Knowledge-Driven Service Offloading Decision for Vehicular Edge Computing: A Deep Reinforcement Learning Approach

Qi Qi, *Member, IEEE*, Jingyu Wang, *Member, IEEE*, Zhanyu Ma, *Senior Member, IEEE*, Haifeng Sun, Yufei Cao, Lingxin Zhang, and Jianxin Liao, *Member, IEEE*

***Abstract*—The smart vehicles construct Internet of Vehicle (IoV), which can execute various intelligent services. Although the computation capability of a vehicle is limited, multi-type of edge computing nodes provide heterogeneous resources for intelligent vehicular services. When offloading the complex service to the vehicular edge computing node, the decision for its destination should be considered according to numerous factors. This paper mostly formulate the offloading decision as a resource scheduling problem with single or multiple objective function and constraints, where some customized heuristics algorithms are explored. However, offloading multiple data dependence tasks in a complex service is a difficult decision, as an optimal solution must understand the resource requirement, the access network, the user mobility, and importantly the data dependence. Inspired by recent advances in machine learning, we propose a knowledge driven (KD) service offloading decision framework for IoV, which provides the optimal policy directly from the environment. We formulate the offloading decision for the multiple tasks as a long-term planning problem, and explore the recent deep reinforcement learning to obtain the optimal solution. It can scruple the future data dependence of the following tasks when making decision for a current task from the learned offloading knowledge. Moreover, the framework supports the pre-training at the powerful edge computing node and continually online learning when the vehicular service is executed, so that it can adapt the environment changes and can learn policy that are sensible in foresight. The simulation results show that KD service offloading decision converges quickly, adapts to different conditions, and outperforms a greedy offloading decision algorithm.**

***Index Terms*—Internet of Vehicle, service offloading decision, multi-task, knowledge driven, deep reinforcement learning.**

Manuscript received August 20, 2018; revised December 11, 2018 and January 14, 2019; accepted January 17, 2019. Date of publication January 21, 2019; date of current version May 28, 2019. This work was supported in part by the National Natural Science Foundation of China under Grants 61471063, 61671079, 61771068, and 61773071, and in part by the Beijing Municipal Natural Science Foundation under Grant 4182041. The review of this paper was coordinated by the Guest Editors of the Special Section on Machine Learning-Based Internet of Vehicles. *(Corresponding author: Zhanyu Ma.)*

Q. Qi, J. Wang, H. Sun, L. Zhang, and J. Liao are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: qiqi8266@bupt.edu.cn; wangjingyu@bupt.edu.cn; hfsun@bupt.edu.cn; zhanglingxin@ebupt.com; liaojx@bupt.edu.cn).

Z. Ma is with the Pattern Recognition and Intelligent System Laboratory, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: mazhanyu@bupt.edu.cn).

Y. Cao is with the EBUPT Information Technology Company, Ltd., Beijing 100191, China (e-mail: caoyufei@ebupt.com).

Digital Object Identifier 10.1109/TVT.2019.2894437

I. Introduction

ALONG with the development of information technology, the vehicles with advanced computation and communication capabilities spring up. With the promising electronic control units (ECUs), the vehicles construct as Internet of Vehicle (IoV) which can execute multimedia entertainment services, navigation, remote vehicle diagnosis and even Artificial Intelligence (AI) based automatic driven services. Moreover, unmanned aerial vehicle (UAV) can be used as patrol robot which travels in the area of industry field. These applications bring new experience for the vehicular users. However, the limited capability of the ECU and the instability of the vehicular network still impacts the quality of the service. Fortunately, edge computing is combined to traditional vehicular network to strengthen its service capability. The edge computing environment providing for vehicles includes computation nodes located with Base Station (BS) and many kinds of computational hot points of WLAN (Wireless Local Area Networks) access technology, such as intelligent roadside unit, cloudlet in buildings and so on. All of these computational nodes and the neighboring vehicles can provide service execution for the traveling vehicles. The current vehicular edge computing environment is depicted in Fig. 1, which supports the efficient communication, control, and computation requirements of intelligent services for vehicles [1].

1) The edge computing nodes located with multiple BSs always consist of high-performance servers or distributed data centers. The BS edge computing nodes are distributed in different geographic regions, and may have different serving areas, rental cost and available resources, so as to satisfy the service requirement in the moving vehicles. The vehicles act like thin clients connecting over to the edge computing node through mobile network without the need of accessing to the remote cloud.
2) The cloudlet edge computing servers are deployed with wireless APs (Access Points) located at intelligent roadside units, such as the road lamp, intersection, shops or buildings [2], [3]. These nodes provide various local capability and services for the vehicles within the coverage area, including computation, networking, storage and applications by limited transmission coverage communication technology. This type of nodes in the hierarchy edge computing environment is important, as it is closer

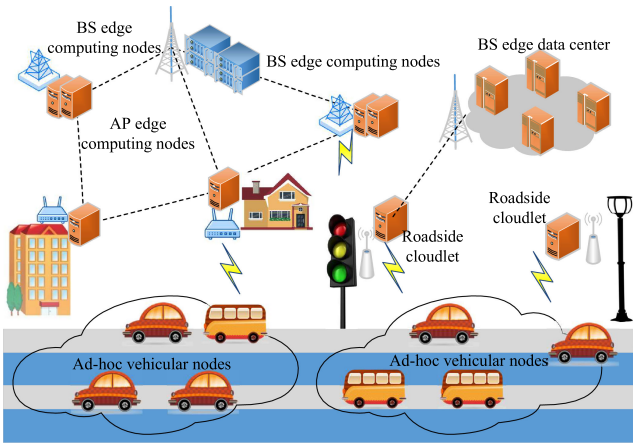


Fig. 1. The vehicular edge computing environment.

to users, has powerful computation potential, and is well-connected via a high-speed local communication technology, which is of benefit to provide low-latency computation and rich computational resources.

3) The ad hoc vehicular nodes can be also treated as edge computing nodes since the user can exploit the capabilities of the neighboring vehicles to execute some services [4]. The computation for services are offered by neighboring vehicles that have sufficient resources to act as edge computing servers. Consumers need to discover the vehicle edge nodes, be aware of their resources, communicate, and request resources from them. In this scenarios, the resource provider and resource consumer can be transferred according to the service requirement.

Offloading the services to the edge computing nodes other than hosing the execution on the vehicular device itself, can expand the limited capabilities of the smart vehicles. The above heterogeneous vehicular edge computing nodes have different capability and scenarios, which provides multiple choices for the vehicular users. The BS edge computing nodes have large scale coverage and high-performance computation. If an application in a fast traveling vehicle requires high computational power, it is more desirable to offload to the BS edge computing node. However, a weakness of the BS edge computing nodes is that the users may experience long latency for data exchange with the corresponding far away nodes through the 3G/4G access network. Long latency may hurt the interactive response, since humans are acutely sensitive to delay and jitter. For the AP edge computing nodes, if too many vehicles access the same wireless network simultaneously, the scarce bandwidth may affect the application QoS. For the real-time applications, it would be more beneficial for the user to execute the service locally on the ad hoc vehicle node to avoid the long communication time. Since the current vehicular service are always complex which contain multiple tasks with various requirements, we should make offloading decision for each task overall considering communication latency, computation capability and vehicular mobility.

