Asynchronous Deep Reinforcement Learning for Data-Driven Task Offloading in MEC-Empowered Vehicular Networks

Penglin Dai[*], Kaiwen Hu[*], Xiao Wu[*], Huanlai Xing[*] and Zhaofei Yu[†]
[*] School of Information Science and Technology, Southwest Jiaotong University, Chengdu, 611756, China
[†] School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China
Email:{penglindai@swjtu.edu.cn, kaiwenhu_swjtu@163.com, hxx@home.swjtu.edu.cn,
wuxiaohk@gmail.com, yzf714@126.com}

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488886

***Abstract*—Mobile edge computing (MEC) has been an effective paradigm to support real-time computation-intensive vehicular applications. However, due to highly dynamic vehicular topology, these existing centralized-based or distributed-based scheduling algorithms requiring high communication overhead, are not suitable for task offloading in vehicular networks. Therefore, we investigate a novel service scenario of MEC-based vehicular crowdsourcing, where each MEC server is an independent agent and responsible for making scheduling of processing traffic data sensed by crowdsourcing vehicles. On this basis, we formulate a data-driven task offloading problem by jointly optimizing offloading decision and bandwidth/computation resource allocation, and renting cost of heterogeneous servers, such as powerful vehicles, MEC servers and cloud, which is a mixed-integer programming problem and NP-hard. To reduce high time-complexity, we propose the solution in two stages. First, we design an asynchronous deep Q-learning to determine offloading decision, which achieves fast convergence by training the local DQN model at each agent in parallel and uploading for global model update asynchronously. Second, we decompose the remaining resource allocation problem into several independent subproblems and derive optimal analytic formula based on convex theory. Lastly, we build a simulation model and conduct comprehensive simulation, which demonstrates the superiority of the proposed algorithm.**

***Index Terms*—Mobile edge computing, vehicular networks, task offloading, asynchronous deep reinforcement learning**

I. Introduction

Mobile edge computing (MEC) has been an effective paradigm to support real-time intelligent transportation systems (ITSs) by providing computation, communication and caching resources at the edge devices [1, 2]. These ITS services, such as autonomous driving, video surveillance and traffic control, are always data-driven and computation-intensive tasks and have to process a large amount of traffic data carried by mobile vehicles in a large-scale road networks. Though the MEC servers are deployed close to mobile terminals, it still imposes a strict requirement of computation and bandwidth resources beyond the local capability of MEC servers. Particularly, due to unique characteristics of vehicular networks [3], such as highly dynamic network topology and uneven distribution of vehicle density, the users may suffer from service delay caused by unpredictable network congestion and unbalanced workload distribution. Heterogeneous computation servers such as powerful vehicles, MEC servers and cloud, are expected to be integrated for satisfying the increasing demand of ITS services. Hence, it is still non-trivial to investigate task offloading mechanism in MEC-based vehicular networks.

In the last decades, great efforts have been paid on developing types of computation offloading mechanisms in vehicular networks [4–6], which is used for determining the servers for processing tasks offloaded from terminal users. Some researchers formulated the frameworks of vehicular edge computing (VEC) [7], where computation resources of powerful vehicles are utilized for local task processing via vehicle-to-vehicle (V2V) communication. However, the VEC cannot provide stable services due to unreliable V2V connection and dynamic computation capacity. To improve reliability, some other researchers proposed MEC-based offloading mechanisms [8], such as multi-user game model [9], and reinforcement learning [10], where the MEC servers deployed at roadside handle tasks offloaded from neighboring vehicles via vehicle-to-infrastructure (V2I) communication. However, these studies only focused on optimizing offloading decision, where the edge resource competition among multiple tasks is neglected. To solve this issue, some studies designed several resource allocation mechanisms to ensure the QoS of users, such as convex-based optimization [11] and Semi-Markov Decision Process [12]. In addition, some researchers formulated joint optimization model by integrating task offloading and resource allocation [13] and developed several scheduling algorithms, such as deep Q-learning [14] and ADMM [15]. However, these joint optimization models are based on non-linear programming, which are typically NP-hard and cannot be optimally solved in polynomial time [16, 17]. In particular, these existing studies are based on centralized or decentralized scheduling with synchronous information exchange, which cannot be applied to large-scale vehicular networks due to overhigh communication overhead and scheduling complexity.

Based on the motivation above, this paper investigates a novel service scenario of data-driven task offloading in MEC-empowered vehicular networks, where traffic data sensed by crowdsourcing vehicles are offloaded to heterogeneous computation servers, such as neighboring computing vehicles, MEC servers and cloud. Specifically, data-driven tasks are characterized by the amount of crowdsourcing data set and the required computation resources, which are divided into multiple subtasks according to data distribution among crowd-

Corresponding author: Penglin Dai

978-1-6654-0325-2/21/$31.00 ©2021 IEEE

sourcing vehicles. To model the heterogeneity of computation servers, three different types of transmission and computation models for computing vehicles, MEC servers and cloud, are formulated to simulate task offloading procedure. Particularly, the renting cost of bandwidth and computation resources is also considered, which differs in types of computation servers. In addition, the MEC servers are responsible for making scheduling decision in a distributed way, including offloaded server and resource allocation, for nearby crowdsourcing vehicles within V2I coverage, which aims at minimizing the service time and cost of completing tasks simultaneously by fully exploiting heterogeneous computation and communication resources. Accordingly, the following issues have to be addressed. First, we have to strike a balance between these two objectives since they contradict with each other. It is because that to complete the task in a shorter time, higher cost has to be paid for renting more computation resources. Second, to keep up-to-date global knowledge, MEC servers require frequent communication with each other, as well as mobile vehicles, which may result in over-high communication overhead. Third, the scheduling optimization synthesizing task offloading and resources allocation may require extremely high time complexity, especially in large-scale network .

Therefore, this paper develops an asynchronous task offloading algorithm inspired by the basic idea of both asynchronous advantage actor-critic (A3C) and deep Q-network (DQN), which achieves fast convergence in an asynchronous way. Further, the optimal solution of resource allocation is derived based on decomposition method and convex theory, which is implemented at MEC server in a distributed way. The main contributions of this paper are outlined as follows.

