Minimum Cost Offloading Decision Strategy for Collaborative Task Execution of Platooning Assisted by MEC

Taiping Cui(✉), Xiayan Fan, Chunyan Cao, and Qianbin Chen

School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
{cuitp, chenqb}@cqupt.edu.cn,
Fanxiayan, Caocyan@yeah.net

Abstract. In this paper, we study the offloading decision of collaborative task execution between platoon and MEC (Mobile Edge Computing) 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 task offloading and meet the deadline of tasks execution. We transform the cost minimized task decision problem into the shortest path problem, which is limited by the deadline of the tasks on a directed acyclic graph. The classical LARAC algorithm is used 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 task in cooperation with the MEC server. In addition, compared with different execution models, the optimal offloading decision for collaborative task execution can significantly reduce the cost of task execution and meet lower deadlines.

Keywords: Platooning · Mobile edge computing · Offloading decision

1 Introduction

Platooning realizes the reduction of fuel consumption and gas emission, as well as safe and efficient transportation in the context of intelligent transportation system (ITS). Generally, the platoon is consisted of two parts: one is the leader and the second are the members of the platoon (including the tail vehicle, the relay vehicle and the controller). The higher the frequency of the vehicle information exchange in the platoon, the faster the mobile response of the members in the platoon, the more the status of the platoon instability can be avoided.

This work was supported in part by the National Natural Science Foundation of China (61401053, 6157073), and the Doctor Start-up Funding of CQUPT (A2016-83).

X. Liu et al. (Eds.): ChinaCom 2018, LNICST 262, pp. 104–115, 2019.
https://doi.org/10.1007/978-3-030-06161-6_10

Vehicle terminals have been widely deployed in the automotive industry. More novel and attractive vehicle services have attracted more and more people to use them. In the future, cars will be equipped with AR (Augmented Reality) applications that allow drivers to observe the surroundings of vehicles in windowless vehicles [1]. These types of vehicle applications are typical computing resource-hungry services, requiring high density computing resources and computing costs. Therefore, the tension between computing resource-hungry applications and vehicle terminals with limited computing resources poses a major challenge to the development of mobile platforms [2].

MEC provides a promising solution to this challenge and extends the capabilities of vehicle terminals, by providing additional computing, storage, and bandwidth resources in an on-demand manner. For example, there is a paper on energy consumption [3], which proposes collaborative execution between the end-user and the cloud, and accomplishes the task with minimum energy consumption within the task deadline. In addition, some people consider offloading decision from the point of view of the servers [4], set the price for the unit resource provided to each vehicle user, and use the gain function of task offloaded to maximize the profit of the server.

In this paper, we consider the platooning scenario with MEC server to complete the task offloading. Specifically, within the deadline of the task, the resource cost price is used to determine whether the tasks are offloaded to the other members of the platoon or the MEC server, or not. We aim to find an optimal collaborative offloading decision between each member of the platoon and the MEC with minimum resource cost price within the deadline. Mathematically, we model a minimum cost 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 a suitable result of the constrained optimization problem.

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

2 System Model

Suggest that the MEC server coexists with the base station (BS). Because the platoon controller controls and manages the whole platoon [5], when the platoon members communicate with the BS, they need to transmit the messages to the BS through the controller, and the members in a platoon can communicate with others directly [6]. Position of the controller in the platoon is not clearly defined. For the sake of simplification, the current researches directly select the leader as platoon controller. Also in this paper, we do not discuss in detail how to choose.

2.1 Task Model

Figure 1 illustrates the task model in a linear topology. 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. Figure 1 shows that there are $n+1$ tasks in an application. Define the needed computation cycles of the task k as ω_k, $k = 0, 1, 2, ..., n+1$.

