Reinforcement Learning based Computation Migration for Vehicular Cloud Computing

Fei Sun*, Nan Cheng†,Shan Zhang‡, Haibo Zhou§, Lin Gui*, and Xuemin (Sherman) Shen†
*Shanghai Jiao Tong University, China Email: {sf19912010, guilin}@sjtu.edu.cn
†University of Waterloo, Canada Email: {n5cheng, sshen}@uwaterloo.ca
‡Beihang University, China Email: zhangshan18@buaa.edu.cn
§Nanjing University, China Email: haibozhouuw@gmail.com

***Abstract*—By employing the exponentially increasing communication and computing capabilities of vehicles brought by the development of connected and autonomous vehicles, vehicular cloud computing (VCC) can improve the overall computational efficiency by offloading the computing tasks from the edge or remote cloud. In this paper, we study the computation migration problem in VCC, where a vehicle transfers unfinished computing missions to other vehicles before leaving a network edge to avoid mission failures. Specifically, we consider a computing mission offloaded from edge cloud to the vehicular cloud. The mission has a linear logical topology, i.e., consisting of tasks which should be executed sequentially. The migration problem is formulated as a sequential decision making problem aiming to minimize the overall response time. Considering the vehicular mobility, communication time, and heterogeneous vehicular computing capabilities, the problem is difficult to model and solve. We thus propose a novel on-policy reinforcement learning based computation migration scheme, which learns on-the-fly the optimal policy of the dynamic environment. Numerical results demonstrate that the proposed scheme can adapt to the uncertain and changing environment, and guarantee low computing latency.**

***Index Terms*—vehicular cloud, computation migration, vehicular mobility, decision making, and reinforcement learning.**

I. Introduction

Technological evolutions of hand-held devices have motivated the development of computation-intensive mobile applications, e.g., virtual reality (VR), augmented reality (AR), and online gaming [1], [2]. Running these applications locally can quickly exhaust the battery or computing resources of mobile devices. To address this issue, computation-intensive applications have been suggested to be offloaded to the remote centralize cloud (CC), and take advantage of the abundant resources. However, the remote execution could cause high latency and increase the backhaul bandwidth consumption. By moving the storage and computing resources closer to the mobile users, mobile edge computing (MEC) [3] is promising to reduce the latency and relieve the backhaul pressure. Yet, the computation and storage capacity of edge cloud is rather limited compared with CC. Therefore, the combination of MEC and CC is considered as a promising emerging computing architecture.

With the development of intelligent transportation system, smart city, and connected and autonomous vehicles, vehicles are granted with more communication and computing resources [4]. For example, the NVIDIA DRIVE™ PX platform integrates the computing capabilities of deep learning, surround vision, and sensor fusion for autonomous driving and can support up to 320 trillion deep learning operations in one second [5]. Exploiting the vehicular networks (VANETs) [6], [7], vehicles with strong computing capabilities can compose a novel cloud, which is termed as the vehicular cloud [8]. Through vehicular cloud computing (VCC), the computing capabilities of edge cloud can be enhanced by offloading computing tasks to the vehicular cloud.

One of the fundamental problems in MEC is the computation migration over edge cloud servers taking into account the mobility of the offloading users. The computation migration problem stems from the tradeoff between the cost of migration, the network overhead, and the latency for users. Furthermore, the distributed deployment of MEC servers and the uncertainty in user mobility pose challenges to the optimal migration decision making in MEC. The computation migration in MEC has been widely studied in the literature [9]–[11]. Sun *et al.* in [9] proposed a computation migration framework, where the MEC servers can either execute the offloaded computing tasks locally or migrate them to the CC. The migration problem is then formulated as an optimization problem aiming to minimize the mobile energy consumption and computation latency. Wang *et al.* in [10] formulated the computation migration problem as an MDP problem based on the random-walk mobility model. Then in [11], they further integrated the workload scheduling in MEC with the computation migration to minimize the reconfiguration and transmission costs by using the Lyapunov optimization techniques.