Traditionally, the offloading decision algorithms depend on the human experience. The relevant knowledge is abstracted by researchers, including the system formulation, algorithm solving and method optimization. However, abstracting knowledge by hand has mainly three limitations. Firstly, the vehicular environment which contains vehicles, edge computing nodes, and access network, cannot has a comprehensive precise, and real-time evaluation. For example, for the vehicular service offloaded to the heterogenous nodes, the task execution progress is always abstracted as Non-Poisson queuing model, which is hard to solve and guarantee the fairness [5]. Moreover, the objective for the offloading decision model is almost non-convex function, and Lagrange Relaxation is need to process the constraint conditions or some specific heuristic algorithms are used to find the near-optimal solutions. Finally, each time the vehicular edge computing environment changing, the offloading decision should be re-computed, which results in more service delay and higher cost. Therefore, it is necessary to find an intelligent method that can learn knowledge as human, and provide foresight offloading decision. Fortunately, the advantage of edge computing nodes is that they are deployed with the network devices, and can capture multi-dimensionality data from the environment, including vehicular behavior, task information and network status. It is feasible that utilizing the data from environment to keep continually learning the decision relevant knowledge and providing offloading policy.

Deep learning is the hot area for the Computer Vision, Speech Recognition, Natural Language Processing, and even the computer networks, with some remarkable achievements. It includes DNN (Deep Neural Network), CNN (Convolutional Neural Network), RNN (Recurrent Neural Networks), ResNet, DenseNet and so on [6]. These various types of neural networks utilize the multi-layer network structure and nonlinear transformation, construct the lower-level features, and formulate the abstract and distinguish high-level expression, so that to percept and express of the things.

The service offloading decision in vehicle edge computing environment is affected by many factors, and it can be abstracted to a non-convex optimization problem with complicated objective function and constraints. The deep learning model divides the complex mapping problem into several embedded simple mappings depicted by the multiple layers of the model. The iteration optimization based on gradient during the process of training minimizes the loss function which expresses the approximate degree. After the supervised learning process based on the human labeled history data or the solution from heuristic algorithms, the deep learning model can obtain the approximate optimization solution of the complex mapping problem. When the model is deployed in the real-word setting after the training, the approximate optimization solution can be independently obtained according to the environment data in real-time.

Considering the environment changing in the future, the deep learning method needs the new labeled data. The service offloading decision for complex service with tasks of different requirement should have the capabilities of long-term programming and continually learning. Deep reinforcement learning (DRL) method combines the perceive capability of the deep learning and the decision capability of the reinforcement learning.

Reinforcement learning is based on the MDP (Markov Decision Process) theory, but it need not to formulate the states transfer probabilities. It decides according to the current situation, depending on the samples of the system states and objective reward from the experience policy. Moreover, the effective perceive of deep learning models improves the poor performance for traditional RL methods when the environment contains the high dimension input state space or large action sets.

In this paper, we address the shortcomings of traditional service offloading via a knowledge driven (KD) service offloading decision framework. It includes the decision model with the DRL algorithm to learn the offloading decision knowledge, and the observation function that is responsible for obtaining the data of vehicular mobility and the edge computing nodes. The KD service offloading decision framework provides a unique platform for various vehicular services which can be offloaded to the three types of edge computing nodes as they need them. It aims at achieving long-term optimal performance experienced by the vehicular users. The main contributions of this paper are described as follows:

1) Making the service offloading decision according to the learned long-term optimal decision knowledge for IoV. We propose a DRL based offloading decision model, which understands the resource requirement, the access network, and the user mobility. Importantly, it considers the future data dependency of the following tasks when making decision for a current task from the learned offloading knowledge. By this model, the optimal policy can be obtained directly from the environment without the complicated computation of the offloading solution.
2) The mobility model of the vehicular edge computing environment is formulated for service offloading decision. Since the vehicle moving impacts the offloading destination selection and it is hard to modeling, we formulate the impact of task delay by mobility according to the accessed vehicular edge computing node. This model can be directly used in the online service offloading learning.
3) Exploring the A3C algorithm to realize the online optimization offloading decision for the moving vehicles. The offloading decision model is trained for each type of the complex services at the edge computing node, and then distributed to the vehicles. The vehicles perform asynchronous online learning when they are running the service, and update the new model to the edge computing node. By the feedback reward in each running time, the KD service offloading decision framework can adapt the environment changing.

In Section II, we first describe the related work about service offloading decision for vehicular edge computing and review import development of joint optimization based on Deep learning and DRL. We then present the architecture of the KD service offloading decision framework and the problem formulation in Section III. The DRL model and corresponding service offloading algorithm are presented with details in Section IV. Performance evaluation are presented in Section V. Section VI concludes the paper and presents the future work.

II. Related Work

A. Offloading Decision for Vehicular Edge Computing

The vehicular edge computing is proposed along with the development of edge computing and VANET technology [1]. The high performance of edge computing servers provides vehicles more capability for executing complicated application [7]. Qin *et al.* [6] propose VehiCloud as a service-oriented cloud architecture providing routing service by predicting vehicles future locations. The three edge computing nodes support vehicles offloads their services, such as image detection, speech recognition, web fasten and online game. Generally, complicated service can be divided into multiple tasks which can be executed independently at local ECU, or edge computing nodes. Some part of the tasks contains the dependency by data transmission, which construct as a DAG (Directed Acyclic Graph). Therefore, the offload decision should consider data transmission between the dependent tasks.

From the mobile cloud computing, there are many valuable works focusing on offloading decision, which includes offload or not, offload volume and offload location, considering service type, user perfect, access technology, network traffic, device capability, edge node property and so on. The offloading decision is extreme complicated, consisting of "single-user to single-node", "multi-user to single-node" and "multi-nodes" scenarios. The "single-user to single-node" problem only has one target node and the decision should make a joint optimization for all the tasks in a DAG [8]. If the service arrives in stochastic model, the long-term reward and the cache stability should be considered. The "multi-user to single-node" problem focus on the communication interference and resource competition among the users [9], [10]. The service offloading policy should consider the resource allocation in the edge computing nodes and the utilization of the point-to-point cooperation among the users. Finally, the "multi-node" problems for single-user and multi-user all pay attention on the selection and cooperation of the edge computing nodes [11]–[14]. They analyze the relationship of service offloading volume and the resource of edge computing nodes, and try to avoid to select the hot node and hot line which will result in overload.

The above three problems can be formulated by combinatorial optimization, MDP, Semi- Markov Decision Process, Cooperative Game, or Non-cooperative Game models [11], [14], with specific objective function and constraints. Since most of these models are NP-hard problems, multi-stage heuristic algorithm, Lagrangian Relaxation Approach, Lyapunov algorithm [10], Particle Swarm algorithm, Genetic Algorithm, and Simulated Annealing are used to solve the problems in the acceptable time.

B. Combinatorial Optimization Based on Deep Learning

The deep learning model solves the combinatorial optimization problem by several embedded simple mappings depicted by the multiple layers. After the supervised learning process, it learns the approximate optimization solution by nonlinear

fitting. When the model is deployed, the approximate optimization solution can be obtained according to the input data directly.

Some works take use of the deep learning model instead of the traditional formulation and heuristic methods, to find the optimized solution from the high dimensionality data in network and cloud resource management. The Depth Boltzmann Machine is used to the network traffic control and routing [15], [16]. The input traffic matrix is constructed by the number of arriving packets to the router in a period of time. The output of the model is the next-hop router. The labeled data is obtained by the OSPF (Open Shortest Path First) algorithm in each router for training next-hop policy. In additional, the DNN model is used to find the near-optimal solutions of caching placement, user association, and content delivering [17]. This work reduces complexity in the delay-sensitive operation since the computational burden is shifted to the DNN training phase. The deep learning-based auction mechanism is used for optimization of edge computing resources [18]. The goal is obtaining the max reward of edge computing resource provider by reasonable resource allocation and payment rule, under the condition of incentive compatibility and individual rationality.

