Therefore, the tasks with a large amount of transmitted data cannot be completed within the deadline in MEC server execution mode. Third, only collaborative task execution can complete the application less than the deadline 0.21 s, and it costs less than that in the MEC server execution mode. Fourth, with increase of the deadline, the cost of platoon task execution and collaborative task execution are approximately same, because the task execution does not require the participation of MEC server with the release of the deadline. The computational resources provided in the platoon are sufficient to enable the application to be completed within the deadline. The results of the local execution were not drawn, because 3.3 s is the minimum deadline, far from meeting the requirements of the application time constrained. Fifth, we can see that the cost of the collaborative task execution is very close to that of the CBF algorithm, which verifies the proposed algorithm can be well applied in this scenario.

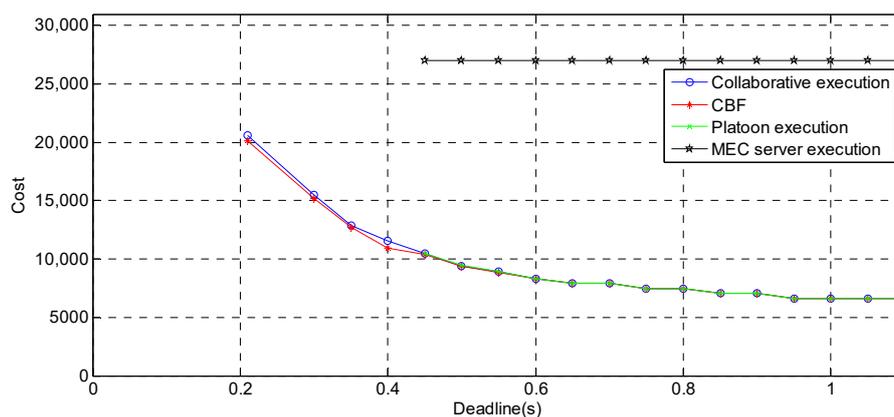


Figure 9. Deadline vs. cost.

7. Conclusions and Future Works

In this paper, the procedure of cooperative tasks execution for an MEC-assisted platoon within the execution deadline is studied. The task offloading decision problem is transformed into the shortest path problem in a directed acyclic graph. We employ the “LARAC” algorithm to obtain the optimal decision strategy for task offloading. The results show that there exists more than one migration between the platoon members and the MEC server, and all the members have the opportunity to participate in the tasks execution. Moreover, the proposed collaborative task execution strategy can greatly reduce task execution cost and execution time.

For our future work, we will consider tasks offloading for a more general case of vehicular network, such as a cluster of vehicles in a cellular network. In addition, tasks offloading of crossing cell—which depends on the deployments of the MEC servers—will be achieved in future discussion.

Author Contributions: Conceptualization, X.F.; Funding acquisition, T.C. and Q.C.; Project administration, T.C.; Supervision, T.C.; Validation, X.F.; Writing—original draft, X.F. and C.C.; Writing—review & editing, Q.C. and K.S.K.

Funding: This work was supported in part by the National Natural Science Foundation of China (6157073), the National Science and Technology Specific Project of China (2016ZX03001010-004) and the special fund of Chongqing key laboratory (CSTC), the Program for Changjiang Scholars and Innovative Research Team in University (IRT_16R72) and the Doctor Start-up Funding of CQUPT (A2016-83).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Beck, M.T.; Werner, M.; Feld, S.; Schimper, T. Mobile Edge Computing: A Taxonomy. In Proceedings of the Sixth International Conference on Advances in Future Internet, Lisbon, Portugal, 16–20 November 2014.
2. Yan, M.; Song, J.; Yang, P.; Tang, Y. Distributed adaptive sliding mode control for vehicle platoon with uncertain driving resistance. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 9396–9400.
3. Axelsson, J. Safety in Vehicle Platooning: A Systematic Literature Review. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1033–1045. [[CrossRef](#)]
4. El-Zaher, M.; Dafflon, B.; Contet, J.-M.; Gechter, F. Vehicle Platoon Control with Multi-configuration Ability. In Proceedings of the International Conference on Computational Science (ICCS 2012), Omaha, NE, USA, 4–6 June 2012.
5. Wang, L. The Embedded ARM user Program design of Vehicle Terminal. In Proceedings of the 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, 26–28 February 2010; pp. 563–567.
6. Shah, S.A.A.; Ahmed, E.; Imran, M.; Zeadally, S. 5G for Vehicular Communications. *IEEE Commun. Mag.* **2018**, *56*, 111–117. [[CrossRef](#)]
7. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2322–2358. [[CrossRef](#)]
8. Kemp, R.; Palmer, T.; Kielmann, T.; Seinstra, F.; Drost, N.; Maassen, J.; Bal, H. eyeDentify: Multimedia Cyber Foraging from a Smartphone. In Proceedings of the 2009 11th IEEE International Symposium on Multimedia, San Diego, CA, USA, 2009; pp. 392–399.
9. Al-Shuwaili, A.; Simeone, O. Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wireless Commun. Lett.* **2017**, *6*, 398–401. [[CrossRef](#)]
10. TR22.885 V14.0.0. Study on LTE support for Vehicle to Everything (V2X) services. Available online: <https://www.tech-invite.com/3m22/tinv-3gpp-22-885.html> (accessed on 10 September 2017).
11. Li, L.; Li, Y.; Hou, R. A Novel Mobile Edge Computing-Based Architecture for Future Cellular Vehicular Networks. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
12. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient Multi-User Computation Offloading for Mobile Edge Cloud Computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [[CrossRef](#)]
13. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [[CrossRef](#)]
14. Jararweh, Y.; Doulat, A.; Darabseh, A.; Alsmirat, M.; Al-Ayyoub, M.; Benkhelifa, E. SDMEC: Software Defined System for Mobile Edge Computing. In Proceedings of the 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), Berlin, Germany, 4–8 April 2016; pp. 88–93.
15. Mäkinen, O. Streaming at the Edge: Local Service Concepts Utilizing Mobile Edge Computing. In Proceedings of the 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, Cambridge, UK, 9–11 September 2015; pp. 1–6.
16. Dinh, T.Q.; Tang, J.; La, Q.D.; Quek, T.Q.S. Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling. *IEEE Trans. Commun.* **2017**, *65*, 3571–3584.
17. Zhang, K.; Mao, Y.; Leng, S.; Maharjan, S.; Zhang, Y. Optimal delay constrained offloading for vehicular edge computing networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
18. Jia, M.; Cao, J.; Yang, L. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 27 April–2 May 2014; pp. 352–357.
19. Zhang, W.; Wen, Y.; Wu, D.O. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing. In Proceedings of the 2013 IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 190–194.
20. Cheng, Z.; Li, P.; Wang, J.; Guo, S. Just-in-time code offloading for wearable computing. *IEEE Trans. Emerg. Topics Comput.* **2015**, *3*, 74–83. [[CrossRef](#)]