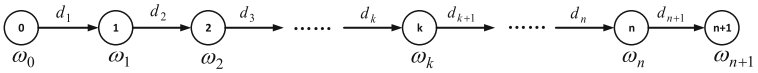

Fig. 1. Task model in a linear topology

2.2 Path Loss Model

When one vehicle is served by another, the path loss model [7] between the two vehicles is defined as $PL_v(l_{x,y}) = 63.3 + 17.7log_{10}(l_{x,y})$. And the path loss model [8] between vehicle and BS is defined as $PL_{MEC}(l_{x,y}) = 128.1 + 37.5log_{10}(l_{x,y})$. $l_{x,y}$ is distance in kilometers. Here, we do not consider fast fading and shadow fading. The carrier frequency used in V2I (Vehicle to Infrastructure) communication is 2 GHz and V2V communication is 5.9 GHz, so there is no interference between them, and we think of the whole channel as an ideal channel.

The MEC server numbered 0, a leader vehicle numbered 1, and the sequence number of vehicles behind it increases in turn. There are a total of m vehicles in the paltoon. $l_{x,y}$ is the transmission distance from x to y, $x \in \{0, 1, 2, ..., m\}$, $y \in \{0, 1, 2, ..., m\}$, assuming the antenna position of each vehicle is the same and that the distance between vehicles is fixed in the platoon, so $l_{x,y} = l_{y,x}$.

Considering that under ideal conditions all the vehicles in the platoon are of the same length and the same spacing so the distance between the signal receiver and the transmitter is the length of the vehicle plus the vehicle spacing. So we can get $l_{v_1,v_2} = \mu|v_1 - v_2|$, where $v_1 \in \{1, 2, ..., m\}$ is for the current vehicle number about task $k-1$ and $v_2 \in \{1, 2, ..., m\}$ is for the destination vehicle of offloading decision about task k. Specially, consider offloading tasks to the MEC server, and the distance between the leader and the BS is set to η, so we obtain $l_{1,0} = \eta$. When the task is offloaded to the MEC server, the change of distance between the leader and the BS has little effect on the path loss, so we set $l_{1,0} = l_{0,1}$. Only the leader can communicate with the BS, and the platoon members communicate with the BS through the leader.

2.3 Task Execution Model

Platoon execution. Assume that the tasks is initiated by any member of the platoon. The task execution of the platoon includes the local task execution and the tasks platoon members execution. If the task k is offloaded to execute by a member of the platoon, the completion time of the task k is defined as $t_k^{v_2} = \frac{\omega_k}{f_{v_2}}$, and the cost it has to pay is $b_k^{v_2} = \alpha_v f_{v_2}$, 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. It is assumed that the f_{v_2} is fixed during the execution of the task.

MEC server execution. If the task k is executed in the MEC server, the platoon members are idle during the execution of the task. Define the computing completion time of the task k offloaded to the MEC server as $t_k^0 = \frac{\omega_k}{f_0}$, and the cost of offloading the task is $b_k^0 = \alpha_{MEC} f_0$. f_0 is the computation resource that the MEC server can provide, and the computation resource cost of each unit provided by the MEC server is α_{MEC}. Note $f_0 > f_1, f_0 > f_2, ..., f_0 > f_m$, speculate that the CPU speed of the MEC server is faster than that of any platoon member.

Platoon data transmission. If the task k is executed inside the platoon, the data needs to be offloaded to the destination vehicle before it is executed. If the task is calculated on the current vehicle, there is no need to transmit the data. Define the time of data transmission that offloads the task k from vehicle v_1 to vehicle v_2 as $t_{v_1,v_2}^k = \frac{d_k}{R_{v_1,v_2}^k}$, R_{v_1,v_2}^k is the data transmission rate from vehicle v_1 to vehicle v_2. We have $R_{v_1,v_2}^k = B_1 log_2(1 + \frac{\zeta_v PL_v(l_{v_1,v_2})}{N_0})$. ζ_v is the signal transmission power of the vehicle, N_0 is the noise power and B_1 is the bandwidth used for V2V (Vehicle to Vehicle) communication. If $v_1 = v_2$, $t_{v_1,v_2} = 0$.