C. Combinatorial Optimization Based on Deep Reinforcement Learning

Deep learning method has better generalization ability, without the need to formulate the environment. After the training the deep learning method, it can achieve approximate optimal solution of the optimization results in real time. However, the deep leaning method is trained by labeled data from the human labeling or heuristic algorithms, which can hardly deal with possible changes in the future.

Deep Reinforcement Learning has achieved remarkable success in a range of tasks, from continuous control problems in robotics to playing games like Go and Atari. The DQN (Deep Q-Network) model firstly proposed by DeepMind combines CNN and Q-learning, trains the CNN by the rewards obtained by the Q-learning in each epoch and realizes the directly control from input states to the output policy [19]. Moreover, deep policy gradient expresses the policy by the parameters of the deep neural network, i.e. the probability for choosing each action in each decision epoch. Through the finding of the gradient for the policy, this method can find the approximate optimal policy. A3C (Asynchronous Advantage Actor-Critic) model utilizes two independent deep neural networks, which are taking charge of update policy and providing policy gradient, respectively [20]. The asynchronous gradient descent method is used for optimizing the parameters of the deep neural network.

Taking use of the DRL method to optimizing the resources is different with the playing game, including the input data, action space and the long-term reward related to the optimal objective. The DQN with CNN and Q-learning is used in the cache-enabled interference alignment wireless network [21], where the system state is channel state and cache stage. The optimal policy is selecting active users for realizing minimization of the throughput. Mao et al. [22] find the optimal resource management policy by DRL. The resource requirement for all the job arriving in a period is abstracted as an image which is treated as the system state and inputted into the CNN. The resource scheduling decision is made by policy gradient method. Liu *et al.* [23] utilize the Long Short-Term Memory (LSTM) and DNN combined with Q-learning to dynamic optimize energy and resource for VM, respectively. The system state consists job arriving time, resource utilization rate, job status and resource requirement; the action set is the candidate virtual machines for each job. Currently, the A3C algorithm obtain outstanding results for discrete or continuous controlling problems, which is successfully used in video code selection [24]. From the observation of QoE, network bandwidth, residual cache and the supported codes, A3C algorithm provides a video code selection, which can realize a long-term optimization of user experience. Moreover, A3C is used in user scheduling and resource allocation in heterogeneous networks [25], and A3C with the improved training is used in the traffic allocation among multiple paths [5].

From the above work, DRL has two advantages, i.e. adaptively and long-term planning. Comparing with the heuristic algorithms, the DRL can adjust the policy according to the environment changing and find the temporarily suboptimal decision for the long-term optimal solution.

III. System Architecture and Problem Formulation

The KD service offloading decision framework for vehicular edge computing, provides a unique platform for various services and all the controlled vehicles. It includes the decision model with the DRL algorithm to learn the service offloading knowledge, and the observation function that is responsible for obtaining the environment data of vehicular mobility and the edge computing nodes. Since the services have various numbers of tasks, the KD service offloading decision framework keeps one basic decision model for each service. The basic decision model is trained at the powerful edge computing nodes, such as BS edge computing nodes, and then distributed to the vehicles for the real-world service offloading decision. After this training, the system learns the long-term optimal decision knowledge for data dependency tasks in a complex service, from the experience of the history offloading rewards. As the DRL model contains a reward for each decision, the model can be trained online continually when the running services. During this procedure, the parameters are transmitted from the vehicles to the BS edge computing node for updating the basic model periodically.

The number of accessible edge computing nodes for the vehicles are different, and it may change when the vehicle moving. The observation function sorts the accessible nodes for the three types of the edge computing nodes according to their computation power, and adopts the fixed number of each type of the nodes as the candidate offloading destination.

The heterogeneous resources provided by vehicular edge computing nodes are abstracted to several containers with specific functions and parameters. The multi-task in a complicated application is modeled as a specific dataflow graph and denoted by DAG, depicted in Fig. 2. Due to the differences existing in the edge computing nodes, including several cloudlets, BS that

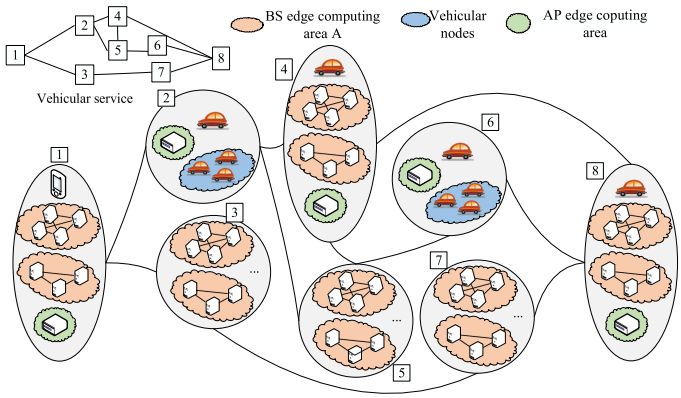


Fig. 2. The vehicular service denoted by a DAG.

contains computing and storage capability and the neighboring vehicles on the road, each task in a service has multiple offloading destinations. We formulate the vehicular service delay by analyzing the task delay, the node performance and the vehicle mobility. The notations are defined in Table I.

A. Task Execution Delay

The delay for each task offloaded to outside the vehicle contains four parts. The computation requirement for the task is f_i^t, which can be denoted by the number of commands. The task execution time is calculated by the total number of CPU cycles required to accomplish the computation task i. The data transfer delay for transmitting the data from the forward task $i-1$ to the task i, which is calculated by input data volume $d_{i-1,i}$ and the bandwidth of the link between the two execution nodes of task i and task $i-1$, i. e. b_{c_{i-1},c_i}. The interactive delay is the time of transmitting the data from the vehicle to the offload destination, which is calculated by interactive data volume d_i^u and the wireless bandwidth of the access network link b_{c_i}. Thus, the execution time of task i in edge computing node c_i is:

$$D_i = \frac{f_i^t}{f_{c_i}} + \frac{d_i^u}{b_{c_i}} + \frac{d_{i-1,i}}{b_{c_{i-1},c_i}} \tag{1}$$

Moreover, for analyzing the bandwidth of the offloading destination, we consider two cases, the vehicle accessing the edge computing node at a BS and the vehicle accessing the edge computing node at an AP or the vehicle accessing another vehicle by WLAN.

The edge computing nodes which are deployed with the BS can be accessed through the 4G technology. If the vehicular user chooses the BS edge computing node c_i to execution task i, the uplink communications are serving multiple users. According to [26], the data transmission rate between the edge computing node and the vehicle v is:

$$b_{c_i} = w \log_2 \frac{1 + q_v g_{v,c_i}}{\varpi_0 + \sum_{u \in U_{c_i}, u \neq v} q_u g_{u,c_i}}, c_i \in BS \tag{2}$$

The edge computing nodes which are deployed with the WLAN AP has limited transmission coverage and a stochastic characteristic for communicating such that a vehicle can offload their tasks only when it stays within the coverage area for at least