- We investigate a service scenario of MEC-based vehicular crowdsourcing where heterogeneous bandwidth and computation resources, including computing vehicles, MEC servers and cloud, are exploited for processing traffic data sensed by crowdsourcing vehicles. In particular, the MEC server is regarded as local agent and independently makes offloading decision of pending tasks, as well as resource allocation of computation servers.
- We formulate the data-driven task offloading (DTO) problem as a mixed-integer programming model by jointly considering task offloading decision and resource allocation among heterogeneous servers, as well as the corresponding renting cost, which aims at minimizing average service time (AST) and average service cost (ASC), simultaneously. In particular, the characteristics of data-driven tasks, and three types of task transmission and computation models for heterogeneous servers, are theoretically modelled, respectively.
- To reduce time complexity, we derive the solution of the DTO in two stages. First, we develop an asynchronous deep Q-network (ADQN) algorithm to enable each MEC server make offloading decision in a distributed way. By utilizing the advantage of A3C, each agent can train its local DQN model in parallel and share its local knowledge for global model update asynchronously, which can achieve fast convergence. Second, we further decompose the remaining resource allocation problem into three small subproblems and derive the optimal analytic formula based on Karush–Kuhn–Tucker (KKT) condition [18], which is implemented at each MEC server.
- We build a comprehensive simulation model by integrating real-world map, traffic simulator and python-based task offloading module. Further, we implement the proposed algorithm, as well as two competitive solutions. The simulation results demonstrate the superiority of the proposed algorithm compared to two competitive algorithms in a wide range of service scenarios.

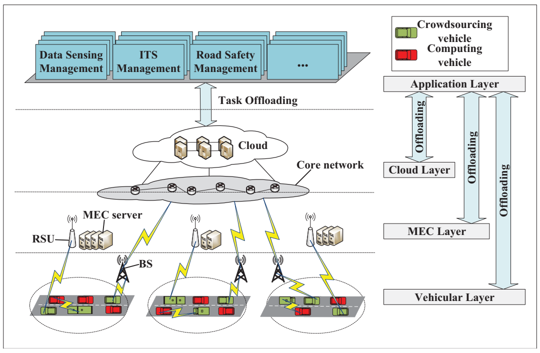


Fig. 1: Service Architecture for Data-Driven Task Offloading in MEC-Empowered Vehicular Networks

The rest of this paper is organized as follows. Section II presents the system model in detail. Section III formulates the optimization problem. Section IV proposes the solution. The simulation result is shown in Section V. Finally, Section VI gives the conclusion.

II. SYSTEM MODEL

In this section, we present a service architecture for data-driven task offloading in MEC-empowered vehicular network. As shown in Fig. 1, the service architecture consists of application layer, vehicular layer, MEC layer and cloud layer, respectively, which are introduced as follows.

In application layer, various types of ITS services, such as traffic signal control, abnormal vehicle detection and traffic flow prediction, are supposed to be data-driven tasks, which require for processing traffic data collected by crowdsourcing vehicles distributed among road networks. Accordingly, a data-driven task is divided into multiple subtasks based on data distribution. Each subtask is associated with a set of traffic data sensed by a crowdsourcing vehicle and can be processed in parallel. The task can only be completed when all the subtasks are completed. With the advantage of MEC-based service architecture, the subtasks can be offloaded to nearby computation server without centralized processing at the cloud, which greatly reduces transmission time and improves computation efficiency. In this paper, the task offloading procedure includes task transmission and computation, while the retrieval time of computation result is neglected due to its trivial size, which is a common setting in related literatures [19, 20].

In vehicular layer, mobile vehicles are classified into two categories: crowdsourcing vehicles and computing vehicles. On one hand, crowdsourcing vehicles are equipped with various sensors and able to collect types of traffic data. A crowdsourcing vehicle can cache data associated with several types of subtasks, which has to be offloaded to nearby computation servers via wireless communication. On the other hand, computing vehicles are assumed to own one processor and can undertake local computing for subtasks within V2V communication range. However, due to the limited computation capability, at most one subtask is allowed to offload to a computing vehicle at each time. Accordingly, the renting cost of computing vehicles is the cheapest.

In MEC layer, the MEC server is deployed at the RSU and plays two roles: computation server and local scheduler, respectively. As the computation server, the MEC server is able to process multiple pending tasks simultaneously offloaded via wireless V2I communication. Due to the mobility of vehicles, the subtask has to be completely uploaded within limited V2I connection time. Therefore, the competition for wireless bandwidth and computation resource may occur among multiple pending subtasks, which will be modeled in the following section. The renting cost of MEC servers is assumed to be higher than computing vehicles. Further, as the local scheduler, the MEC server is responsible for making the offloading decision of each subtask, including offloaded server, wireless bandwidth and computation resource allocation, based on collected task information via overhearing heartbeat message periodically broadcast by crowdsourcing vehicles.

In cloud layer, the cloud server is resided in backbone network and assumed to own unlimited computation resources. The vehicles can offload their subtasks to the cloud server via cellular interface. Due to wide deployment of base stations (BSs), the vehicles can always access to the cloud server but have to pay the transmission cost, which is proportional to the size of uploaded data. Particularly, for vehicles outside the V2I coverage or fail to complete the associated subtasks within the connection time of MEC servers or computing vehicles, they have to choose the cloud as the offloaded server. Vehicles also have to pay the renting cost, which is proportional to allocated computation resources from the cloud. The renting cost of the cloud is the highest among all the computation servers.

Based on the above observation, the completion of a task needs the coordination among multiple MEC servers and the integration of heterogeneous resources. Therefore, a distributed mechanism is urgently needed to be designed and implemented at each MEC server to minimize both service time and service cost by optimizing the utilization of heterogeneous wireless bandwidth and computation resources of vehicular, MEC and cloud layers in a comprehensive way.

III. Problem Formulation

In this section, we introduce data-driven task offloading (DTO) problem in detail. In general, the procedure of processing data-driven tasks is presented, which includes task communication and computation models for MEC server, cloud and computing vehicles, respectively. On this basis, we define the objective function and formulate problem definition. Before elaboration, the basic notations are introduced as follows.

A. Preliminary

The set of MEC servers is denoted by M. Each MEC server $m \in M$ is characterized by a two-tuple (p_m, b_m), where p_m and b_m are the computation capability and the V2I wireless bandwidth, respectively. Then, the set of vehicles in the V2I coverage of m is denoted by V_m, which can be further divided into two categories: crowdsourcing vehicle set and computing vehicle set, denoted by V_m^s and V_m^p, respectively. Accordingly, the set of crowdsourcing and computing vehicles neighboring to v is denoted by V_v^s and V_v^p, respectively. The crowdsourcing vehicle $v \in V_m^s$ has a cached data set denoted by D_v. Each computing vehicle $v \in V_m^p$ has computation capacity p_v.

Further, the set of data-driven tasks is denoted by R. Each task $r \in R$ is characterized by two-tuple (D_r, cr_r), where D_r and cr_r are required data set and computation resource, respectively. Particularly, each task r is supposed to be divided into multiple subtasks based on the distribution of D_r among crowdsourcing vehicles. Specifically, each subtask r_v of r is associated with a data set D_{r_v} cached by a crowdsourcing vehicle v. Accordingly, the required data set D_{r_v} is denoted by $D_r \cap D_v$ and the required computation resource cr_{r_v}, is proportional to the size of D_{r_v}, i.e., $cr_{r_v} \propto ||D_{r_v}||$. For simplicity, we use $||r||$ and $||R||$ to denote the number of subtasks in r and tasks in R, respectively.