MEC server data transmission. If the task k needs to be executed on the MEC server, the time that the data is transferred from the vehicle v_1 to the MEC server and from the MEC server to the vehicle v_2 are $t_{v_1,0}^k = \frac{d_k}{R_{v_1,1}^k} + \frac{d_k}{R_{1,0}^k}$ and $t_{0,v_2}^k = \frac{d_k}{R_{1,v_2}^k} + \frac{d_k}{R_{0,1}^k}$ respectively. $R_{1,0}^k$ and $R_{0,1}^k$ also indicating the transmission rate of the data from the leader to the MEC server and from MEC server to leader respectively. We have $R_{1,0}^k = B_2 log_2(1 + \frac{\zeta_v PL_{MEC}(l_{1,0})}{N_0})$ and $R_{0,1}^k = B_2 log_2(1 + \frac{\zeta_{MEC} PL_{MEC}(l_{0,1})}{N_0})$. ζ_{MEC} is the signal transmission power of the BS and B_2 is the bandwidth used for V2I communication. The connection between BS and MEC is wired, and the time of wired transmission is ignored here. The same as before, $t_{0,0}^k = 0$.

In this paper, the following assumptions are used to model the practical problem of collaborative task execution. First, the tasks of the application needs

to have been replicated on the other platoon members or the MEC server before it is executed. Second, the first and last task must be executed in the same vehicle in the platoon.

3 Delay Constrained Offloading

The tasks of an application is modeled by the directed acyclic graph $G = (V, C)$ between the platoon members and the MEC server. V represents the finite node set, and C represents the edge set. The task execution flow shown in Fig. 2. shows that S is the starting point of the task (Assume that the vehicle m is the requestor of the task) and D is the task destination node. An application contains $n+2$ tasks. Except for the initial task and the last task, all the platoon members have the opportunity to execute the other n tasks. The weights of the edges of nodes x and y are a non-negative value, that is $c^k_{x,y}$, and each edge corresponds to a task offloading decision s_k. This weight value includes the cost of offloading the task and the time it takes to execute after offloading the task. Specifically, if the weight value of the edge is considered to be a cost price and the task k needs to be offloaded from x to y to be executed, we obtain the weight $c^k_{x,y} = b^y_k$. And if the weight value is considered to be a time delay, we obtain the weight value $c^k_{x,y} = t^y_k + t^k_{x,y}$ that is the sum of the computation time and the transmission time of the task. So the two directed acyclic graphs with respect to the cost price and time delay can be obtained. s_k is defined as a decision variable for state k, indicating the choice of which vehicle the task k should be offloaded to. s_0 denotes the task initiation decision and s_{n+1} denote the end decision of the task result, $s_0 = s_{n+1}$. This goal is to find an optimal offloading decision strategy $S^* = \{s_0, s_1, ..., s_{n+1}\}$.

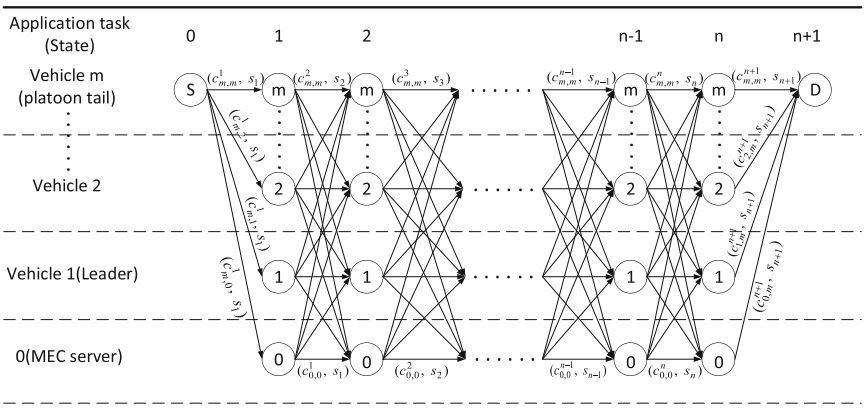

Fig. 2. The task execution flow and state transition procedure of collaborative offloading decision

Under this framework, we can transform the task optimal offloading decision into a shortest path problem to find the path with minimum cost between nodes

