





# Adaptive Computing Scheduling for Edge-Assisted Autonomous Driving

Mushu Li , *Student Member, IEEE*, Jie Gao , *Member, IEEE*, Lian Zhao , *Senior Member, IEEE*,  
and Xuemin Shen , *Fellow, IEEE*

**Abstract**—This paper investigates computing resource scheduling for real-time applications in autonomous driving, such as localization and obstacle avoidance. In our considered scenario, autonomous vehicles periodically sense the environment, offload sensor data to an edge server for processing, and receive computing results from the server. Due to mobility and computing latency, a vehicle travels some distance in the duration between the instant of offloading its sensor data and the instant of receiving the computing result. Our objective is finding a scheduling scheme for the edge server to minimize the above traveled distance of vehicles. The approach is to determine the processing order according to individual vehicle mobility and computing capability of the edge server. We formulate a restless multi-arm bandit (RMAB) problem, design a Whittle index based stochastic scheduling scheme, and determine the index using a deep reinforcement learning (DRL) method. The proposed scheduling scheme avoids the time-consuming policy exploration common in DRL scheduling approaches and makes effectual decisions with low complexity. Extensive simulation results demonstrate that the proposed indexed-based scheme can deliver computing results to the vehicles promptly while adapting to time-variant vehicle mobility.

**Index Terms**—Autonomous vehicles, computing scheduling, mobile edge computing, restless multi-armed bandit.

## I. INTRODUCTION

**S**AFETY is one of the main focuses in autonomous driving related industries. Various techniques have been adopted to improve safety, e.g., embedding advanced sensors in vehicles and developing precise perception using the sensor data. Such safety measures usually yield a large amount of sensor data to be processed with low latency [1], [2], which could be too demanding for local in-vehicle processing given the usually limited on-board computing capability. To facilitate the safety measures, mobile edge computing (MEC) has emerged as an approach

to provide additional computing power with low processing delay [3], [4]. In such an approach, vehicles can offload their sensor data to a proximal edge server, such as a base station or a roadside unit, for fast data processing.

Despite the potential of MEC in enabling low-latency computing offloading for autonomous driving, many challenges exist for MEC to support real-time safety-related computing services. In most safety applications, such as localization and obstacle avoidance, vehicles may need to offload sensor data to the edge server and require computing results [5], [6]. Because of the mobility and computing delay, a vehicle would have traveled some distance between the instant of offloading sensor data to the edge server and the instant of receiving the computing result from the edge server. Evidently, it is important to reduce the above traveled distance as much as possible, considering that the computing result may involve the vehicle's position at the instant of sensor data collection. Otherwise, the computing result may no longer be accurate for real-time applications. Consider a vehicle traveling at 50 km/h as an example. If the vehicle receives the computing result 2 seconds after sensor data offloading, the gap in the distance between the real-time location and the location at the instant of offloading would be 28 m. This gap is larger than the reaction distance at 50 km/h, which is 21 m [7]. The challenge in reducing the traveled distance is that the edge server may have limited computation units, which must be shared among all vehicles in its proximity. As a result, the delay from the sensor data offloading to the result delivery may increase with the number of vehicles, and so does the traveled distance. Without proper scheduling, the delay can become excessive [8].

There are extensive existing works on computing resource scheduling in MEC [9]–[11]. Generally, the main objective of the proposed schemes is to minimize the computing latency. However, most of the works focus on myopic computing service scheduling, which considers the existing computing load at the edge server but not future computing service demands. As vehicles may collect sensor data and request computing service from time to time, developing a long-term proactive scheduling scheme is important. An effectual scheduling scheme should provide timely and frequent result delivery for vehicles so that the computing results can reflect their real-time status, e.g., position, as much as possible.

Another important requirement for effectual scheduling is the capability to adapt to the dynamics of vehicular mobility.