In addition, the offloading decision of each subtask r_v is represented by a set of notations $a_{r_v}^l$, $\forall l \in N_v$, where each $a_{r_v}^l$ indicates whether r_v is offloaded to computation server l or not and N_v is the set of computation servers available to v, which is formulated as follows.

$$N_v = \{l | l \in V_v^p \cup m \cup c\} \tag{1}$$

Each subtask is assumed to be indivisible and has to be only assigned to one of the computing servers, which is formulated as follows.

$$a_{r_v}^l \in \{0, 1\}, \forall l \in N_v \tag{2}$$

$$\sum_{\forall l \in N_v} a_{r_v}^l = 1, \forall r \in R, \forall v \in V^s \tag{3}$$

B. Task Transmission Model

In this section, we formulate the task transmission models of MEC server, cloud and computing vehicle, respectively.

First, the wireless V2I bandwidth of MEC server is competed among multiple vehicles for task transmission. If subtask r_v is offloaded to MEC server m, i.e., $a_{r_v}^m = 1$, let x_{r_v} denote the ratio of wireless V2I bandwidth allocated for r_v, then the summation of total allocation ratios should not exceed one, which is formulated as follows.

$$\sum_{\forall r \in R} \sum_{\forall v \in V_m^s} a_{r_v}^m x_{r_v} \le 1 \tag{4}$$

Given wireless V2I bandwidth b_m, the transmission time of offloading r_v from v to m is computed as follows.

$$tt_{r_v,m} = \frac{||D_{r_v}||}{x_{r_v} \cdot b_m \cdot log_2(1 + \frac{P_v \cdot g_{mv}}{N_0})} \tag{5}$$

Where P_v is the transmission power of vehicle v and g_{mv} is the channel gain between m and v.

Second, one-to-one V2V communication model is adopted for task transmission between vehicles. Only one subtask of neighboring crowdsourcing vehicles can be offloaded to each computing vehicle $v' \in V^c$, which is expressed as follows.

$$\sum_{\forall v \in V^s_{v'}, \forall r \in R} a^{v'}_{r_v} \leq 1 \tag{6}$$

Given wireless V2V bandwidth $b_{v'}$, if subtask r_v is offloaded to v', i.e., $a^{v'}_{r_v} = 1$, then the transmission time of offloading r_v from vehicle v to v' is computed as follows.

$$tt_{r_v,v'} = \frac{||D_{r_v}||}{b_{v'} \cdot log_2(1 + \frac{P_v g_{vv'}}{N_0})} \tag{7}$$

Third, task offloading from vehicles to cloud is transmitted via cellular wireless interface. Due to the wide deployment of BS in urban area, the vehicle is assumed to be always connected to the cloud via cellular interface. Given the cellular bandwidth b_c, then the transmission time of offloading r_v from v to cloud is computed as follows.

$$tt_{r_v,c} = \frac{||D_{r_v}||}{b_c \cdot log_2(1 + \frac{P_v \cdot g_{vc}}{N_0})} \tag{8}$$

Accordingly, the transmission cost is described as follows. It is a common assumption that the cost of V2I and V2V communication is neglected. Further, let ω_t denote the unit cost of using cellular interface, then the transmission cost of offloading r_v to the cloud is computed as follows.

$$tc_{r_v,c} = \omega_t \cdot ||D_{r_v}|| \tag{9}$$

C. Task computation model

In this section, we formulate the task computation models of MEC server, computing vehicles and cloud, respectively. In particular, the computation time of processing task, as well as the renting cost of computation resources, are introduced.

First, the computation resource of MEC server is competed among multiple pending subtasks. Given the computation resource of each processor owned by each MEC server m, denoted by f_m, let t_m denote the number of processors per MEC server and y_{r_v} denote the ratio of computation resources allocated for r_v, then the computation time of completing subtask r_v is formulated as follows.

$$pt_{r_v,m} = \frac{cr_{r_v}}{y_{r_v} \cdot f_m} \tag{10}$$

Due to the mobility of vehicles, the summation of transmission time $tt_{r_v,m}$ and computation time $pt_{r_v,m}$ cannot exceed the maximum V2I connection time between v and m, denoted by L_{vm}, which is formulated as follows.

$$tt_{r_v,m} + pt_{r_v,m} \leq L_{vm} \tag{11}$$

Then, the summation of allocated computation resources cannot exceed the maximum computation capability, which is formulated as follows.

$$\sum_{\forall r \in R} \sum_{\forall v \in V^s_m} a^m_{r_v} y_{r_v} \leq t_m \tag{12}$$

Second, the computing vehicle can only process one subtask at each time. Therefore, given the computation resource of vehicle v', denoted by $f_{v'}$, if r_v is offloaded to v', the computation time of processing r_v, is formulated as follows.

$$pt_{r_v,v'} = \frac{cr_{r_v}}{f_{v'}} \tag{13}$$

Similar to Eq. (11), the summation of transmission time $tt_{r_v,v'}$ and computation time $pt_{r_v,v'}$ cannot exceed the maximum V2V connection time between v and v', denoted by $L_{vv'}$, which is expressed as follows.

$$tt_{r_v,v'} + pt_{r_v,v'} \leq L_{vv'} \tag{14}$$

Third, the cloud is assumed to own unlimited computation resources. Given the required computation resource of r_v, denoted by $f^c_{r_v}$, then the computation time of processing r_v by the cloud is formulated as follows.

$$pt_{r_v,c} = \frac{cr_{r_v}}{f^c_{r_v}} \tag{15}$$

Accordingly, the cost of renting computation resources is described as follows. Let ω_m, ω_v and ω_c denote the unit cost of renting computation resources from the MEC, computing vehicle and the cloud, respectively, then the computation cost of r_v by the corresponding server is formulated in Eq. (16).

$$pc_{r_v,m} = \omega_m \cdot y_{r_v} \cdot f_m \tag{16}$$

$$pc_{r_v,v'} = \omega_v \cdot f_{v'} \tag{17}$$

$$pc_{r_v,c} = \omega_c \cdot f^c_{r_v} \tag{18}$$

Where $\omega_c > \omega_m > \omega_v$.