S and D. And it is constrained by the deadline time of the task, and the time delay path of the task can only be less than or equal to T_d. If the task time delay of a path p satisfies the constrained condition, p is an appropriate path. A path p^* is an optimal path with the minimum path cost of all appropriate paths. We can Mathematically formulate this problem as a constrained shortest path problem:

$$\begin{aligned} \min_{p \in P} \quad & b(p) = \sum_{k=1}^{n} b_{x,y}^{k} \\ \text{s.t.} \quad & (\text{C1}): \quad d(p) = \sum_{k=1}^{n+1} (t_{x,y}^{k} + t_{k}^{y}) \leq T_d \\ & (\text{C2}): \quad f_{v_1} < f_0, \ f_{v_2} < f_0, \quad v_1 \in x, \ v_2 \in y \\ & (\text{C3}): \quad \alpha_v f_{v_2} < \alpha_{MEC} f_0 \end{aligned} \tag{1}$$

p is a path from node S to D and P is a set of all possible paths. Since each task has $m+1$ offloading decisions options, there are $(m+1)^n$ possible options for the strategy. The constrained optimization problem has been proved to be NP-complete [9].

4 Optimal Offloading Decision for Collaborative Task Execution

4.1 Based on LARAC algorithm

This constrained shortest path problem can be solved by LARAC algorithm [12]. we first define a LARAC function $L(\lambda) = \min_{p \in P}[b_\lambda(p)] - \lambda T_d$ where $b_\lambda(p) = b(p) + \lambda d(p)$ and the Lagrangian multiplier is λ. By using the Lagrangian duality principle, we can obtain the proof of $L(\lambda) \leq b(p^*)$.

Then, we use algorithm 1 to find the path of smallest b_λ between S and D. In algorithm 1, *PathAlgorithm* is a procedure of finding a shortest path of the cost C. If we find the minimum cost path within the deadline, this path is the offloading strategy, or we update p_b and p_d repeatedly to get an optimal λ. Although the algorithm cannot guarantee to find the optimal path, it can obtain a lower bound of the optimal solution. And its running time is shown to be polynomial [10].

Algorithm 1 Finding minimum cost path of b_λ for collaborative task execution

Input: S, D, T_d
$p_b \leftarrow PathAlgorithm(S, D, b)$
if $d(p_b) \leq T_d$ **then**
  return p_b
end if
$p_d \leftarrow PathAlgorithm(S, D, d)$
if $d(p_d) > T_d$ **then**
  return "There is no solution"
end if
while *true* **do**
  $\lambda \leftarrow \frac{c(p_b)-c(p_d)}{d(p_d)-d(p_b)}$
  $p_\lambda \leftarrow PathAlgorithm(S, D, b_\lambda)$
  if $b_\lambda(p_\lambda) = b_\lambda(p_d)$ **then**
    return p_d
  else
    if $d(p_\lambda \leq T_d)$ **then**
      $p_d \leftarrow p_\lambda$
    else
      $p_b \leftarrow p_\lambda$
    end if
  end if
end while
Output: p_λ^*

4.2 Dynamic Programming Algorithm for Optimal Offloading Decision

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

The state transition process for the optimal offloading decision of collaborative task execution is shown in Fig. 4. State 0 and state $n+1$ represent the start and end of the whole application execution respectively. State k represents that the task k has been executed, and $k = 0, 1, 2, ..., n+1$. We define r_k as the location identifier of the task k that has been executed, so $r_k = 1$ indicates that the task k is executed at the position of the vehicle 1. Particularly, r_k keeps tracking the position of the task. Task $n+1$ is considered to be the output result of the application. Since the output result need to be sent back to the starting point after the completion of task n, it is assumed that the task is initiated from vehicle m, so it can be obtained that $r_0 = m$ and $r_{n+1} = m$ are the start and end point of the application task. The output result does not need to be calculated, and the user does not need to buy the computing resource so we have $b_{n+1}^m = 0$.