Manuscript received August 1, 2020; revised December 18, 2020; accepted February 16, 2021. Date of publication March 1, 2021; date of current version July 8, 2021. This work was supported in part by the Natural Sciences, and Engineering Research Council of Canada under Grant STPGP-493787. The review of this article was coordinated by the Guest Editors of the Special Section on Vehicular Networks in the Era of 6G: End-Edge-Cloud Orchestrated Intelligence. (*Corresponding author: Lian Zhao.*)

Mushu Li and Xuemin Shen are with the Department of Electrical, and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: m475li@uwaterloo.ca; sshen@uwaterloo.ca).

Jie Gao is with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI 53233 USA (e-mail: j.gao@marquette.edu).

Lian Zhao is with the Department of Electrical Computer and Biomedical Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: l5zhao@ryerson.ca).

Digital Object Identifier 10.1109/TVT.2021.3062653

Different vehicles may travel at different speeds, which can result in different service delay tolerance for real-time applications. Consider obstacle avoidance as an example. The edge server should schedule high-speed vehicles with high priority to minimize their traveled distance, and low-speed vehicles may have low priority. In addition, the speed of a vehicle may change over time, which may further complicate long-term scheduling as such change is not known in advance. There are existing works on resource allocation in MEC considering the mobility dynamics, e.g., the offloading bandwidth assignment in [12], [13], computing resource allocation in [14], [15], and cooperative computing in [9], [16]. However, mobility has a more significant impact on the performance in our considered problem since the traveled distance, which we aim to minimize, is dependent on vehicle mobility.

In this work, we design a computing resource scheduling scheme at the edge server for real-time applications in autonomous driving, considering the vehicle mobility dynamics. Vehicles periodically offload sensor data, referred to as observations, to the edge server for processing, while the edge server determines the processing order considering its computing capability and the unknown vehicle mobility dynamics. We define the age-of-result (AoR) of a vehicle as its traveled distance since the last data offloading before receiving the latest result delivery from an edge server. Our objective is to minimize the expected AoR of vehicles to deliver computing results timely. We formulate the long-term scheduling problem as a restless multi-arm bandit (RMAB) problem and propose a Whittle index-based scheduling scheme. Two offloading scenarios are investigated in the paper: *synchronous*, in which all computing requests arrive simultaneously, and *asynchronous*, in which the computing requests arrive arbitrarily.

The main contributions of this work are as follows:

- 1) We design a novel computing resource scheduling scheme for the edge server to support autonomous driving, targeting at minimizing the AoR while considering the vehicle mobility dynamics. The scheduling scheme can support a large number of computing tasks with low complexity.
- 2) We obtain the Whittle index of the formulated RMAB problem in closed form and prove the indexability of the scheduling problem. The index can reveal the value of scheduling each computing request and guide the computing policies in both the synchronous and the asynchronous scenarios.
- 3) We exploit the deep reinforcement learning (DRL) method to estimate the unknown mobility dynamics of vehicles in the future according to their mobility dynamics in the past. The learning process does not rely on the scheduling decisions and can be pre-trained either at the edge server or the vehicles.

The remainder of the paper is organized as follows. Section II introduces the related works. Section III describes the system model of the considered scenarios. Section IV introduces the proposed index-based scheduling scheme for the synchronous offloading scenario. The learning approach that matches the index policy is presented in Section V. Section VI introduces the scheduling scheme for the asynchronous scenario. Simulation

results are presented in Section VII. Section VIII concludes this article.