Different from MEC, the computation migration in VCC is caused by the mobility of vehicles, which are the service providers rather than the users. For example, when a vehicle leaves the vehicular cloud, the computing tasks being processed on that vehicle should be migrated to other vehicles in the cloud to ensure the continuity of the computing services. Therefore, the uncertainty in the vehicular mobility could pose severe challenges to the optimal computation migration decision making in the vehicular cloud. In this paper, we investigate the migration problem in a vehicular cloud assisted computation offloading scenario, where the prior knowledge of vehicular mobility is not available. Specifically, when a computing task is completed on a vehicle in the cloud, a decision needs to be made, i.e., whether the next task should be

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

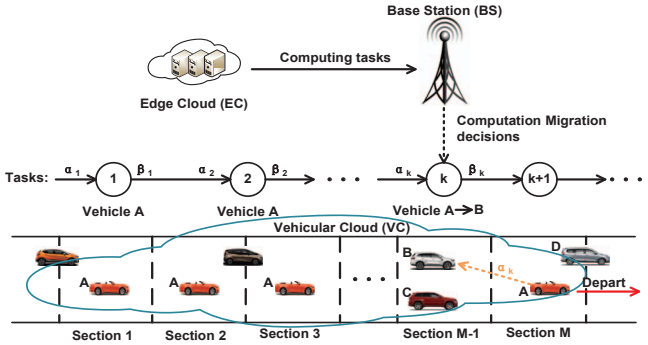

Fig. 1. Computation migration in the vehicular cloud.

migrated from the current vehicle to guarantee the continuity of the computing services. If the task is decided to be migrated, we then need to allocate a vehicle for migration. Due to the lack of the prior knowledge of the vehicular mobility, the migration decision is made according to the current locations of the vehicles in the cloud, the vehicular computing capabilities, and the computation workload of the task that needs to be migrated. Therefore, the computation migration problem is formulated as a sequential decision problem, where the future system state information (i.e., vehicular location information) is unavailable. An on-policy reinforcement learning (RL) based migration scheme is then developed to solve this problem.

Our main contributions are summarized as follows.

1) We investigate the computation migration problem in the vehicular cloud, which is formulated as a sequential decision making problem, considering the uncertainty in the vehicular mobility and the heterogeneity of the vehicular computing capabilities.
2) We develop an on-policy reinforcement learning based computation migration scheme to solve the stated sequential decision making problem. The objective of the proposed scheme is to minimize the overall response time of the offloaded computing tasks.
3) Numerical results demonstrate that the proposed scheme can adapt to the uncertain and changing environment, while guaranteeing low latency.

The remainder of the paper is organized as follows. Section II presents the system model and problem formulation. The RL based computation migration scheme is proposed in Section III. Simulation results are given in Section IV. Finally, Section V concludes this paper and suggests our future works.

II. System Model and Problem Formulation

A. System Overview

We consider the vehicular cloud assisted computation offloading scenario on a unidirectional highway, where N vehicles $\mathcal{V} = \{v_1, \ldots, v_N\}$ within the coverage of a base station (BS) are organized together to form a vehicular cloud. The highway within the coverage of the BS is divided into M sections, where the location of vehicle v_n is defined as $p_n \in \{0, 1, 2, ..., M, M+1\}$. If $p_n = 0$, it means that v_n hasn't reached the cloud yet. If $p_n = M+1$, it means that v_n has left the vehicular cloud. In order to enhance the overall computing efficiency, the edge cloud may offload part of its computation tasks $\mathcal{B} = \{b_1, ..., b_K\}$ to the nearby vehicular cloud. The computing tasks from edge cloud are first sent to the BS, which is aware of the computing capabilities of the vehicles in the vehicular cloud. The BS will then assign these tasks to vehicular cloud for processing. In order to ensure the continuity of the computing services in the vehicular cloud, a computation migration decision is made by the BS after the completion of each task. Fig. 1 shows such a highway scenario, where vehicles within the coverage of the BS are organized together as a vehicular cloud. According to the vehicular computing capabilities, a set of tasks with a linear logical topology is then assigned to some of these vehicles for processing. As shown in this figure, the first task b_1 is assigned to vehicle A for processing. After the completion of task b_1, the second task b_2 is still assigned to vehicle A. However, after the completion of task b_{k-1}, the remaining time for vehicle A in the vehicular cloud may not be sufficient to accomplish task b_k, due to the mobility of vehicle A. According to the current location and computing capability of each vehicle in the cloud, task b_k is decided to be migrated from vehicle A to vehicle B in section $M-1$, in order to ensure the continuity of the offloaded tasks. Therefore, vehicle A transmits the output data β_{k-1} (i.e., α_k) of task b_{k-1} to vehicle B for the processing of task b_k.