D. Problem Definition

Based on task transmission and computation model, the service time and cost of each subtask r_v, denoted by st_{r_v} and sc_{r_v}, are formulated as follows, respectively.

$$st_{r_v} = \sum_{\forall l \in N_v} a^l_{r_v}(tt_{r_v,l} + pt_{r_v,l}) \tag{19}$$

$$sc_{r_v} = a^c_{r_v} \cdot tc_{r_v,c} + \sum_{\forall l \in N_v} a^l_{r_v} \cdot pc_{r_v,l} \tag{20}$$

Then, the service time of a task r is defined as the average service time of all the subtasks $r_v \in r$, formulated as follows.

$$st_r = \sum_{\forall r_v \in r} \frac{st_{r_v}}{||r||}, \forall r \in R \tag{21}$$

Similarly, the service cost of a task r is defined as the average service cost of all the subtasks $r_v \in R$, formulated as follows.

$$sc_r = \sum_{\forall r_v \in r} \frac{sc_{r_v}}{||r||}, \forall r \in R \tag{22}$$

On this basis, two objectives are defined as follows.

Definition 3.1: Average service time (AST), it is defined as the summation of the service time of tasks divided by the total task number, which effectively evaluates system efficiency.

$$f_1 = \sum_{\forall r \in R} \frac{st_r}{||R||} = \sum_{\forall r \in R} \sum_{\forall r_v \in r} \frac{st_{r_v}}{||r|| \cdot ||R||} \tag{23}$$

Definition 3.2: Average service cost (ASC): it is defined as the summation of the service cost of all tasks divided by the total task number, which evaluates the system overhead.

$$f_2 = \sum_{\forall r \in R} \frac{sc_r}{||R||} = \sum_{\forall r \in R} \sum_{\forall r_v \in r} \frac{sc_{r_v}}{||r|| \cdot ||R||} \tag{24}$$

To minimize the two objectives simultaneously, the objective function is defined as the weighted sum of AST and ASC, which is formulated as follows.

$$f = \eta_1 \cdot f_1 + \eta_2 \cdot f_2 = \sum_{\forall r \in R} \sum_{\forall r_v \in r} \frac{\eta_1 \cdot st_{r_v} + \eta_2 \cdot sc_{r_v}}{||r|| \cdot ||R||} \tag{25}$$

Where η_1 and η_2 represent the weights of AST and ASC and $\eta_1 + \eta_2 = 1$.

Finally, the formal definition of the DTO problem is formulated as an optimization model, which is shown follows.

$$\min_{\mathbf{A}, \mathbf{X}, \mathbf{Y}, \mathbf{F}} \quad f = \sum_{\forall r \in R} \frac{\eta_1 \cdot st_r + \eta_2 \cdot sc_r}{||R||} \tag{26}$$

$$\text{s.t.} \quad \sum_{\forall l \in N_v} a^l_{r_v} = 1, \forall r \in R, \forall v \in V^s \tag{26a}$$

$$\sum_{\forall r \in R} \sum_{\forall v \in V^s_m} a^m_{r_v} x_{r_v} \le 1, \forall m \in M \tag{26b}$$

$$\sum_{\forall r \in R} \sum_{\forall v \in V^s_m} a^m_{r_v} y_{r_v} \le t_m, \forall m \in M \tag{26c}$$

$$a^l_{r_v} tt_{r_v,l} \le L_{vl}, \forall l \in N_v \tag{26d}$$

$$\sum_{\forall r \in R} \sum_{\forall v \in V^s_{v'}} a^{v'}_{r_v} \le 1, \forall v' \in V^c_m \tag{26e}$$

$$a^l_{rv} \in \{0, 1\}, \forall l \in N_v, \forall r_v \in r, \forall r \in R \tag{26f}$$

$$f^c_{r_v} \in R^+, \forall r_v \in r, \forall r \in R \tag{26g}$$

It is observed that $\mathbf{A} = \{a^l_{r_v}\}$ are the set of offloading decision, where each element is an 0-1 integer variable. $\mathbf{X} = \{x_{r_v}\}$, $\mathbf{Y} = \{y_{r_v}\}$ and $\mathbf{F} = \{f^c_{r_v}\}$, are continuous positive variables, which represent the allocation of V2I wireless bandwidth, MEC's and cloud's computation resources, respectively. Further, the constraints in Eqs. (26b) $\sim$ (26d) are non-linear. Therefore, the DTO problem is a mixed-integer non-linear programming model, which is typically an NP-hard problem and cannot be solved in polynomial time.

IV. Algorithm Design

In this section, we derive the solution of the DTO in two stages: task offloading and resource allocation. We first design an asynchronous deep Q-learning algorithm at each MEC server side for task offloading. Then, we derive the theoretical optimal solution of resource allocation based on decomposition method and convex theory.

A. Asynchronous Deep Q-Learning for Task Offloading

The basic idea of asynchronous deep q-learning (ADQN) is to achieve fast convergence by synthesizing the advantages of both asynchronous advantage actor-critic (A3C) [21] and deep Q-learning (DQN). As shown in Fig. 2, each agent has a local DQN model and a global model resides in the cloud. The basic elements of reinforcement learning in the DQN model are designed as follows.

- **System State**: due to the limited V2V range, it is assumed that at most n computing vehicles are available to each crowdsourcing vehicle. Then, the system state of current pending subtask r_v at time t is defined as a multi-dimensional vector, which is formulated as follows.

$$\begin{aligned}\mathbf{s}_{r_v}(t) = & [D_{r_v}, cr_{r_v}, D_{total}, D_{load}, b_m, f_m, b_{v'_1}, f_{v'_1}, \cdots, \\ & b_{v'_n}, f_{v'_n}]\end{aligned} \tag{27}$$

  Where D_{r_v} and cr_{r_v} represent data size and required computation resource of r_v. D_{total} represents the total data size of pending subtasks before scheduling and D_{load} represents the workload already offloaded to m.
- **Action Space**: it is defined as the set of candidate computation servers available to pending task r_v. Specifically, one-hot encoding is adopted to represent the action, which is represented as a n+2-dimensional binary vector, as shown in Eq. (28).

$$\mathbf{u}_{r_v}(t) = [a^m_{r_v}, a^c_{r_v}, a^{v'_1}_{r_v}, a^{v'_2}_{r_v}, \cdots, a^{v'_n}_{r_v}] \tag{28}$$

- **Reward Function**: the basic principle of reward function is that the lower service time and cost bring with the action, the higher reward is granted. If the subtask can be successfully completed within the connection time, then the reward is defined as the product of the reciprocal of the weight sum of service time and cost of subtask and a constant M_1. Otherwise, the reward is defined as a negative value, which represents a punishment, denoted by $-M_2$. The reward function is formulated as follows.

$$\mathbf{r}_{r_v}(t) = \begin{cases} \frac{M_1}{\eta_1 st_{r_v} + \eta_2 sc_{r_v}}, & st_{r_v,l} - L_{v,l} < 0 \\ -M_2, & otherwise \end{cases} \tag{29}$$

  Where M_1 and M_2 are two predefined positive constants and l represents the selected server. It is noted that the reward can only be obtained after all the pending tasks are completed.