## II. RELATED WORKS

In the literature, MEC technology is adopted to support computation-intensive applications with low latency requirements for autonomous vehicles, such as localization [17], high-definition (HD) map generation [18], and on-board infotainment services [19], [20]. There are many works on multi-resource (e.g., spectrum and computing resources) allocation for computing offloading and scheduling [21]–[24]. Some of the existing works utilize machine learning techniques to assist decision making for edge computing in autonomous driving. Peng *et al.* developed a learning approach to allocate communication bandwidth and computing units to facilitate the computing offloading of autonomous vehicles [12]. Sun *et al.* proposed a multi-armed bandit based offloading strategy to adapt to a fast-varying network topology [15]. Li *et al.* studied a collaborative computing approach in vehicular networks and proposed a DRL technique to tackle a complex decision-making problem [9]. Computing resource scheduling is also investigated in [11], [25] to maximize the utilization of limited computing resources at edge servers. The above works mainly focus on supporting non-real-time computing applications. For the real-time safety-related computing applications, vehicles may require frequent computing offloading and timely delivery of computing results [5], [6]. In such a scenario, a customized computing scheduling scheme for autonomous driving is important yet has not been investigated, to the best of our knowledge.

To measure the timeliness of the received information, the age-of-information (AoI) is well investigated in various communication and computing scenarios. Extensive research works study the minimization of the AoI using techniques such as stochastic scheduling [26] and reinforcement learning [27]. In vehicular networks, the freshness of the information is critical [28]. AoI minimization is considered to support the real-time on-board services, such as vehicle-to-vehicle communications [29], [30] and HD map caching for automated driving [31]. Meanwhile, the computing result aging issue caused by processing delay has been considered in [32]–[35]. Li *et al.* introduced a general model for AoI minimization in edge computing [32]. Kuang *et al.* studied the AoI performance among different computing policies and provided closed-form expressions of AoI under stochastic processing delay distributions in MEC [33]. Chen *et al.* investigated the timeliness of computing result delivery in vehicular networks and minimized AoI by reinforcement learning [34]. Inspired by AoI, we define a new metric, *i.e.*, AoR, to measure the average traveled distance from the position where the last computing result was received. The most relevant works to this paper are [36] and [37]. Both works formulated an RMAB problem, utilized Whittle's Index policies to cope with the randomness in the network environment, and proactively scheduled the packets. However, unlike the AoI, the factors that impact AoR include both the elapse of time and, more importantly, vehicle mobility. The dynamics of vehicle mobility render the considered problem more challenging.

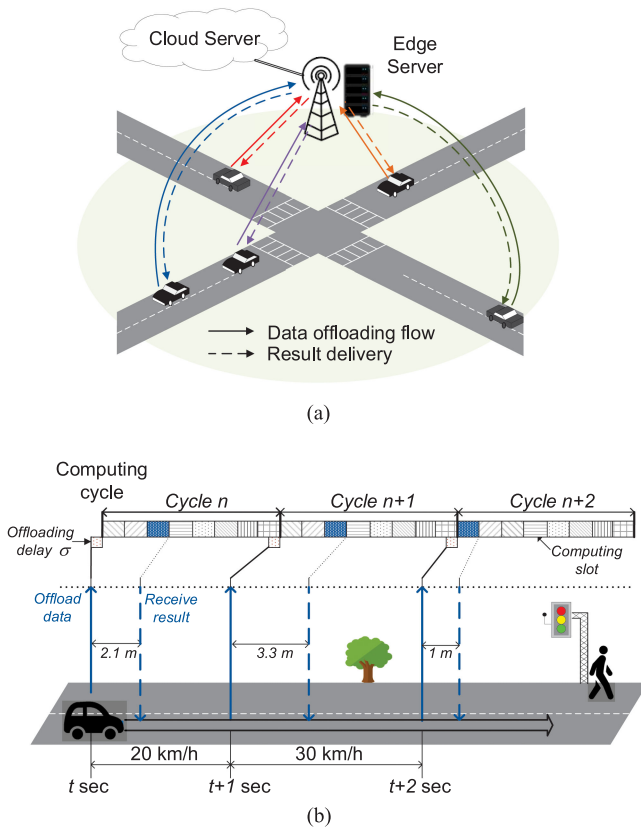


Fig. 1. System model. (a) Edge-assisted autonomous driving network. (b) An example on scheduling and result delivery on one processor at an edge server.