TABLE I
THE NOTATIONS DEFINATION

Notations	Defination
f_i^t	The number computation commands of task i
$d_{i-1,i}$	The data volume from task i-1 to current task i
d_i^u	The interactive data volume between the local vehicle and the task execution node i
f_{c_i}	The CPU frequency of edge computing node c_i
b_{c_i}	The link bandwidth between the vehicle to node c_i
b_{c_{i-1},c_i}	The link bandwidth between the nodes of task i and task $i-1$
D_i	The execution time of task i
w	The channel bandwidth of BS edge computing node
q_v	The transmission power of the vehicle v
g_{v,c_i}	The channel gain between the vehicle v and BS c_i
ϖ_0	The background noise power of the BS
W_{min}	The minimum contention window
m_b	The maximum back off stage
p_c	The probability of a collision seen by a transmitted packet
p_t	The probability of a device transmitting a packet in a slot of time
τ_s	The average channel busy time due to packet transmitting
τ_c	The average channel busy time due to collision
L	The average packet payload size of vehicular service
h	The number of devices connect to a AP contending for transmitting data at the same time
μ_i^{-1}	The mean time of the task i execution
$\eta_{c_i}^{-1}$	The mean residence time in one access area of the node
$N_{c_i}^h$	The number of handoff times during task i execution
z^*	The largest transmission range between to vehicles
z_{min}	The minimum inter-vehicle distances
z_{max}	The maximum inter-vehicle distances
p, q, β	The vehicle density parameters
R_{c_i}	The resource usability of the vehicular node c_i for task i
D_h	The migration time when a vehicle changes edge computing node

a certain amount of time with the mobile vehicle going through its coverage region due to limited communication capacity and time-varying wireless channels. The channel model is important since that the data rates of WLAN changes with different network conditions, which cannot be simply classified into "good" or "bad" states. The probability that a device transmits a packet in a slot of time is deduced as:

$$p_t = \frac{2(1-2p_c)}{(1-2p_c)(W_{min}+1) + p_c W_{min}[1-(2p_c)^{m_b}]} \tag{3}$$

Assume τ_s is the average channel busy time due to a successful transmission, and τ_c is the average busy time when the channel is suffering from a collision. L is the average packet payload size. According to [27], the data rate of the edge computing node with WLAN access can be calculated as follow:

$$b_{c_i} = \frac{hp_t L}{(1-p_t)(1+\tau_s) + [(1-p_t)^{1-h} - (1-p_t-hp_t)]p_t}, \quad c_i \in AP \tag{4}$$

B. Performance of Edge Computing Node

The performance of the resource provided by the three types of edge computing node is formulated. The vector $C = (c_1, c_2, \ldots, c_m)$ denotes the m available edge computing nodes surrounding the vehicle, including the neighboring running vehicles, the accessed BS with computing capability and various the roadside units, where the task can be offloaded. The parameter f_{c_i} is the computation capability (i.e., CPU cycles per second) of the execution node c_i when the task is offloaded. When calculating the data transmission delay, the link to the edge node c_i are of two types: the link from the vehicle to an edge computing node which can be calculated by (2) or (4), and the link between two non-mobility edge computing nodes, which can be measured.

As the mobility model is different in cases of vehicle accessing the BS or AP nodes and vehicle accessing the neighboring vehicle nodes, we analyze the impact on task execution of the cases, respectively.

1) BS or AP edge computing nodes. The vehicle may access to multiple edge computing nodes geographical distributed. The edge computing node deployed with BSs and APs usually provides resource to the vehicles that are connected, and the offloaded task may be migrated to a new node according to the vehicle moving. Therefore, considering the node geographical location, the number of traversed access networks during the vehicle movement impacts the service delay. Assume the execution time of task i is exponentially distributed with the mean value of μ_i^{-1}. The mean residence time in one access area of the node $\eta_{c_i}^{-1}$ is the general continuous random variable with the probability density function of $f_{res}(x)$. Assume $f_{res}^*(x) = \int_0^\infty f(x)e^{-\lambda x}dx$ is the Laplace-Stieltjes Transform for the $f_{res}(x)$, and then $\eta_{c_i}^{-1} = \int_0^\infty xf(x)dx$. The number of handoff times during task i execution is $N_{c_i}^h$ which can be deduced according to [28]

$$N_{c_i}^h = \sum_{k=0}^{\infty} kP(k) = \sum_{k=1}^{\infty} \frac{k\mu_i}{\eta_{c_i}}[1 - f_{res}^*(\mu_i)]^2[f_{res}^*\mu_i]^{k-1} = \frac{\eta_{c_i}}{\mu_i} \tag{5}$$

2) Neighboring vehicular nodes. The availability of resources provided by the neighboring vehicles construct as an ad hoc cloud environment. The vehicle node usability is determined by the communication link when all the vehicles are moving on the highway. The inter-vehicle distance is called distance headways. We consider a multi-lane highway, and the vehicle density is time-variant. Assume the transmission range of the vehicles is z^*. When the distance headway of any two vehicles becomes larger than z^*, the communication link is failed, and the offloaded task cannot be used. To model the variation of the distance headway, a discrete-time finite-state Markov chain is used according to [29].

Let $X = \{x_1, x_2, x_3, \cdots, x\}$ be the distance headways between a vehicle and one of its neighbors. The random variables $x_i \in [z_{min}, z_{max}]$ are the distance headways in each time step during the task execution. Here, z_{min} and z_{max} are the minimum and maximum inter-vehicle distances, respectively. Let z be the unit length of distance headway changing. The state X_j can be the ranges of a distance headway between $[x_j, x_j + z]$ at a time step. The state X_{max} corresponding to the distance headway just larger than z_{max}. Then, let $x_j = z_{min} + jz$. Within a time slot, a distance headway in state X_j can transit to the next state, the previous state, or remain in the same state with probabilities p_j, q_j, or l_j, respectively. X_{z^*} state corresponding to the distance headway with the largest communication range z^*. Assume the time step is k seconds. During a task execution time with mean value of u_i^{-1} , the period is divided into $1/k\mu_i$ time steps. If the largest distance headways during the task execution is smaller than z^*, the node usability is 100%. For example, when the vehicles density is large in a traffic jam, the distance headways may be always less than z^*. Otherwise, the node usability is the sum of the state probability when the state number is less than X_{z^*}.

According to [29], the state transition probability is as follows.

$$p_j = p\left[1 - \beta\left(1 - \frac{z_{\min} + jz}{z_{\max}}\right)\right] \tag{6}$$

$$q_j = q\left[1 - \beta\left(1 - \frac{(z_{\min} + jz}{z_{\max}}\right)\right] \tag{7}$$

$$l_j = 1 - p_j - q_j, 0 \le p, q, \beta \le 1 \tag{8}$$

The parameters p, q and β can be set in terms of the vehicle density. Generally, the value of β increases as the vehicle density increases, and thus the dependency on the state value increases. When the vehicle density is at a low level, β is close to zero, and the transition probabilities are independent of the state. We can obtain the one-step transmission matrix:

$$Q = \begin{pmatrix} l_0 & p_0 & 0 & \cdots & \cdots & 0 \\ q_0 & l_1 & p_1 & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 0 & \cdots & l_{z^*} & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 \end{pmatrix} \tag{9}$$

Let the P_j^I denotes the probability of the state X_j in the Ith step state transmission. The initiation state probability vector is $\pi(0) = (P_0^0, P_1^0, P_2^0, \ldots, P_{X_{z^*}}^0, \ldots, P_{X_{max}}^0)$. Assume in the time step ζ, $0 < \zeta \le 1/k\mu_i$, $\pi(\zeta) = (P_0^\zeta, P_1^\zeta, P_2^\zeta, \ldots, P_{X_{z^*}}^\zeta, \ldots, P_{X_{max}}^\zeta)$ is the probability vector of the states. From the property of Markov chain,

$$\pi(\zeta) = \pi(0)Q^\zeta = \pi(0) \times \underbrace{(Q \times \cdots \times Q)}_{\zeta} \tag{10}$$