B. Vehicular Mobility Model

We consider the computation migration scenario on a unidirectional highway. The distance of the highway within the coverage of the BS is denoted by d. We assume the arrivals of vehicles into the vehicular cloud following the Poisson distribution with the arrival rate of λ_v [8]. The velocity of each vehicle is assumed to be bounded by $[\overline{v} - \varepsilon,\ \overline{v} + \varepsilon]$, where $\overline{v}$ and ε represent the average velocity and the velocity bound on the highway, respectively. We then define the contact interval of each vehicle in the cloud as the time interval, during which the vehicle is within the coverage of the BS, and it is denoted by $[\delta_n, \mu_n]$, where δ_n and μ_n represent the arrival and departure time of v_n, respectively. The departure time of vehicle v_n can then be given by

$$\mu_n = \delta_n + d/s_n \tag{1}$$

where $s_n \in [\overline{v} - \varepsilon,\ \overline{v} + \varepsilon]$ represents the velocity of vehicle v_n. Since the prior knowledge of the vehicular mobility is unavailable, the BS is not aware of the vehicular contact intervals when making the computation migration decisions. The only basis for the migration decision making is the current location p_n of each vehicle $v_n \in \mathcal{V}$.

C. Task Model

In order to characterize the continuity of the computing services in the vehicular cloud, we assume that the tasks offloaded from the edge cloud are with linear logical topology, and each task $b_k \in \mathcal{B}$ can be processed on any of the N

Fig. 2. Computing tasks with linear logical topology.

vehicles in the vehicular cloud. To describe the parametric context of each task, we define a ternary tuple [12] as $\phi_k = (\omega_k, \alpha_k, \beta_k)$, where ω_k, α_k, and β_k represent the computation workload, the size of the input data, and the size of the output data of task b_k, respectively. Fig. 2 shows the linear logical topology of the offloaded computing tasks, of which the output data of the previous task is the input data for the next task, i.e., $\alpha_k = \beta_{k-1}$. Linear logical topology is a single input structure with strong data dependency between tasks. Therefore, to ensure the continuity of the computing services, tasks should be migrated or completed before the vehicles leave the vehicular cloud.

D. Execution Model

To calculate the overall response time of the offloaded tasks, we need to first model the execution process of every single task. Specifically, the response time of each task consists of three parts, i.e., processing time, communication time, and queuing time.

- **Processing Time:** We consider that vehicles may have different computation capabilities such as different clock frequencies. The processing time of task b_k processed on vehicle v_n is then given by

$$p_n^k = \omega_k / f_n \tag{2}$$

  where ω_k represents the computation workload of task b_k, and f_n denotes the processing capability (i.e., clock frequency) of vehicle v_n. In addition, we define $t_u = \omega_{\min}/f_{\max}$ as the normalize time for the computation migration problem, where $\omega_{\min}$ is the minimum computation workload of all the offloaded tasks, and $f_{\max}$ represents the maximum clock frequency of the vehicles in the vehicular cloud.
- **Communication Time:** The communication time is defined as the transmission time of the input/output data of a certain task between two different vehicles. For example, if task b_k is decided to be migrated from vehicle v_l to v_n, the communication time of b_k can then be given by

$$c_{l,n}^k = \alpha_k / r_{l,n} \tag{3}$$

  where α_k represents the size of the input data of task b_k, and $r_{l,n}$ represents the data transmission rate between vehicle v_l and v_n. More specifically, we assume that the data transmission rate $r_{l,n}$ is negatively correlated with the distance between vehicles v_l and v_n, which is denoted by $d_{l,n} = |p_l - p_n|$. If the distance $d_{l,n}$ exceeds the threshold d^c, the communication between the two vehicles is not achievable.
- **Queuing Time:** When a task is assigned to a vehicle, if the vehicle has not reached the cloud yet, it may not be processed immediately. The amount of time it takes for a task to wait for being processed is defined as the queuing time. Let t_k^m denote the time that the task b_k is decided to be migrated, the queuing time of task b_k migrated to vehicle v_n can then be given by