In the next part, we will find the minimum cost, time delay, and aggregation cost by using the established iterative equations, respectively.

First, we define $G_{k-1}(r_{k-1})$ as the minimum cost of 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 can establish an iterative formula of the latter term value for the minimum cost. Knowing $G_k(r_k)$ at state k for location r_k, we can obtain every decision at state $k-1$ so that the cost from state $k-1$ to state $n+1$ is minimized. The backward value iteration equation of minimum purchase cost is shown as follows:

$$\begin{aligned} G_{k-1}(r_{k-1}) &= \min_{s_k}[b_k(r_{k-1}, s_k) + G_k(r_k)] \\ G_{n+1}(r_{n+1}) &= 0 \end{aligned} \tag{2}$$

$b_k(r_{k-1}, s_k)$ refers to the cost to be paid for making the offloading decision s_k of task 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 values. The value of latter one can be obtained when the task is offloaded, $b_k(r_{k-1}, s_k) = b_k^y$, and $s_k = y$. The system state starts from the state n, and the value of the previous state $G_n(r_n)$ can be obtained from the initial condition $G_{n+1}(r_{n+1}) = 0$ given in the state $n+1$. Then repeat this procedure for numerical iteration, and you can find the optimal objective function value, optimal decision, and optimal path for the entire multistep decision problem in reverse order.

Second, we define $H_{k-1}(r_{k-1})$ as the minimum completion time of task $k-1$ to task $n+1$. $G_0(r_0)$ is for the minimum task completion time of all tasks, given $H_{n+1}(r_{n+1}) = 0$. The backward value iteration equation of minimum task completion time is shown as follows:

$$\begin{aligned} H_{k-1}(r_{k-1}) &= \min_{s_k}[t_k(r_{k-1}, s_k) + H_k(r_k)] \\ H_{n+1}(r_{n+1}) &= 0 \end{aligned} \tag{3}$$

Third, we define $J_{k-1}(r_{k-1})$ as the minimum aggregated cost of task $k-1$ to task $n+1$. $J_0(r_0)$ is the minimum aggregated cost for all tasks of the application, given $J_{n+1}(r_{n+1}) = 0$. The backward value iteration equation of minimum task aggregated cost is shown as follows:

$$\begin{aligned} J_{k-1}(r_{k-1}) &= \min_{s_k}[b_k(r_{k-1}, s_k) + \lambda d_k(r_{k-1}, s_k) + J_k(r_k)] \\ J_{n+1}(r_{n+1}) &= 0 \end{aligned} \tag{4}$$

We can use the iterative equations (2), (3), and (4) to implement the processes of $PathAlgorithm(S, D, b)$, $PathAlgorithm(S, D, d)$, and $PathAlgorithm(S, D, b_\lambda)$ to find the minimum cost, time delay and aggregated cost, respectively. Finally, the minimum cost offloading decision strategy is obtained in Algorithm 1.

5 Numerical Analysis

In this section, we evaluate the performance of task decision strategies for collaborative task execution. We use some of the parameters in [8,11] and communication between vehicles consider the use of carrier aggregation technology

[12] to increase bandwidth, as shown below: $B_1 = 20\,\text{MHz}$, $B_2 = 100\,\text{MHz}$, the thermal noise density $-174\,\text{dbm/Hz}$, $p_v = 23\,\text{dBm}$, $p_{MEC} = 46\,\text{dBm}$. Assume that the members of the platoon consist of nine vehicles, $\{f_v\} = \{3000\ 100\ 620\ 660\ 600\ 900\ 700\ 630\ 550\ 800\}$ MHz, $\alpha_v = 1$, $\alpha_{MEC} = 0.9$, $\mu = 0.008$, $\eta = 0.5$.