Then, the usability of vehicular node c_i for task i is

$$R_{c_i} = \sum_{j=0}^{z^*} P_j^\zeta \tag{11}$$

If the task is offloaded to edge computing node deployed with BS or AP, the handoff delay should be added to the task execution delay, assumed as D_h. If the task is offloaded to another vehicle node (VN) and the distance of the two vehicles are larger than z^*, the task may fail, and a re-execution delay at the local vehicle should be added to the task execution delay, i.e. $c_i = lo$

$$D_i' = \begin{cases} D_i + D_h N_{c_i}^h, & c_i \in BS \cup AP \\ D_i + R_{c_i}\left(\dfrac{f_i^t}{f^{c_i=lo}} + \dfrac{d_{i-1,i}}{b_{i-1,c_i=lo}}\right), & c_i \in VN \end{cases} \tag{12}$$

It is generally known that four basic topologies exist in an application, sequence, parallel, selective and loop. These topologies are able to construct the vast majority of composited applications. Assume M tasks are contained in a vehicular service. When the task i is a parallel task, $F(i)$ is dedicated that whether the task i is the longest one.

$$\min D_s = \min \sum_{i=1}^{M} F(i) D_i' \tag{13}$$

IV. Knowledge Driven Service Offloading Decision

DRL extends the well-known traditional reinforcement learning to enable end-to-end system control based on high-dimensional sensory inputs. Different from supervised learning, reinforcement learning does not learn from samples provided by an experienced external supervisor. Instead, it has to operate based on its own experience despite that it faces with significant uncertainty about the environment. In this section, we mainly focus on the offloading decision of the above framework and propose the KD service offloading decision based on an online A3C algorithm to reduce execution time for running vehicular service.

A. Deep Reinforcement Learning Model

A DRL model comprises an agent that interact with the environment based on its observations. At each time step t, the environment is in the state s_t, and the agent executes an action a_t. Then, the environment may transfer to any achievable following stage s_{t+1} by some probability, and the agent receives a reward r_{t+1}. The long-term goal of the agent is to maximize the cumulative reward it earns, by taking a policy π which adapts its action according to its observations. The accumulated return for the step t with a discount factor γ of the future rewards is $R^t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. From the goal of the agent, the value of state s_t is the expected return for following the policy π, which is defined as $V^\pi(s_t) = E[R^t | s = s_t]$. When taking use of the deep neural network as the function approximate, its parameters is denoted as θ. Currently, there are various RL algorithms for updating θ, including DQN, Deep Deterministic Policy Gradient (DDPG) and A3C.

The Actor-Critic architecture derives from the policy-based model-free methods, which combines the advantage of Q-learning and policy gradients. It directly parameterizes the policy, denoted by $\pi(a_t|s_t;\theta)$ and updates the parameter θ by calculating the gradient ascent on the variance of the expected accumulated return R^t and the learned value function under the policy π, i.e. $R^t - V^\pi(s_t)$. In the Actor-Critic architecture, the actor is the policy function $\pi(a_t|s_t;\theta)$, which decides the action currently based on learning the policy that can achieve the highest reward. Then the environment changes, and the agent receives corresponding reward. While, the critic uses this reward to evaluate the current policy according to the TD-error between current reward and the estimation of the value function $V(s_t;\theta_v)$. The value function can be calculated by value-based learning method. For the DRL model, the policy function and the value function are both neural networks. The feedback of TD-error is used to update the actor network parameter θ for adding the probability of selecting the better action as well as the critic network parameters θ_v for obtaining more accurate estimation value. By this way, the AC algorithm learns the policy function and value function together. Along with the iteration, the critic archives more accurate estimation and the actor makes better selection and finally, the system converges.

The A3C algorithm based on AC algorithm utilizes asynchronous multi-threads as multiple actors to train DNN reliably. The multiple actor learners running in parallel can explore different parts of the environment with different policies. By this way, these updates of parameters are less correlated in time than a single agent, so that the replay memory of traditional DQN is not needed. Moreover, the training time is reduced.

Each agent keeps its own parameters of the actor network and the critic network, and also a global actor network and a global critic network. Each time the agent updates its parameters of the two networks, then it submits the parameters to the global networks. When the global network receives the new parameters, it transmits them to the agents. After several iterations, the two networks converge.

The A3C algorithm uses k steps rewards to update the parameters. The critic network updates the parameters θ_v of the value function $V(s_t;\theta_v)$ and make the value function to approximate the real reward [20].

$$\begin{aligned} \hat{G}(s_t) &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{k-1} r_{t+k-1} \\ &\quad + \gamma^k V(s_{t+k};\theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) \end{aligned} \tag{14}$$

The A3C algorithm follows the AC model and defines the advantage function as the estimation of the difference of the real reward and the value function that is the estimation at the state s_t under the parameters of θ_v.

$$A(s_t, a_k; \theta, \theta_v) = \hat{G}(s_t) - V(s_t;\theta_v) \tag{15}$$

When update the function value, we try to minimize the difference, i.e.

$$\min_{\theta_v} [\hat{G}(s_t) - V(s_t;\theta_v)]^2 \tag{16}$$

Then the critic network update is performed by the gradient

$$d\theta_v \leftarrow d\theta_v + \frac{\partial A(s_t, a_k; \theta', \theta_v')^2}{\partial \theta_v'} \tag{17}$$

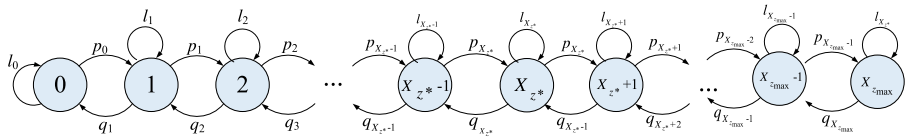


Fig. 3. The state transfer of vehicular distance headway.

The actor network update is performed by the gradient

$$d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_t \mid s_t; \theta') A(s_t, a_k; \theta', \theta'_v) + \delta \nabla_{\theta'} H(\pi s_t; \theta') \tag{18}$$

Here, H is the entropy. The A3C algorithm uses the entropy of the policy π to the objective function of network update, which prevents the premature convergence to suboptimal deterministic policy. The hyperparameter δ controls the strength of the entropy regularization term [20].

B. A3C Based Vehicular Service Offload Algorithm

We enable the DRL model to learn the long-term optimal service offloading knowledge, which is the mapping of the environment observation to the offloading destination, represented by a deep neural network. The observation includes the performance of the edge computing nodes, the status of the vehicular nodes, the vehicle moving speed and the task requirements. The neural network provides an expressive and scalable way to incorporate a rich variety of observations into the service offloading policy. We consider the offloading decision function in the vehicle act as the agent of the DRL model, which interacts with the vehicular edge computing environment through a sequence of observations, actions and rewards. The goal is to select actions in a fashion that maximizes cumulative future reward for all of the tasks in a service.

The state space of the vehicular edge computing is defined as follows: $S = (T, C, v)$. T is a tensor of the task profile, which is defined as $T = (f_i^t, d_i^u, d_{i-1,i})$. C is the tensor that expresses the current state of the offloading destination nodes including the edge computing nodes and vehicular nodes, $C = (c_{type}, f_{c_i}, b_{c_i}, N_{c_i}^h, R_{c_i})$. The parameter c_{type} is the node type including BS edge computing node, AP edge computing node, and vehicle node. The v is the moving speed of the vehicle. The action space is $A = (a_{local}, a_1, \ldots, a_m, \ldots)$, denoting that a task is executed locally on the vehicle, i.e., $a_{local} = 1$, or offload to the accessible edge computing nodes n (i.e., $a_m = 1$).