Due to high dimension and continuity of system state and action space, the size of traditional Q-table will be too large, which results in dimension curse and slow convergence. To overcome this issue, the DQN model utilizes deep neural network to approximate Q-table instead, represented as follows.

$$Q(s, a, \theta) \approx Q'(s, a) \tag{30}$$

Where $Q(s, a, \theta)$ is the approximated Q-value of a given state-action pair by neural networks, θ is the parameter and $Q'(s, a)$ is the real Q-value of a given state-action pair.

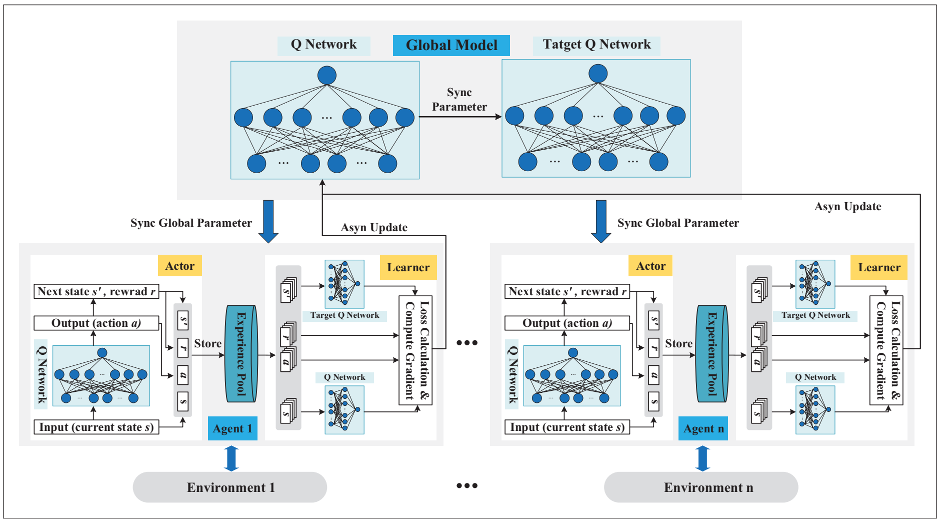


Fig. 2: The Diagram of Asynchronous Deep Q Learning Algorithm

According to Fig.2, local DQN model at each agent has two Q networks and experience pool. One current Q network, denoted by Q, is used for selecting action under a system state. The target Q network, denoted by Q' is used to calculate the target Q-value. The experience pool can store the tuples of state, action, reward and next state, denoted by (s, a, r, s'), which is used for repeatedly training. For a pair of s and a, the loss function of DQN is computed as below.

$$L(\theta_m) = E[(Q_{target} - Q(s, a, \theta_m))^2] \tag{31}$$

Where $Q(s, a, \theta_m)$ is the Q-value of local agent m. Q network uses gradient descent algorithms to update its own parameters to approach the target Q-value, where the parameter θ is updated as $\theta_m = \theta_m - \beta\nabla_{\theta_m} L$. β controls update space. In particular, the parameters of target Q network do not need to be updated frequently. It is copied from the parameters of Q network periodically, that is, delayed update, which can reduce the correlation between target Q network and Q network.

According to Fig. 2, based on the framework of A3C, the procedure of ADQN is described as follows. First, for each agent, given a list of pending subtasks, the action of task offloading is determined in an iterative way based on its local Q network. The generated experience, denoted by (s, a, r, s'), is stored in experience pool. Second, each agent extracts a batch of tuples from experience pool and computes the loss function based on Eq. (31), as well as its gradient. Third, the local gradient information of each agent is asynchronously uploaded to the global model at the cloud via wired connection. Fourth, once receiving the gradient information, the global Q network is immediately updated and its network parameters are shared to each local agent, i.e., $\theta_m = \theta_g, \forall m \in M$. Fifth, when the update number of Q network achieves a predefined threshold, the latest θ_g of Q network is copied to θ'_g of Q target network, i.e. $\theta'_g = \theta_g$, and then shared to each local agent, i.e., $\theta'_m = \theta'_g, \forall m \in M$. It should be noted that the first to third step can be performed at each agent simultaneously.

B. Optimal Resource Allocation based on Convex optimization

Based on the ADQN, the value of $\mathbf{A}$ can be computed in advance. Thus, the optimization model of Eq. (26) is only related to the resource allocation, which is rewritten as follows.

$$\min_{\mathbf{X},\mathbf{Y},\mathbf{F}} \quad f = \sum_{\forall r \in R} \frac{\eta_1 \cdot st_r + \eta_2 \cdot sc_r}{||R||} \tag{32}$$

$$\text{s.t.} \quad \sum_{\forall r \in R} \sum_{\forall v \in V_m^s} a_{r_v}^m x_{r_v} \leq 1, \forall m \in M \tag{32a}$$

$$\sum_{\forall r \in R} \sum_{\forall v \in V_m^s} a_{r_v}^m y_{r_v} \leq t_m, \forall m \in M \tag{32b}$$

$$f_{r_v}^c \in R^+, \forall r_v \in r, \forall r \in R \tag{32c}$$

It is observed that the variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{F}$ in Eq. (32) are independent with each other. In particular, the three constraints of Eqs.(32a)$\sim$(32c) are separable since the variables are not overlapped. Thus, the optimization model can be decomposed into three submodels, which are formulated as follow.

1) Computation Resource Allocation Of Cloud: the first submodel with respect to $\mathbf{F}$ concerns on computation resource allocation of cloud, which is formulated as follows.

$$\min_{\mathbf{f_{r_v}^c}} \quad g_1 = \eta_1 \cdot \frac{cr_{r_v}}{f_{r_v}^c} + \eta_2 \cdot \omega_c \cdot f_{r_v}^c \tag{33}$$

$$\text{s.t.} \quad f_{r_v}^c \in R^+, \forall r_v \in r, \forall r \in R \tag{33a}$$

The optimal solution is achieved by finding the solution with the gradient of g_1 equals zero, i.e., $\nabla g_1 = 0$. The optimal computation resource allocated for each r_v offloaded to the cloud is computed as follows.

$$f_{r_v}^{c*} = \sqrt{\frac{\eta_1 cr_{r_v}}{\eta_2 \omega_c}}, \forall r_v \in r, \forall r \in R \tag{34}$$

2) Wireless Bandwidth Resource Allocation: the second submodel with respect to the variables $\mathbf{X}$ concerns on wireless V2I bandwidth allocation, which is formulated as follows.