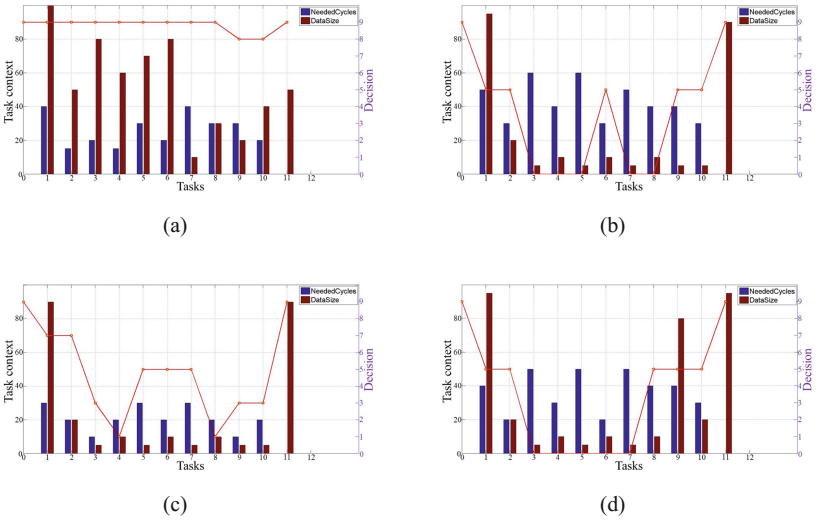

Fig. 3. Task offloading decision

We consider a mobile application that consists of 12 tasks, the application deadline is 0.4s, and the offloading decision and details are shown in Fig. 3. In Fig. 3a, the decision strategy is $S^* = \{9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 8, 9\}$ and the amount of computation cycles required for each task of the application is small, but relatively large transmission data size of the task limits the execution of the task on the MEC server, because the transmission data rate between vehicles is much faster than that between vehicles and BS. If the task is to be offloaded to the MEC server, it will lead to a large transmission delay. In Fig. 3b, the decision strategy is $S^* = \{9, 5, 5, 0, 0, 0, 5, 0, 0, 5, 5, 9\}$ and the size of the transmitted data is small, and the computation cycles of each task requirement is increased. In order to satisfy the time constrained requirement of the task, a task whose computation cycles is too high is offloaded to the MEC server. The reason why the first task is not offloaded to the MEC server is that an excessively high transmission delay will be caused. In Fig. 3c, the decision strategy is $S^* = \{9, 7, 7, 3, 1, 5, 5, 5, 1, 3, 3, 9\}$ and each task requires a small amount of computation cycles, so the MEC server doesn't need to provide computational assistance to the task execution. In Fig. 3d, the decision strategy is $S^* = \{9, 5, 5, 0, 0, 0, 0, 0, 5, 5, 5, 9\}$ and the last four tasks are executed in platoon to avoid high data transmission delays from the BS back to the vehicle. These

four cases show that the MEC server and the platooning can complete the task cooperatively. According to the different parameters of the task, the tasks can be offloaded reasonably, and the deadline of the task can be satisfied. In addition, there is a ratio between the computation cycles and the data size of a task, which should be greater than a threshold when a task needs to be offloaded to a MEC server; similarly, when a task needs to be offloaded to a vehicle in the platoon, the ratio of the computation cycles and the size of the data should also be greater than a threshold. We will find this threshold in future research.

In Fig. 4, we compare the cost under three execution modes and CBF (the Constrained Bellman-Ford [13]) algorithm, that is, platoon execution, collaborative execution, and the results obtained using the CBF algorithm. The parameters are set to: $\{d_k\} = \{100\ 40\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 1\ 100\}$ kb, $\{w_k\} = \{40\ 20\ 50\ 30\ 50\ 20\ 40\ 30\ 30\ 20\}$ $Mcycles$. First, compared with MEC server execution, collaborative execution can greatly reduce the cost when the task deadline is large. In most deadlines, collaborative task execution reduces task execution cost by more than four times. Second, collaborative task execution is more flexible than MEC server execution. That's because of the low transmission rate between the vehicle and the BS, the high transmission delay will be caused by the MEC server execution, so the tasks with a large amount of data transmission can not be completed within the deadline. Third, only collaborative task execution can complete the application within 0.25 s of the deadline, and it costs less than the MEC server to execute. Fourth, due to the increase of the deadline, the cost of platoon task execution and collaborative task execution is the same when the deadline is 0.45s, because after this deadline, the task execution does not require the participation of MEC server. The computational resources provided in the platoon are sufficient to enable the task to be completed within the deadline. The results of the local execution of the task were not drawn, because 3.3s is the minimum deadline, far from meeting the requirements of the task time constrained. Fifth, we can see that the collaborative task execution is very similar to the CBF algorithm, so the algorithm of collaborative task execution can be well applied in this scenario.