Since the number of accessible edge computing nodes for a vehicle is larger than the action space, we choose fixed number of the three types of nodes according to their computation power and construct the action set. Otherwise, if the number of nodes is less than the length of the actions set, we set some pseudo nodes with poor performance to fill the action space. The reward for each decision slot r_t is determined by task execution delay in real set, i.e. D_i'. The vehicular mobility may result in the task migration or node unavailable. Additionally, when there exist the parallel tasks, only the task with longest delay is considered in the learning process. The goal of the service offloading is achieving the minimized expected cumulative discounted reward for all of the M tasks.

$$J = E\left[1/M \sum_{t=1}^{M} \gamma^{M-1} r_t\right] \tag{19}$$

Algorithm 1: The Vehicular Service Offloading Decision Algorithm.

if *training at the powerful edge computing node* **then**
  Initialize the critic network with parameter θ_v
  Initialize the actor network with parameter θ
  Obtain the threads of the CPU
else
  Fetch the corresponding model for a service from edge computing node
end
while *DataSet is not empty or All vehicles running the service* **do**
  Set the gradient of the two networks $d\theta = 0$ and $d\theta_v = 0$
  Synchronous thread parameters by global parameters $\theta' = \theta, \theta_v{}' = \theta_v$
  Obtain the vehicular edge computing nodes status and the task profile
  Construct the environment state S_t
  for $t = 1, t \leq M$ **do**
    Select the task offload location a_t according to policy $\pi(a_t|s_t;\theta')$ in actor network
    Calculate the reward r_t and construct the new environment state S_{t+1}
    t=t+1
  end
  Obtain the $R = V(s_t, \theta'_v)$ from the critic network
  for $t = M, t \geq 1$ **do**
    $R = r_t + \gamma R$
    Calculate the accumulate gradient $d\theta_v$ for critic network by (17)
    Calculate the accumulate gradient $d\theta$ for actor network by (18)
    t=t-1
  end
  if *training at the BS edge computing node* **then**
    Asynchronous update θ_v and θ, respectively
  else
    Send the $d\theta_v$ and $d\theta$ to BS edge computing node
  end
end

The A3C based vehicular service offloading algorithm is depicted as follows. The DRL model is firstly training at the powerful edge computing node by the generated service data. Then, when a vehicle initiates a service, the corresponding pretrained model is fetched, and the online learning by the actor network and the critic network is performed. The vehicles running the same service can update the parameters of the DRL asynchronously. Finally, the offloading decision knowledge can be online learned by vehicle, which can adapt the environment changing.

In the online learning period, the vehicles perform asynchronous exploration when running the service, and they update the new model to the edge computing node periodically. Actually, the synchronization of parameters from vehicles to edge computing node may consume lots of communication resource. We utilize some distributed machine learning method to solve this problem, including the ternary gradients [30] which can compress the parameter data, and the optimal synchronization which can reduce synchronization times [31].

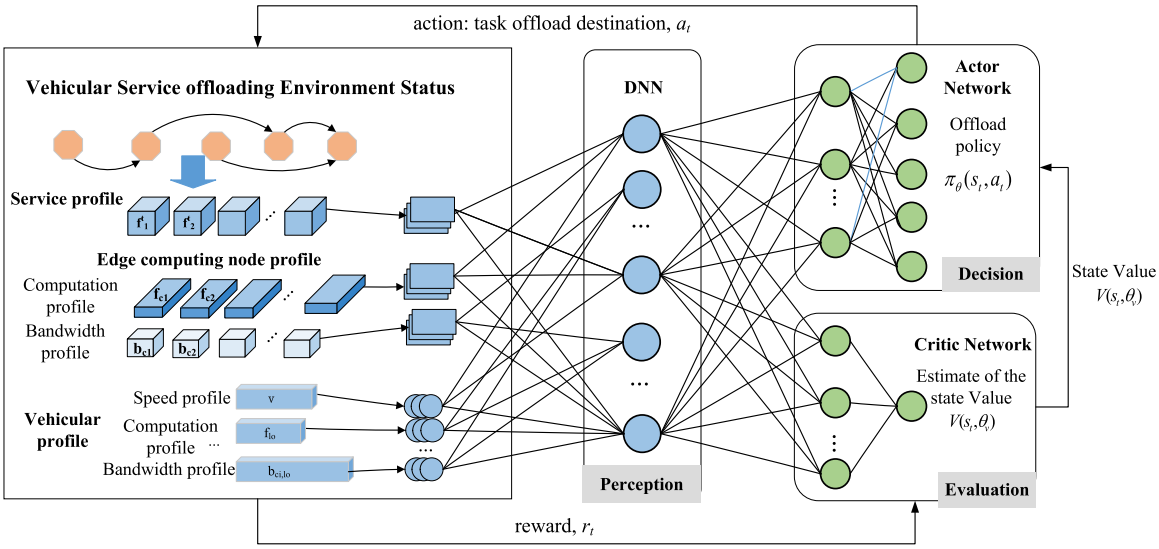


Fig. 4. The deep reinforcement learning model for vehicular service offloading decision.

V. Performance Evaluation

In this section, the performance of the KD service offloading decision is evaluated. First, we introduce the simulation scenarios for the vehicular edge computing environment and the vehicular applications used to train the DRL model. Second, we evaluate the usability of the edge computing nodes based on the proposed mobility model with difference parameters. Then, the DRL model for offloading decision is analyzed with different parameters. Finally, the service offload decision policy based on the A3C algorithm is evaluated by comparing with greed algorithm.

A. Simulation Scenarios

In the simulation, we generate various service scenarios that include different number of tasks and the three types of edge computing nodes for training the DLR model. The vehicular environment contains two BS edge computing nodes, with frequency of 560 and 676 , two AP edge computing nodes with frequency of 526 and 430, and six accessible neighboring vehicular nodes with frequency of 124, 120, 177, 144, 165 and 130. When each task execution, the computation capabilities of theses nodes changes according to a normal distribution with standard deviation as 5. The bandwidths between BS to BS, BS to AP, and AP to AP, AP to vehicle are set to be 100 M. The bandwidth of BS to vehicle is set to be 50 M, while vehicle to vehicle is set to be 300 M.

We design several complicated services for vehicular users which contains multiple tasks deployed in Docker. For example, the location sight recognition service for vehicle drivers includes object detection [32], [33], feature learning [34], image auto-annotation [35], image segment [36], and recommendation based on location [37]. We generate the dataset according to these services including group of tasks for each service. The service property data contains the required CPU cycles, user transmitting data volume and dependency data volume for the previous task. Moreover, two or three tasks can be emerged into one task in the test, so that they may be deployed in one node. Therefore, in the following test, the task number may change from 5 to 30 according to the setting.

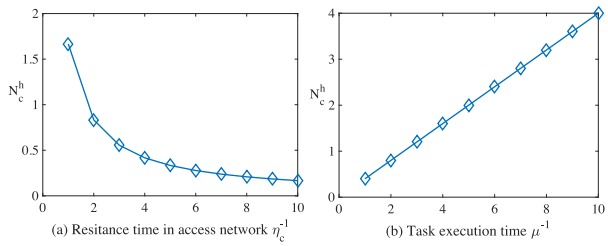


Fig. 5. The handoff times when accesses to the non-mobility edge computing nodes.

B. Usability of Offload Impacted by Vehicular Mobility

We generate users moving parameters according to the mobility model introduced in Section IV. We first evaluate the impact on the service delay of BS edge computing nodes and AP edge computing nodes. Fig. 5(a) and 5(b) depict the handoff times during a task by effect of parameters resident time and task execution time. When the resident time $\eta_{c_i}^{-1}$ increases, the handoff times decrease; but when the mean execution time of task i μ_i^{-1} increases, the handoff times rise. Therefore, if the vehicle keeps a high moving speed and the task execution time is long, the vehicular user may suffer a task migration, which adds the service delay.