$$\min_{\mathbf{X}} \quad g_2 = \sum_{\forall r \in R} \sum_{\forall r_v \in r} \frac{\eta_1 a^m_{r_v} tt_{r_v,m}}{||r|| \cdot ||R||} \tag{35}$$

$$\text{s.t.} \quad \sum_{\forall r \in R} \sum_{\forall v \in V^s_m} a^m_{r_v} x_{r_v} \leq 1, \forall m \in M \tag{35a}$$

It is observed that the variables related to the MEC servers are independent of each other, the submodel Eq.(35) can be further divided into multiple simple models, where each one is only related with an MEC server m, shown as below.

$$\min_{\mathbf{X_m}} \quad g^m_2 = \sum_{\forall r_v \in R^m} \frac{\eta_1 a^m_{r_v} tt_{r_v,m}}{||r|| \cdot ||R||} \tag{36}$$

$$\text{s.t.} \quad \sum_{\forall r_v \in R^m} a^m_{r_v} x_{r_v} \leq 1 \tag{36a}$$

Where R^m represents all subtasks within the coverage of MEC server m and $\mathbf{X_m}$ represents the variables in $\mathbf{X}$ associated with MEC server m. Obviously, the objective in Eq. (36) is convex and the constraint in Eq. (36a) is linear. Thus, the model of Eq.(36) is a convex optimization problem. Based on the KKT condition [18], we can get the following formulas:

$$\begin{aligned} \nabla_{\mathbf{X_m}} g^m_2 + \lambda_m \nabla_{\mathbf{X_m}} \Big(\sum_{\forall r_v \in R^m} a^m_{r_v} x_{r_v} - 1 \Big) = 0, \\ \lambda_m \Big(\sum_{\forall r_v \in R^m} a^m_{r_v} x_{r_v} - 1 \Big) = 0, \\ \lambda_m \geq 0 \end{aligned} \tag{37}$$

By solving the set of equations, the optimal solution of wireless bandwidth allocation for subtask r_v can be obtained as follow:

$$\begin{aligned} x^*_{r_v} &= \frac{a^m_{r_v} \sqrt{\xi_{r_v}}}{\sum\limits_{\forall r_v \in R^m} a^m_{r_v} \sqrt{\xi_{r_v}}}, \forall r_v \in R^m \\ \xi_{r_v} &= \frac{\eta_1 \cdot D_{r_v}}{b_m \cdot ||r|| \cdot log_2(1 + \frac{P_v g_{mv}}{N_0})}, \forall r_v \in R^m \end{aligned} \tag{38}$$

3) Computation Resource Allocation Of MEC Server: the third submodel with respect to variables $\mathbf{Y}$ concerns on computation allocation of MEC Server, formulated as follows.

$$\min_{\mathbf{Y}} \quad g_3 = \sum_{\forall r \in R} \sum_{\forall r_v \in r} \frac{\eta_1 a^m_{r_v} pt_{r_v,m} + \eta_2 a^m_{r_v} pc_{r_v,m}}{||r|| \cdot ||R||} \tag{39}$$

$$\text{s.t.} \quad \sum_{\forall r \in R} \sum_{\forall v \in V^s_m} a^m_{r_v} y_{r_v} \leq t_m, \forall m \in M \tag{39a}$$

Similar to the submodel of Eq. (35), the submodel of Eq.(39) can also be divided into multiple simple models, where each one is only associated with an MEC server m, shown as below.

$$\min_{\mathbf{Y_m}} \quad g^m_3 = \sum_{\forall r_v \in R^m} \frac{\eta_1 a^m_{r_v} pt_{r_v,m} + \eta_2 a^m_{r_v} pc_{r_v,m}}{||r|| \cdot ||R||} \tag{40}$$

$$\text{s.t.} \quad \sum_{\forall r_v \in R^m} a^m_{r_v} y_{r_v} \leq t_m \tag{40a}$$

Where $\mathbf{Y_m} = \{y_{r_v}\}, \forall r_v \in R^m$ represents variables in $\mathbf{Y}$ associated with MEC server m. Based on KKT condition, we can obtain two alternative solutions. For the first case, where dual variable $\lambda_m = 0$, the solution is derived as follows.

$$\begin{aligned} y^*_{r_v} &= a^m_{r_v} \sqrt{\frac{\theta_{r_v}}{\gamma_{r_v}}}, \forall r_v \in R^m \\ \theta_{r_v} &= \frac{\eta_1 \cdot cr_{r_v}}{f_m \cdot ||r|| \cdot ||R||}, \forall r_v \in R^m \\ \gamma_{r_v} &= \frac{\eta_2 \cdot \omega_m \cdot f_m}{||r|| \cdot ||R||}, \forall r_v \in R^m \end{aligned} \tag{41}$$

For the second case, where dual variable $\lambda_m \neq 0$, the solution is derived as follows.

$$\begin{aligned} y^*_{r_v} &= a^m_{r_v} \sqrt{\frac{\theta_{r_v}}{\gamma_{r_v} + \lambda_m}}, \forall r_v \in R^m \\ \sum_{\forall r_v \in R^m} & a^m_{r_v} \sqrt{\frac{\theta_{r_v}}{\gamma_{r_v} + \lambda_m}} - t_m = 0, \end{aligned} \tag{42}$$

Where the variable λ_m can be efficient achieved by solving the second subequation of Eq. (42) via binary search. Obviously, the optimal solution of the third submodel is the one of two solutions in Eq. (41) and (42) with higher objective function.

It is noted that the solution of resource allocation, consists of Eq.(34), (38), (41) and (42), can be independently implemented at the MEC server side with local knowledge.

V. Performance evaluation

A. Default setting

In this section, we implement the simulation model presented in Section II. Specifically, the road map is extracted from a $4km \times 4km$ area of Tianfu new district in Chengdu, China. The vehicular traces are simulated by an open-source traffic simulator called SUMO [22]. There exist five MEC servers distributed among road network and five types of data-driven tasks. Each task consists of five sub-tasks, which indicates that the associated data set consists of five part and each part is cached by a crowdsourcing vehicle. It is assumed that each crowdsourcing vehicle has at most three neighboring computing vehicles for task offloading. Data size and required computing resources of a task are randomly selected from the intervals [45, 75] MB and [30, 50] G CPU cycles. In addition, the wireless bandwidth of vehicle and MEC server are set to 30 and 150 MHz. The computation capacity of computing vehicle and MEC server are set to 10 and 60 G CPU cycles/s. The price of renting one unit computation resource of computing vehicle, MEC server and cloud is set to 0.05, 0.2 and 1 \$/G cycles, respectively. The price of uploading data via cellular interface is set to $0.5 \times 10^{-4} \$/MB$. In addition, the transmission power of vehicles P_v, guassian channel noise N_0 and channel gain g_{mv} are set to $150mW$, $10^{-6}mW$ and $5DB$, respectively. The weights η_1 and η_2 are set to 0.5. The setting of parameters are referred to these literature [17, 23–28].