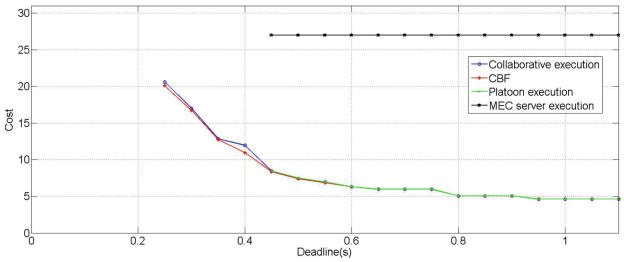

Fig. 4. Cost vs deadline

6 Conclusion

In this paper, the procedure of platoon and MEC server executing the task cooperatively within the task deadline is studied. We transform the task decision problem into the shortest path problem in a directed acyclic graph. We use the "LARAC" algorithm to obtain the optimal decision strategy for the tasks. Our research shows that there are more than one migration between the platoon members and the MEC service, and all the members have the opportunity to participate in the tasks execution. In addition, collaborative task execution can greatly reduce task execution cost and execution time.

In future research, the topological model of the tasks can be extended to various graphs (such as grid, tree, etc.). According to these structural characteristics, we will establish an optimal task decision strategy.

References

1. Shah, S.A.A., Ahmed, E., Imran, M., Zeadally, S.: 5G for Vehicular Communications. IEEE Commun. Mag. **56**(1), 111–117 (2018)
2. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017). Fourthquarter
3. Zhang, W., Wen, Y., Wu, D.O.: Energy-efficient scheduling policy for collaborative execution in mobile cloud computing. In: Proceedings IEEE INFOCOM, pp. 190–194. Turin (2013)
4. Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y.: Optimal delay constrained offloading for vehicular edge computing networks. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. Paris (2017)
5. Khaksari, M., Fischione, C.: Performance analysis and optimization of the joining protocol for a platoon of vehicles. In: 5th International Symposium on Communications. Control and Signal Processing, pp. 1–6. Rome (2012)
6. Shao, C., Leng, S., Zhang, Y., Vinel, A., Jonsson, M.: Analysis of connectivity probability in platoon-based Vehicular Ad Hoc Networks. In: International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 706–711. Nicosia (2014)
7. Karedal, J., Czink, N., Paier, A., Tufvesson, F., Molisch, A.F.: Path loss modeling for vehicle-to-vehicle communications. IEEE Trans. Veh. Technol. **60**(1), 323C328 (2011)
8. Lyu, X., Tian, H., Sengul, C., Zhang, P.: Multiuser joint task offloading and resource optimization in proximate clouds. IEEE Trans. Veh. Technol. **66**(4), 3435–3447 (2017)
9. Wang, Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. IEEE J. Sel. Areas Commun. **14**(7), 1228C1234 (1996)
10. Juttner, A., Szviatovski, B., Mecs, I., Rajk o, Z.: Lagrange relaxation based method for the qos routing problem. In: Proceedings of IEEE INFOCOM, vol. 2, pp. 859C868 (2001)
11. Miettinen, A.P., Nurminen, J.K.: Energy efficiency of mobile clients in cloud computing. In: Proceedings of 2nd USENIX Conference Hot Topics Cloud Computer, p. 4 (2010)

12. TD-LTE Carrier Aggregation White Paper. http://lte-tdd.org/. Accessed 12 Jun 2018
13. Widyono, R.: The design and evaluation of routing algorithms for realtime channels. Technical Report TR-94-024, University of California at Berkeley, June 1994