Moreover, according to the model of vehicle mobility in Section III.B, we analyze the vehicle node usability impacted by parameters of β, Z^* and the p. In Fig. 6(a), the Z^* is set to be the 30th states, with the largest states of the distance headway as 40. We can see when the vehicular density parameter β increases, the node usability increases. When the value of β is close to zero, the vehicle density is at a low level, and the transition probabilities are independent of the state. Since high density results in the large dependency of the mobility transfer

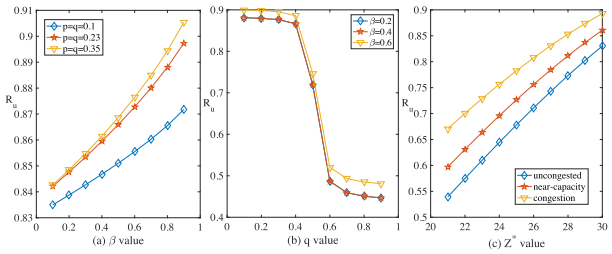


Fig. 6. The usability of the vehicular edge computing nodes.

state, the node usability may be bigger when the p and q with little higher values. Generally, the parameter β does not impact the node usability extremely. In the vehicle mobility model, the parameter p indicates the probability of the distance headway increasing. When the vehicle speed is fast, the headway may increase, which results in the higher probability of the neighboring node inaccessible. Then, the usability of the vehicular node decreases with the rising of parameter p, which is depicted in Fig. 6(b). In addition, 6(c) shows the node usability with the state numbers of the largest accessible distance headway under different traffic conditions, i.e. uncongested, near-capacity, congestion. When the largest accessible distance headway increases, i.e. Z^*, the usability of the vehicular node increases linearly in the mobility model. Then in the following analysis, we use the different parameter combination to generate the vehicular nodes.

C. Analysis of the DRL Model

We analyze the different parameters of DRL model for understanding the convergence and gains of the KD offloading algorithm. Firstly, we use the five layers DNN as both of the actor network and the critic network for learning the data distribution of the task offloading. The neural cell of the input layer is the dimension number of environment state, and the second layer and the third layer both contain 64 neural cells. The output of the critic network is one neural cell which express the value of the input state. The output layer of the actor network uses the Softmax function to obtain the probability of the actions under the policy. We use the ReLU (Recitified Linear Unit) as the activation function. To avoid the overfitting of the deep neural network, the learning rates of both actor network and critic network are set to be 0.001, respectively. The model is training on the PC server with 2.3GHz 4 cores CPU and 8 G memory, which provides 4 threads for the asynchronous learning.

We generate the services containing 10 serial tasks with data dependency. The CPU requirement of the 10 tasks are as follows: four tasks need 5K commands, 3 tasks need 2K commands, and 3 tasks need 9K commands. Moreover, for generating diversified services, the requirements follows a normal distribution with standard deviation as 500. The data transmitted between the tasks are set to be 2G and follows a normal distribution with mean of 500. The results of KD service offloading decision model in 80000 service samples during the DRL training process are depicted in Fig. 7. The Fig. 7(a) shows the results of long-term reward which is expressed by average task delay of the complex service. Fig. 7(b) and 7(c) show the loss values of the actor network and the critic network. We can see the loss values

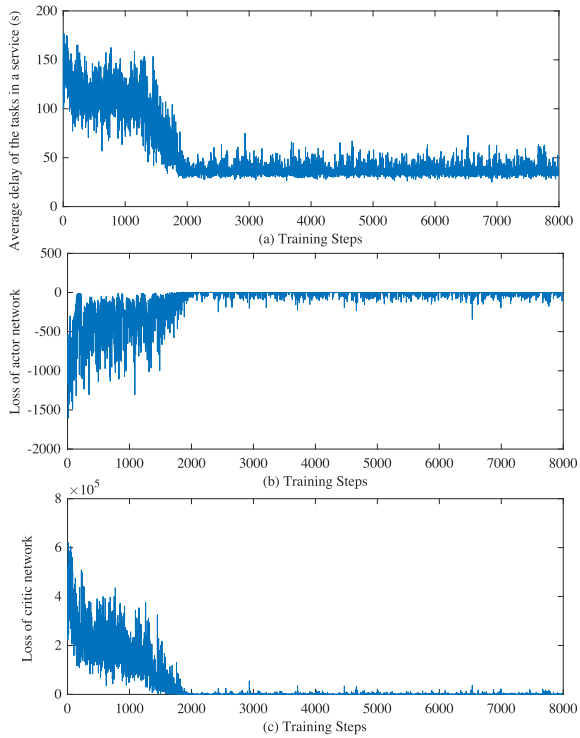


Fig. 7. The training results of the KD service offload algorithm during training.

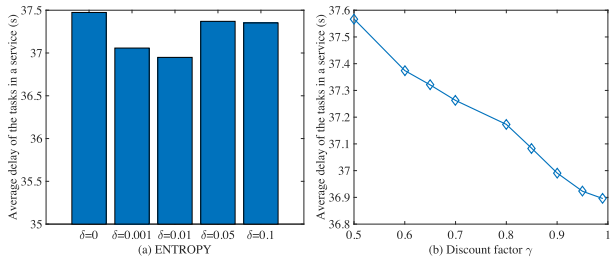


Fig. 8. The performance of the KD service offload algorithm by different parameters.

of both the actor network and the critic network are approaching to zero after about 2000 epochs. At the same time, the average task delay of the service converges at the long-term optimal.

In Fig. 8(a), we compare the average task delay by different entropy hyperparameter δ value in the DRL model. From the results, choosing δ as 0.01 can archive the best result, which can prevent the premature convergence to suboptimal policy by appropriate strength of exploration. In Fig. 8(b), the average task delay decreases along with the discount factor γ. For the complicated vehicular service, we choose the γ with large value which means the more future execution tasks performance is considered when selecting the current offloading destination. However, when the γ value increases, the model should take more service samples to train for convergence. For example, when we choose $\gamma = 0.99$, 60000 samples are generated to train.

D. Service Delay Comparison

We compare the KD offloading decision algorithm with the greedy offloading decision algorithm. The greedy offloading

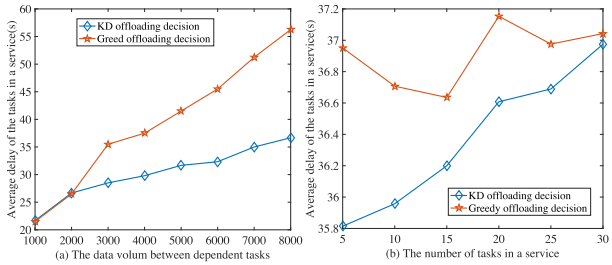


Fig. 9. The comparison of the KD offloading decision and greed offloading decision.

decision algorithm always selects the optimal destination, i.e. the node with least delay for each task in a service. We measure the average delay for all tasks achieved by the greedy offloading decision algorithm and the KD service offloading decision after the training.

From the Fig. 9(a), comparing with the greedy offloading decision algorithm, we can see the KD offloading decision achieves less service delay. Since the data transmission between the former task and the latter task impacts the task delay, choosing the optimal destination for the former maybe not good for the latter task. The KD offloading decision can learn the distribution of the task data dependency by the DRL model, and almost always choose a proper destination for the large data transmission tasks. In Fig. 9(a), the advantage for average task delay by KD offloading decision and the greedy offloading decision increases along with the data volume between the depended tasks in a service. With the more data transmitted between the task, the average task delay by greedy offloading increases. But the delay of greedy offloading decision increases significantly comparing with KD offloading. The reason is greedy offloading decision ignores the future, which results that its offloading decision is not the long-term optimal policy. The KD offloading decision learn the action selection policy by DRL model, which may choose the the non-minimum delay destination for the current task but considers the performance of future tasks.