$$q_n^k = (\delta_n - t_k^m)^+ \tag{4}$$

  where $(\delta_n - t_k^m)^+ = \max\{0, (\delta_n - t_k^m)\}$, and δ_n represents the arrival time of v_n into the vehicular cloud.

E. Problem Formulation

With the analysis in Section II-D, the response time of task b_k migrated from vehicle v_l to v_n can be given by

$$t_{k,n}^r = q_n^k + c_{l,n}^k + p_n^k \tag{5}$$

where l represents the index of the vehicle, in which the previous task b_{k-1} is processed. The migration decision set for task b_k is denoted by $\mathcal{A}_k = \{1, 2, 3, ..., N\}$, where N represents the number of vehicles in the cloud. After the completion of task b_{k-1}, the BS chooses a vehicle with the index of $a_k \in \mathcal{A}_k$ for the migration of the next task b_k aiming at minimizing the response time of all the offloaded tasks. Based on the joint migration decision set $\boldsymbol{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_K$, the global optimization problem in the VCC system can be given by

$$\min_{\boldsymbol{a} \in \boldsymbol{A}} \sum_{k=1}^{K} t_{k,a_k}^r \tag{6}$$

where K is the number of the offloaded tasks, and t_{k,a_k}^r represents the response time of task b_k with the migration decision of a_k. Due to the absence of the prior knowledge of the vehicular mobility, each migration decision $a_k \in \mathcal{A}_k$ can only be made according to the current locations of the vehicles in the cloud. Each migration decision will have an impact on the system state (i.e., vehicular locations) when the next migration decision is going to be made. Therefore, the computation migration problem in the VCC system is formulated as a sequential decision making problem, of which the future system state is not available. Detailed solution to this problem will be introduced in the next section.

III. RL based Computation Migration Scheme

In this section, a RL based computation migration scheme is proposed to solve the sequential decision making problem. Specifically, the migration scheme is based on the on-policy reinforcement learning algorithm (i.e., "Sarsa") [13], where the episodic semi-gradient control is utilized to approximate the optimal Q-value function.

A. On-Policy Reinforcement Learning

As an important branch of machine learning, reinforcement learning is promising in handling tough situations that approach real-world complexity. Different from the supervised learning, statical samples provided by external supervisors are not necessary for reinforcement learning. Agents in reinforcement learning operate according to their own experiences, although they will face significant environmental uncertainties. One of the characteristics of reinforcement learning is trial and

error, i.e., the trade-off between exploration and exploitation. Agents are more likely to take advantage of the existing effective actions, while they also have to explore new actions that may yield higher rewards in the future. Another feature is the delayed reward. Agents not only focus on the immediate rewards but also the cumulative rewards in the long term, which is specified as the value function. Specifically, the cumulative discounted reward of taking an action in a certain state and then following the optimal policy is defined as the Q-value. The proposed computation migration scheme is based on an on-policy reinforcement learning algorithm, where the Q-value is updated through the state-action-reward-state-action (i.e., "Sarsa") experiences. An experience in "Sarsa" is of the form $\langle s, a, r, s', a' \rangle$, indicating that the agent was in state s, took action a, received reward r, and ended in state s', from which it decided to take action a'. Details about the RL based computation migration scheme are presented as follows.