For algorithm implementation, the network architecture and hyperparameters of ADQN are described as follows. The Q network consists of four hidden layers, where the numbers of neurons are set to 32, 16, 8 and 4, respectively. The learning rate and discount factor is set to 0.001 and 0.9, respectively. The size of experience pool and batch is set to 100 and 10, respectively. Since no existing algorithms are suitable for the DTO problem, we implement two competitive task algorithms of task offloading for performance comparison, called deep Q-Learning (DQN) [29] and random offload scheduling (ROS), respectively. The detail is described as follows.

- **DQN**: which is implemented at each MEC server for making task offloading decision. In this way, each agent has its own independent neural network, which is trained independently with its local experience.
- **ROS**: which randomly chooses one of the servers for task offloading, including cloud, dwelling MEC server and neighboring computing vehicles.

All the task offloading algorithms adopt the resource allocation derived in Section IV-B. Due to its optimality, we do not introduce other resource allocation algorithms for comparison.

For performance evaluation, we collected the following statistics for each subtask r_v at each scheduling period t: the time $st^t_{r_v}$ and the cost $sc^t_{r_v}$ of completing subtask r_v, the reward $r^t_{r_v}$ of completing subtask r_v, and offloaded server $a^{lt}_{r_v}$ of r_v. Besides ASC, AST and objective value (OV) defined in Eqs. (23)∼(25), we define extra two metrics for analysis.

- **Average cumulative reward (ACR)**: it is defined as the cumulative rewards divided by the number of scheduling periods t, which is calculated as follows.

$$ACR(t) = \frac{1}{t}\sum_{i=1}^{t}\sum_{\forall r\in R}\sum_{\forall r_v\in r}\frac{r^i_{r_v}}{||r||\cdot||R||} \tag{43}$$

  The cure of ACR indicates the converge speed and the performance of the algorithm.

- **Offloading proportion (OP)**: it is defined as the number of subtasks offloaded to each type of computation server divided by the total number of subtasks.

$$OP(l) = \frac{1}{T}\sum_{t=1}^{T}\sum_{\forall r\in R}\sum_{\forall r_v\in r}\frac{a^{lt}_{r_v}}{||r||\cdot||R||} \tag{44}$$

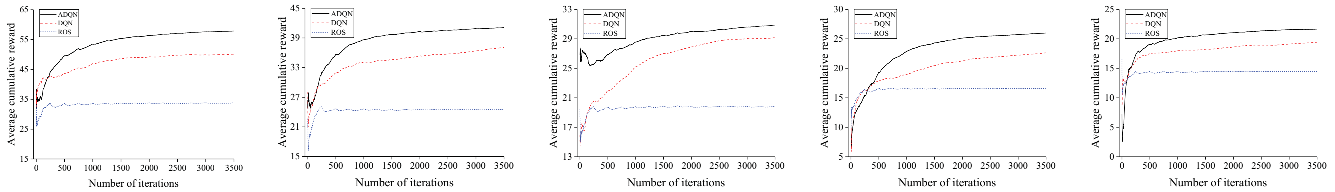
  Where l represents the type of computation server. OP reflects the behavior of task offloading algorithms.

1) Effect of computation resource requirement: Fig. 4a compares the OV of three algorithms under different computation resource requirements. As shown, the OV of all algorithms increases when computation resource requirement increases. The reasons behind are introduced as follows. First, higher computation resource requirement means longer time for computing each task, which results in higher value of AST. It is verified that the AST of three algorithms increases, as shown in Fig. 4b. Second, crowdsourcing vehicles have to rent more computation resources for computing tasks by paying more renting cost. As shown in Fig. 4c, the ASC of three algorithms increases gradually. Third, the tasks are preferred to be offloaded to the cloud since the increasing system workload may exceed the limited computation capacities of computing vehicles and MEC servers. It is verified that the proportion of tasks offloaded to cloud increases as shown in Fig. 4d, which can also increase the ASC.

In addition, it is noted that the ADQN achieves the lowest value of OV among three algorithms according to Fig. 4a, which can be explained as follows. First, the ADQN algorithm explores the environment more efficient by asynchronously sharing the local knowledge for global DQN model update and better exploits the resource to make appropriate offloading choice based on periodically updated DQN model. Fig. 3 compares the ACR of three algorithms under different computation resource requirements. It is clear that the ADQN algorithm can always converge to a higher level than the other two algorithms after a small number of iterations. Further, as shown in Fig. 4d, the proportion of task offloading to the cloud by ADQN is the lowest among three algorithms, which indicates that the ADQN can coordinate the heterogeneous resources well for completing tasks without over-relying on cloud even in heavier workload. Therefore, this set of simulation results demonstrates the scalability of the proposed algorithm.

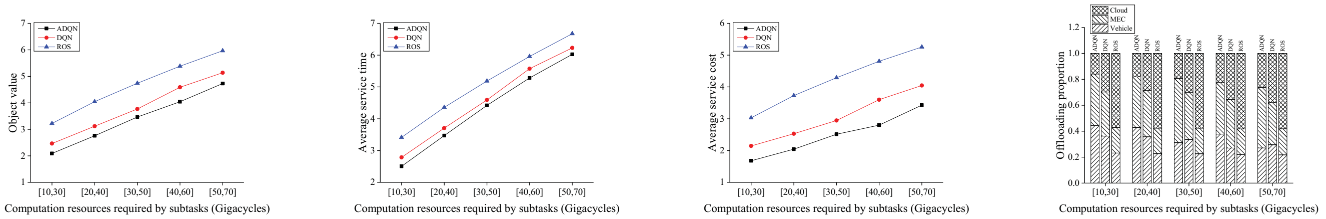
2) Effect of computation capacity of MEC server: Fig. 5a compares the OV of three algorithms under different computation capacities of MEC servers. As shown, the performance of all algorithms gets better when the computation capacity increases. It is observed that the OV of three algorithms decreases much more when the computation capacity increases form 20 GHZ to 40 GHZ than other cases. The reason behind is that the computing capacity of MEC server is too low at first and only handle limited number of tasks with its local processor, which enforces the remaining pending tasks to choose the cloud for task processing. However, when the computation capacity of MEC server increases to more than 40GHZ , the renting price of MEC server will also increase and slow down the reduction of OV, which will be further validated by the following simulation results.