In Fig. 9(b), we use different complicated service topologies to simulation. The average task delay for the greedy offloading decision does not change significantly when the number of tasks in service rises. While, the average task delay for the KD offloading decision increases with the service complexity. Because the more complexity service results in the harder evaluation of the future reward, the advantage of KD offloading decision with the long-term optimal policy may decrease.

VI. Conclusion

This paper targets the problem of service offloading decision in vehicular edge computing environment. Our work goes beyond existing approaches by considering knowledge-driven method to obtain the optimal offloading policy. The current vehicular services always contain multiple tasks with data dependency. We propose a KD service offloading framework by exploring the DRL model to find the long-term optimal service offloading policy. Focusing on the types of vehicular edge computing nodes, the service delay as learning reward is formulated which contains the factors of node accessible due to the vehicle mobility. The offloading decision model for each service is trained at the powerful edge computing nodes, and then distributed to the vehicle. The vehicles perform asynchronous online learning when it running the service, and updates the new model to the edge computing node. By the feedback reward in each running time, the online learning KD service offloading decision framework can adapt the environment changing. We generate several datasets for training the KD service offloading decision. The advantage of the KD service offloading decision is obviously when there are more data dependency between tasks in a service.

References

[1] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructure, "*IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[2] R. Kim, H. Lim, and B. Krishnamachari, "Prefetching-based data dissemination in vehicular cloud systems," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 292–306, Jan. 2016.

[3] C. Wang, Y. Li, D. Jin, and S. Chen, "On the serviceability of mobile vehicular cloudlets in a large-scale urban environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2960–2970, Oct. 2016.

[4] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[5] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 15–19, 2018, pp. 1871–1879.

[6] Z. Ma *et al.*, "The role of data analysis in the development of intelligent energy networks," *IEEE Netw.*, vol. 31, no. 5, pp. 88–95, Sep. 2017.

[7] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Apr. 2017.

[8] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[10] X. Lyu *et al.*, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.

[11] Y. Kim, J. Kwak, and S. Chong, "Dual-side optimization for cost-delay tradeoff in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1765–1781, Feb. 2018.

[12] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11 339–11 351, Dec. 2017.

[13] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[14] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[15] N. Kato *et al.*, "The deep learning vision for the heterogeneous network traffic control proposal, challenges, and future perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 146–153, Jun. 2017.

[16] B. Mao *et al.*, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.

[17] L. Lei, L. You, G. Dai, T. Xuan Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled HetNet," in *Proc. Int. Symp. Wireless Commun. Syst.*, Aug. 28–31, 2017, pp. 449–453.

[18] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, May 2018, pp. 1–6.

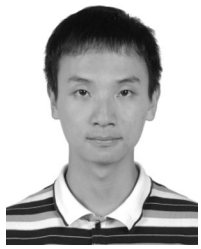
[19] V. Mnih, K. Kavukcuoglu, and D. Silver, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–534, Feb. 2015.

[20] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, Jun. 19–24, 2016, pp. 1928–1937.
[21] Y. He *et al.*, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10 433–10 444, Nov. 2017.
[22] H. Mao *et al.*, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, Nov. 09–10, 2016, pp. 50–56.
[23] N. Liu *et al.*, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 372–382
[24] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 21–25, 2017, pp. 197–210.
[25] Y. Wei, F. Richard Yu, M. Song, and Z. Han, "User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, Jan. 2018.
[26] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[27] T. Liu, F. Chen, Y. Ma, and Y. Xie, "An energy-efficient task scheduling for mobile devices based on cloud assistant," *Future Gener. Comput. Syst.*, vol. 61, pp. 1–12, Aug. 2016.
[28] Y. B. Lin, "Reducing location update cost in a PCS network," *IEEE Trans. Netw.*, vol. 5, no. 1, pp. 25–33, Feb. 1997.
[29] K. Abboud and W. Zhuang, "Stochastic analysis of single-hop communication link in vehicular Ad Hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2297–2307, Oct. 2014.
[30] W. Wen *et al.*, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, Dec. 4–9, 2017, pp. 1509–1519.
[31] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2018, pp. 63–71.
[32] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of Co-salient objects by looking deep and wide," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 215–232, Nov. 2016.
[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[34] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, "Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 6, pp. 3325–3337, Jun. 2015.
[35] J. Han, D. Zhang, X. Hu, L. Guo, J. Ren, and F. Wu, "Background prior-based salient object detection via deep reconstruction residual," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1309–1321, Aug. 2015.
[36] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, Dec. 2016.
[37] Y. Zheng, Y. Wang, L. Zhang, J. Wang, and Q. Qi, "A tag-based integrated diffusion model for personalized location recommendation," in *Proc. Int. Conf. Neural Inf. Process.*, Nov. 14–17, 2017, pp. 327–337.

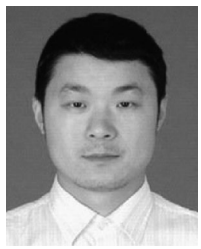
Qi Qi received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2010. She is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has authored or coauthored more than 30 papers in international journal, and is the recipient of two National Natural Science Foundations of China. Her research interests include edge computing, mobile cloud computing, Internet of Things, ubiquitous services, deep learning, and deep reinforcement learning.

Jingyu Wang received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. He is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. He has authored or coauthored more than 50 papers in international journal, including the IEEE COMMUNICATION MAGAZINE, the IEEE SYSTEM, and so on. His research interests include broad aspects of SDN, big data processing and transmission, overlay networks, multi-media services, and traffic engineering.

Zhanyu Ma received the Ph.D. degree from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 2011. Since 2014, he has been an Associate Professor with the Beijing University of Posts and Telecommunications, Beijing, China. Since 2015, he has also been an adjunct Associate Professor with Aalborg University, Aalborg, Denmark. From 2012 to 2013, he was a Postdoctoral Research Fellow with the School of Electrical Engineering, KTH. His research interests include pattern recognition and machine learning fundamentals with a focus on applications in multimedia signal processing, and IoT data mining.



Haifeng Sun received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017. He is currently a Lecturer with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include broad aspects of AI, NLP, big data analysis, object detection, deep learning, deep reinforcement learning, SDN, and processing.



Yufei Cao received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. In 2008, he joined the EBUPT Information Technology Company, Beijing, China, where he is currently working as a Research Engineer and the Manager of the IoT Department. He is responsible for the project of Intelligent IoT in smart home, intelligent vehicular network, edge computing platform for smart gird, and so on. His research interests include vehicular network, Internet of Things, cloud computing, communications software, and 5G core network.

Lingxin Zhang received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2018. She is currently working toward the master's degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include deep reinforcement learning, MDP, deep learning, parallel computing, and resource management.

Jianxin Liao received the Ph.D. degree from the University of Electronics Science and Technology of China, Chengdu, China, in 1996. He is currently the Dean of the Network Intelligence Research Center and a Full Professor with the State Key laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. He has authored or coauthored hundreds of research papers and several books. He has won a number of prizes in China for his research achievements, which include the Premiers Award of Distinguished Young Scientists from National Natural Science Foundation of China in 2005, and the specially invited Professor of the "Yangtse River Scholar Award Program" by the Ministry of Education in 2009. His main research interests include cloud computing, mobile intelligent network, service network intelligent, networking architectures and protocols, and multimedia communication.