*1) **System State**:* The system state $\boldsymbol{s}_k$ reflects the task b_k that whether needs to be migrated, the vehicle where the previous task b_{k-1} is completed, and the current locations of all the vehicles in the cloud. Therefore, the system state set $\mathcal{S}$ can be given by

$$\mathcal{S} = \{\boldsymbol{s}_k \,|\boldsymbol{s}_k = (b, v, p_1, p_2, ..., p_N)\} \tag{7}$$

where b represents the current task (i.e., b_k) that needs to be migrated, v denotes the vehicle, where task b_{k-1} is completed, and p_n represents the location of vehicle $v_n \in \mathcal{V}$. Note that the movement of each vehicle on the highway can be regarded as a uniform process, the location of vehicle v_n at time t can be given by

$$p_n(t) = \begin{cases} 0, & t < \delta_n \\ m, & \frac{m-1}{M} t_n^d \le t \le \frac{m}{M} t_n^d \\ M+1, & t > \mu_n \end{cases} \tag{8}$$

where $t_n^d = \mu_n - \delta_n$ is the amount of time that vehicle v_n spends within the vehicular cloud, and $m \in \{1, 2, 3, ...M\}$ represents the index of the section on the highway, where the vehicle v_n is within. If t is less than the arrival time δ_n, $p_n(t) = 0$, which means that v_n hasn't reached the cloud at time t. If t is larger than departure time μ_n, $p_n(t) = M + 1$ meaning that v_n has left the cloud at time t.

*2) **Actions**:* After the completion of task b_{k-1}, the BS (i.e., agent) has to decide the vehicle, to which the next task b_k should be migrated. The possible migration decision set $\mathcal{A}_k$ for task b_k can be given as

$$\mathcal{A}_k = \{1, 2, 3, ..., N\} \tag{9}$$

where N is the number of vehicles in the vehicular cloud, and $a_k \in \mathcal{A}_k$ represents the vehicle, which the BS chooses for the migration of task b_k.

*3) **Reward Function**:* The objective of the RL based computation migration scheme is to minimize the overall response time of all the offloaded computing tasks. Therefore, the immediate reward is related to the response time of each task. Given a migration decision a_k, the system reward under the current state $\boldsymbol{s}_k$ can be given by

$$R(\boldsymbol{s}_k, a_k) = \begin{cases} H - t_{k,a_k}^r, & a_k \in \mathbb{R}_{a_k}^f \\ -1, & \text{otherwise} \end{cases} \tag{10}$$

where H is a constant larger than the response time t_{k,a_k}^r, $\forall b_k \in \mathcal{B}$, and $\mathbb{R}_{a_k}^f$ is defined as the feasible region of the migration decision a_k. If the migration decision a_k is feasible, the reward is $H - t_{k,a_k}^r$, otherwise, the reward is set to be -1 as a punishment for the agent. The definition of the feasible region is given by

$$\mathbb{R}_{a_k}^f = \left\{ a_k \;\middle|\; \begin{array}{l} \delta_{a_k} \le t_{k+1}^m \le \mu_{a_k} \;\&\; \delta_{a_k} \le \mu_{a_{k-1}} \\ \;\&\; \left| p_{a_{k-1}}(t_k^c) - p_{a_k}(t_k^c) \right| \le d^c \end{array} \right\} \tag{11}$$

where t_k^m represents the time when the task b_k is decided whether to be migrated, which can be given by

$$t_k^m = t^G + \sum_{h=1}^{k-1} t_{h,a_h}^r \tag{12}$$

where t^G represents the time when the first task is offloaded to the vehicular cloud, and $t_k^c = \max\{t_k^m, \delta_{a_k}\}$ denotes the time when the transmission of b_k starts. If the remaining time of vehicle v_{a_k} in the cloud is not sufficient to complete the task b_k, or the distance between the vehicles $v_{a_{k-1}}$ and v_{a_k} is beyond the communication range d^c, the migration decision a_k cannot be implemented. The minimization of the overall response time is then equivalent to the maximization of the cumulative reward, which can be given by

$$\max_{\boldsymbol{a} \in \boldsymbol{A}} \sum_{k=1}^{K} R(\boldsymbol{s}_k, a_k) \tag{13}$$