Fig. 5b compares the AST of three algorithms. As shown, the AST of DQN and ADQN decreases with the increasing of computation capacity of MEC server. However, the AST of ROS maintains at a high level. It is because that ROS does not change the probability of subtask offloaded to MEC server in all cases and the resource allocation of MEC server remains unchanged when computation capacities of MEC server exceeds a certain level according to Eq.(41). Further, Fig. 5c compares the ASC of all algorithms. It is observed that the ASC of ADQN and DQN decreases at first and then increases. It is because that the number of subtasks offloaded to the cloud will drastically decrease when computation capacity of MEC server increases from 20 GHZ to 60 GHZ. However, since the renting cost is proportional to computation capability, the increase of renting cost of MEC server finally exceeds the reduction of renting cost of cloud when computation capability of MEC server keeps increasing, which results in the increasing of ASC. This explanation is strengthened by Fig. 5d, which shows the OP of three algorithms. The proportion of the cloud of ADQN and DQN decreases dramatically and



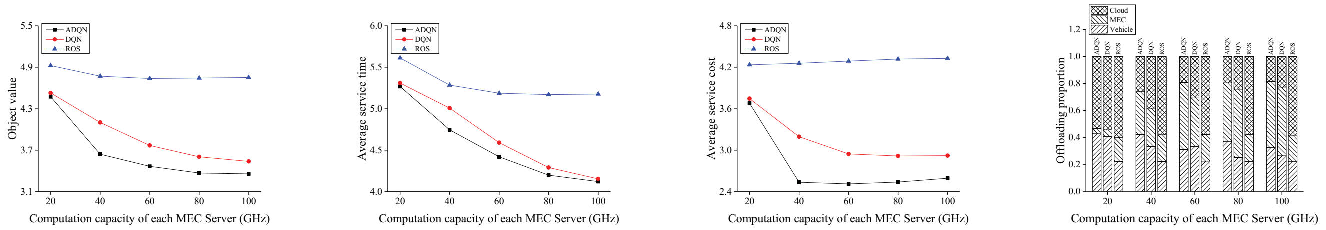
(a) cr_{rv} ∼[10,30] (b) cr_{rv} ∼[20,40] (c) cr_{rv} ∼[30,50] (d) cr_{rv} ∼[40,60] (e) cr_{rv} ∼[50,70]

Fig. 3: The ACR of three algorithms under different computation resource requirements



(a) OV (b) AST (c) ASC (d) OP

Fig. 4: The performance of three algorithms under different computation resource requirements



(a) OV (b) AST (c) ASC (d) OP

Fig. 5: The performance of three algorithms under different computation capacities of MEC Servers

that of ROS remains the same across all the cases. According to Figs. 5a∼5c, the ADQN still achieves the lowest OV, ASC and AST in all cases, which demonstrates the adaptiveness of the ADQN against different computation capacities.

VI. Conclusion

In this work, we investigate a novel service scenario of data-driven task offloading in MEC-empowered vehicular networks, where heterogeneous computation servers, such as computing vehicles, MEC servers and cloud, are expected to cooperate with each other for processing subtasks offloaded by multiple crowdsourcing vehicles. On this basis, we formulate the DTO by characterizing the properties of data-driven tasks, task transmission and computation procedure of heterogeneous servers as well as the corresponding renting cost, which aims at minimizing AST and ASC, simultaneously. Due to the high complexity, we divide the solution of the DTO into two stages: task offloading and resource allocation. First, we design the ADQN algorithm for determining task offloading in a distributed way by combing the framework of A3C and DQN, which achieves fast convergence by enabling parallel training of lcoal DQN model at each MEC server and asynchronously update of global model. Second, we decompose the remaining resource allocation problem into three parts: computation resource allocation of cloud, wireless bandwidth allocation of MEC server, computation resource allocation of MEC server, and derive the optimal analytic form for each part based on KKT condition. Finally, we build the simulation model by integrating real-world map, SUMO and scheduling module. The extensive simulation results show the superiority of the proposed algorithm under a wide variety of circumstances.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61802319, 61772436; China Postdoctoral Science Foundation funded project (No. 2019M660245, 2020T130547); Foundation for Department of Transportation of Henan Province of China (2019J-2-2); Sichuan Science and Technology Program (Grant No. 2020YJ0207, No. 2020YJ0272).

References

[1] W. Chen, Z. Su, Q. Xu, T. H. Luan, and R. Li, "VFC-based cooperative UAV computation task offloading for post-disaster rescue," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 228–236.

[2] K. Liu, K. Xiao, P. Dai, V. Lee, S. Guo, and J. Cao, "Fog computing empowered data dissemination in software defined heterogeneous VANETs," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[3] P. Dai, K. Liu, L. Feng, H. Zhang, V. C. S. Lee, S. H. Son, and X. Wu, "Temporal information services in large-scale vehicular networks through evolutionary multi-objective optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 218–231, 2019.

[4] Z. Jiang, S. Zhou, X. Guo, and Z. Niu, "Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis, and implications on road traffic," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 93–107, 2018.

[5] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Hybrid computation offloading in fog and cloud networks with non-orthogonal multiple access," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 154–159.

[6] X. Huang, R. Yu, J. Kang, Y. He, and Y. Zhang, "Exploring mobile edge computing for 5G-enabled software defined vehicular networks," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 55–63, 2017.

[7] S. Dai, M. Li Wang, Z. Gao, L. Huang, X. Du, and M. Guizani, "An adaptive computation offloading mechanism for mobile health applications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 998–1007, 2020.

[8] P. Dai, Z. Hang, K. Liu, X. Wu, H. Xing, Z. Yu, and V. C. S. Lee, "Multi-armed bandit learning for computation-intensive services in mec-empowered vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7821–7834, 2020.

[9] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[10] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for mec in heterogeneous vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7916–7929, 2020.

[11] P. Dai, K. Hu, X. Wu, H. Xing, F. Teng, and Z. Yu, "A probabilistic approach for cooperative computation offloading in mec-assisted vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2020.

[12] H. Liang, X. Zhang, J. Zhang, Q. Li, S. Zhou, and L. Zhao, "A novel adaptive resource allocation model based on SMDP and reinforcement learning algorithm in vehicular cloud system," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 018–10 029, 2019.

[13] Z. Hao, Y. Sun, Q. Li, and Y. Zhang, "Delay-energy efficient computation offloading and resources allocation in heterogeneous network," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[14] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, Jan 2018.

[15] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[16] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2062–2075, 2020.

[17] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[18] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[19] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[20] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[22] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.

[23] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[24] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1079–1092, 2019.

[25] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[26] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, 2019.

[27] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, 2019.

[28] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, B. Hu, and Y. Li, "When deep reinforcement learning meets 5G-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352–1361, 2020.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *NATURE*, vol. 518, no. 7540, pp. 529–533, FEB 26 2015.