21. Lyu, X.; Tian, H. Adaptive receding horizon offloading strategy under dynamic environment. *IEEE Commun. Lett.* **2016**, *20*, 878–881. [[CrossRef](#)]
22. Shao, C.; Leng, S.; Zhang, Y.; Vinel, A.; Jonsson, M. Performance Analysis of Connectivity Probability and Connectivity-Aware MAC Protocol Design for Platoon-Based VANETs. *IEEE Trans. Veh. Technol.* **2015**, *64*, 5596–5609. [[CrossRef](#)]
23. Jia, D.; Lu, K.; Wang, J. On the network connectivity of platoon-based vehicular cyber-physical systems. *Transp. Res. Part C Emerg. Technol.* **2014**, *40*, 215–230. [[CrossRef](#)]
24. Campolo, C.; Molinaro, A.; Araniti, G.; Berthet, A.O. Better Platooning Control Toward Autonomous Driving: An LTE Device-to-Device Communications Strategy That Meets Ultralow Latency Requirements. *IEEE Veh. Technol. Mag.* **2017**, *12*, 30–38. [[CrossRef](#)]
25. Shao, C.; Leng, S.; Zhang, Y.; Vinel, A.; Jonsson, M. Analysis of connectivity probability in platoon-based Vehicular Ad Hoc Networks. In Proceedings of the 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), Nicosia, Cyprus, 4–8 August 2014; pp. 706–711.
26. Khaksari, M.; Fischione, C. Performance analysis and optimization of the joining protocol for a platoon of vehicles. In Proceedings of the 5th International Symposium on Communications, Control and Signal Processing, Rome, Italy, 2–4 May 2012; pp. 1–6.
27. Wang, Y.; Sheng, M.; Wang, X.; Wang, L.; Li, J. Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling. *IEEE Trans. Commun.* **2016**, *64*, 4268–4282. [[CrossRef](#)]
28. IEEE Std. 802.11p: Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks—Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. 2010. Available online: https://standards.ieee.org/project/802_11bb.html (accessed on 8 December 2018).
29. RP-161894. LTE-based V2X Services, 3GPP. Available online: <https://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=730345> (accessed on 8 December 2018).
30. Karedal, J.; Czink, N.; Paier, A.; Tufvesson, F.; Molisch, A.F. Path loss modeling for vehicle-to-vehicle communications. *IEEE Trans. Veh. Technol.* **2011**, *60*, 323–328. [[CrossRef](#)]
31. Lyu, X.; Tian, H.; Sengul, C.; Zhang, P. Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds. *IEEE Trans. Veh. Technol.* **2017**, *66*, 3435–3447. [[CrossRef](#)]
32. Wang, Z.; Crowcroft, J. Quality-of-service routing for supporting multimedia applications. *IEEE J. Sel. Areas Commun.* **1996**, *14*, 1228–1234. [[CrossRef](#)]
33. Jüttner, A.; Szviatovski, B.; Mecs, I.; Rajko, Z. Lagrange relaxation based method for the QoS routing problem. In Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Society, Anchorage, AK, USA, 22–26 April 2001; pp. 859–868.
34. Yuan, X. Heuristic algorithms for multiconstrained quality-of-service routing. *IEEE/ACM Trans. Networking* **2002**, *10*, 244–256. [[CrossRef](#)]
35. Jüttner, A. On resource constrained optimization problems. In Proceedings of the 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, Budapest, Hungary, 3–6 June 2005.
36. Miettinen, A.P.; Nurminen, J.K. Energy efficiency of mobile clients in cloud computing. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, Berkeley, CA, USA, 22–25 June 2010.
37. Widyono, R. *The Design and Evaluation of Routing Algorithms for Realtime Channels*; Technical Report TR-94-024; University of California: Berkeley, CA, USA, 1994.