B. Q-value Function

Normally, the Q-value function can be implemented simply by a lookup table. However, considering the system state $\mathcal{S}$ of the computation migration problem, the required state space is $KN(M+2)^N$, which is combinatorial and enormous. It is incapable of finding the optimal value function in the finite time and data. Therefore, the function approximation is utilized in the reinforcement learning algorithm. Specifically, the linear approximation with the polynomial features is applied to calculate the Q-value function. Furthermore, to avoid the linear inseparability caused by the linear approach, a N-dimensional Q-value function is designed for the action set $\mathcal{A}_k$. The approximate Q-value function for action a_k can be given by

$$\hat{Q}_{a_k}(\boldsymbol{s}_k, a_k; \boldsymbol{w}_{a_k}) = \boldsymbol{w}_{a_k}^T \boldsymbol{x}_{a_k}(\boldsymbol{s}_k, a_k) \tag{14}$$

where $\mathbf{w}_{a_k}$ is the weight vector corresponding to the system state $\boldsymbol{s}_k$ and action a_k. Each polynomial basis function $x_i(\boldsymbol{s}_k, a_k) \in \boldsymbol{x}_{a_k}(\boldsymbol{s}_k, a_k)$ can be given by

$$x_i(\boldsymbol{s}_k, a_k) = \prod_{j=1}^{N+3} s_j^{o_{i,j}} \tag{15}$$

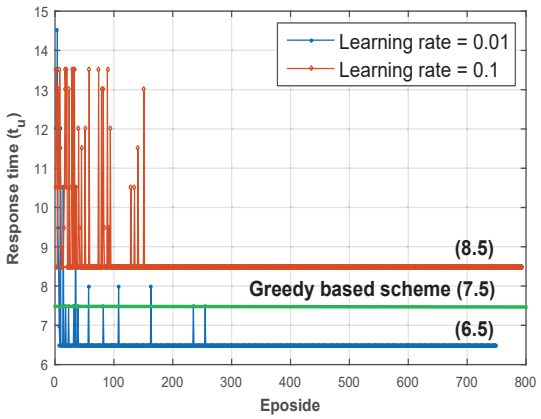

Fig. 3. Overall response time of the RL and Greedy based schemes.

where $(s_1, s_2, ..., s_N) = (p_1, p_2, ..., p_N)$, $s_{N+1} = b$, $s_{N+2} = v$, and $s_{N+3} = a_k$ in response to the system state $\boldsymbol{s}_k$ and action a_k, and $o_{i,j}$ is an integer in the set $\{0, 1, ..., O\}$. These functions make up the order-O polynomial basis, which contains $(O+1)^{N+3}$ different functions. Considering that the number of the feature functions grows exponentially with the state space dimension, we only select a subset of them for the approximation of the Q-value function. Furthermore, the episodic semi-gradient control is utilized to calculate the optimal Q-value function. Detailed training algorithm is shown in Algorithm 1, where

$$\nabla \hat{Q}_{a_k}\left(\boldsymbol{s}_k, a_k; \boldsymbol{w}_{a_k}\right) = \boldsymbol{x}_{a_k}\left(\boldsymbol{s}_k, a_k\right) \tag{16}$$

The $\varepsilon - greedy$ policy is utilized to balance the exploration and exploitation, i.e., to balance the maximization of rewards based on known knowledge and new actions that attempt to acquire unknown knowledge.

IV. Simulation Results

In this section, we conduct extensive simulations to evaluate the performance of the proposed RL based computation migration scheme in the VCC system. Specifically, we compare the proposed scheme with the Greedy based scheme, which focuses on the minimization of the response time of every single task. The main parameters used in our analysis are provided in Table I. The normalized time is set as $t_u = \omega/f$. The highway within the coverage of the BS is divided into $M = 4$ sections to represent the locations of the vehicles in the vehicular cloud. The communication threshold is set as $d^c = 1$, which means that vehicles can only communicate with the vehicles within their own sections or adjacent sections.

A. Overall response time

In order to evaluate the performance of the proposed computation migration scheme, we randomly select a scenario from the simulation, where 10 vehicles with constant motions are included. Fig. 3 shows the overall response time of the 5 offloaded computing tasks achieved by the RL and Greedy based schemes. We can see that the response time achieved by the Greedy based scheme is $7.5t_u$, while the results acquired from the RL based schemes with the learning rate of 0.1 and 0.01 are $8.5t_u$ and $6.5t_u$, respectively. It can also be seen that the RL based scheme with the learning rate of 0.1 converges faster than that with the learning rate of 0.01. The RL based scheme with a larger learning rate has a faster speed of convergence, while it may cross the optimal solution.

Algorithm 1: On-policy Reinforcement Learning based Computation Migration for VCC

Initialization:
Q-value functions $\hat{Q}_{a_k}\left(\boldsymbol{s}_k, a_k; \boldsymbol{w}_{a_k}\right), \forall a_k \in \mathcal{A}_k$;
learning rate α; weight decay γ; exploration probability ε
for *each episode* **do**
  Receive the initial state $\boldsymbol{s}_1$
  Choose a random probability p (ε – greedy)
  if $p < \varepsilon$ **then**
    randomly select an action a_1
  else
    $a_1 = \arg\max_{a_1} \hat{Q}_{a_1}\left(\boldsymbol{s}_1, a_1; \boldsymbol{w}_{a_1}\right)$
  for $i = 2 : K$ **do**
    Take action a_{i-1}, observe state $\boldsymbol{s}_i$
    $R_{i-1} = R\left(\boldsymbol{s}_{i-1}, a_{i-1}\right)$
    if $R_{i-1} == -1$ **then**
      break to the next episode;
    if $i == K$ **then**
      $\boldsymbol{w}_{a_{i-1}} = \boldsymbol{w}_{a_{i-1}} + \alpha\left[R_{i-1} - \gamma \hat{Q}_{a_{i-1}}\right] \nabla \hat{Q}_{a_{i-1}}$
    else
      Choose action a_i (ε – greedy)
      $\boldsymbol{w}_{a_{i-1}} =$
      $\boldsymbol{w}_{a_{i-1}} + \alpha\left[R_{i-1} + \gamma \hat{Q}_{a_i} - \hat{Q}_{a_{i-1}}\right] \nabla \hat{Q}_{a_{i-1}}$

TABLE I
Simulation Parameters

Parameters	Value
Number of vehicles and tasks	$N = 10 \sim 15, K = 5 \sim 10$
Computation workload of each task	$\omega_k = 2\omega \sim 6\omega$
Computing capabilities of each vehicle	$f_n = f \sim 2f$
Size of input/output data	$\alpha_k, \beta_k = \omega \sim 6\omega$
Number of highway sections	$M = 4$
Communication threshold	$d^c = 1$
Data transmission rate between v_l and v_n	$r_{l,n} = f \sim 2f$

B. Response time of each task

Fig. 4 shows the arrival time and departure time of each vehicle in the vehicular cloud. The velocity of each vehicle is bounded by $[\overline{v} - \varepsilon,\ \overline{v} + \varepsilon]$, and thus the amount of time each vehicle spends in the cloud is bounded by $\left[d(\overline{v} + \varepsilon)^{-1},\ d(\overline{v} - \varepsilon)^{-1}\right]$, where d is the distance of the highway within the coverage of the BS. Due to the lack of the prior knowledge of the vehicular mobility, the BS is not aware of the contact interval of each vehicle in the cloud. Therefore,

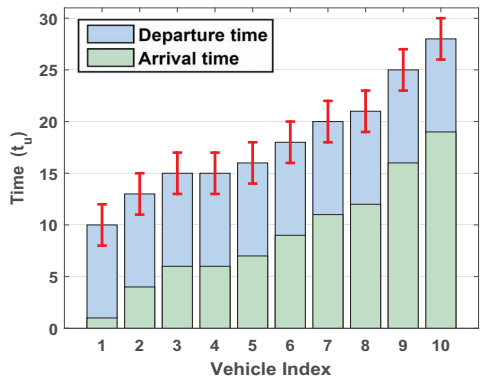

Fig. 4. Variety of the vehicular mobility in the vehicular cloud.

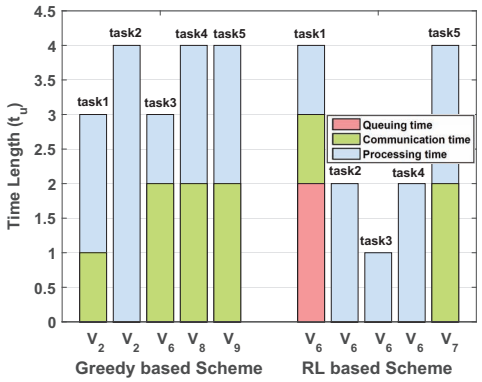

Fig. 5. Response time of each offloaded computing task.

in the RL based migration scheme, the vehicular contact intervals (i.e., vehicular mobility) change randomly for each training episode. According to the Q-value function acquired from the training of the changing vehicular movement, it is capable to make the optimal migration decision based on a given system state (i.e., vehicular locations).

Fig. 5 shows the queuing, communication, and processing time of five offloaded tasks achieved by two migration schemes, based on a randomly selected vehicle motion scenario. The first task b_1 is offloaded from the edge cloud to the BS at the time of $7t_u$. For the Greedy based scheme, b_1 is assigned to vehicle v_2 at the time of $7t_u$. The communication time and computing time of b_1 are t_u and $2t_u$, respectively. After the completion of b_1, the second task is still assigned to v_2, and thus the communication time of b_2 is 0. When v_2 leaves the vehicular cloud, task b_3 is decided to be migrated from v_2 to v_6 within the adjacent section, which leads to the communication time of $2t_u$. The next task b_4 is migrated from v_6 to v_8, and the last task is migrated from v_8 to v_9. The overall response time achieved by the Greedy based scheme is $18t_u$. For the RL based migration scheme, according to the Q-value function, task b_1 is not offloaded to the vehicular cloud at the time of $7t_u$, however, it is offloaded to v_6 at the time of $9t_u$ when v_6 reaches the cloud leading to the queuing time of $2t_u$. Tasks b_1 to b_4 are then processed in v_6, which omits the process of data transmission between vehicles. The last task b_5 is migrated to v_7 when v_6 leaves the cloud. The overall response time of the RL based scheme is $13t_u$, which is superior to that of the Greedy based scheme.

V. Conclusion

In this paper, we have proposed a reinforcement learning based computation migration scheme for the vehicular cloud computing, which aims at minimizing the overall response time of all the offloaded computing tasks. The computation migration problem has been formulated as a sequential decision making problem, where the future system state is unavailable. An on-policy reinforcement learning algorithm has been utilized to solve the problem. Extensive simulations have been conducted to demonstrate the efficiency of the proposed migration scheme. For future works, we will consider multiple types of services in the vehicular cloud, and more general scenarios where different mobility models may apply.

Acknowledgement

Lin Gui is the corresponding author. This work was financially supported by the National Natural Science Foundation of China under Grant 61420106008 and Grant 61471236, the 111 Project (B07022), National Key Laboratory of Science and Technology on Communications (KX172600030), the Shanghai Key Laboratory of Digital Media Processing and Transmissions, and Natural Sciences and Engineering Research Council (NSERC), Canada.

References

[1] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable iot architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Aug. 2017.

[2] N. Zhang, P. Yang, S. Zhang, D. Chen, W. Zhuang, B. Liang, and X. Shen, "Software defined networking enabled wireless network virtualization: Challenges and solutions," *IEEE Netw.*, vol. 31, no. 5, pp. 42–49, May 2017.

[3] Y. Wu, K. Guo, J. Huang, and X. Shen, "Secrecy-based energy-efficient data offloading via dual connectivity over unlicensed spectrums," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3252–3270, Dec. 2016.

[4] S. Zhang, J. Chen, F. Lyu, N. Cheng, W. Shi, and X. Shen, "Vehicular communication networks in automated driving era," *arXiv preprint arXiv:1805.09583*, 2018.

[5] Nvidia, "Nvidia DRIVE PX," [Online].Available:http://www.nvidia.ca/object/drive-px.html.

[6] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. Shen, "Big data driven vehicular networks," *IEEE Netw.*, to appear.

[7] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing crowd collaborations for software defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 80–86, Aug. 2017.

[8] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An SMDP-based resource allocation in vehicular cloud computing systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7920–7928, Dec. 2015.

[9] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[10] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. IEEE MILCOM*, Baltimore, USA, Oct. 2014, pp. 835–840.

[11] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[12] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2011.