### III. SYSTEM MODEL

#### A. Network Model

Consider a vehicular network of autonomous driving shown in Fig. 1(a). Vehicles sense the surrounding environment periodically (e.g., for localization, obstacle detection, and object tracking), using sensors such as light detection and ranging and cameras. The sensor data is referred to as observations, which are offloaded to the closest edge node (e.g., roadside unit or base station) for processing. The sensing and offloading processes occur periodically for each vehicle, and each period is referred to as a sensing cycle. The processing of an observation offloaded to an edge node in each sensing cycle is referred to as a computing task. In the sequel, we focus on a particular edge server.

An edge server is located in the edge node and may have one or more processors. The server determines the execution order and processes the received computing tasks periodically at each processor, and each period is referred to as a computing cycle. We assume that each processor handles a specific type of computing tasks, which has the same processing time. For brevity, we focus on the scheduling of computing tasks in one processor, as the extension to multiple processors is straightforward.

The time length of a computing cycle is  $T$ , which is divided into  $K$  computing slots. The length of a slot is the processing time for a computing task, denoted by  $\omega$ . The length of a computing cycle depends on vehicles' AoR requirements and their sensing frequency. A shorter computing cycle can lead to

a lower AoR since the likelihood of scheduling a vehicle in a certain period can be increased; however, too short computing cycle raises the vehicles' sensing frequency, which may result in redundant task offloading and high communication resource consumption. The objective of scheduling scheme on the edge server is to allocate computing tasks to the slots in each computing cycle. The edge node is connected to a cloud server through a backhaul link. As a controller, the cloud server determines computing cycle length  $T$  and coordinates the computing resources among all edge nodes.

If a task from a particular vehicle and a particular sensing cycle is not scheduled until the next task of the vehicle is generated and offloaded to the edge server, it becomes outdated and discarded, and the server schedules the newly arrived task corresponding to a subsequent and up-to-date observation from the same vehicle. After processing the computing task, the edge server delivers the computing result back to the corresponding vehicle. Therefore, the computing schedule also affects the order of result delivery. Since we consider real-time applications and the cycle length is in seconds, the probability that the vehicle travels out of the communication range of the edge server is neglected. For the case that the vehicle travels out of the communication range during a computing cycle, multiple edge nodes can cooperatively deliver the result back to the vehicle, which has been discussed in our previous work [9]. We ignore the transmission delay on result delivery since the downlink transmission time is neglectable compared to the computing time [4], [12], [23].

Due to the mobility and computing latency, the traveled distance during offloading and edge computing is nonzero for any vehicle and any task. An example of computing scheduling and result delivery is shown in Fig. 1(b), where  $\sigma$  represents the offloading time of a task.<sup>1</sup> A vehicle offloads the computing tasks periodically (e.g., every second as shown in the figure). The speed of the vehicle remains constant within each sensing cycle but may vary in different cycles. Because of the changing speed of the vehicle, its traveled distance in two cycles can be different. This is true even if the vehicle is assigned to the same slot in two different cycles. As shown in the figure, the traveled distances of the vehicle in cycle  $n$  and  $n+1$  are different although the vehicle is assigned to the third slot of both the computing cycles. In the sequel, we focus on minimizing the expected traveled distance from the position implied by the newest received computing results, *i.e.*, minimizing AoR.

#### B. Computing Scheduling Scenarios

Based on the task arrival pattern of vehicles in each computing cycle, two computing scheduling scenarios are analyzed in this work:

- *Synchronous offloading*: In this scenario, all vehicles have the same sensing cycle and offload sensor data to the edge server at the beginning of a computing cycle. In addition, the computing cycle at the edge server is also the same in length as the sensing cycle. This is illustrated in Fig. 2(a).

<sup>1</sup>As all tasks handled by the same processor is of the same type, we assume that the offloading time is constant and the same for all vehicles.



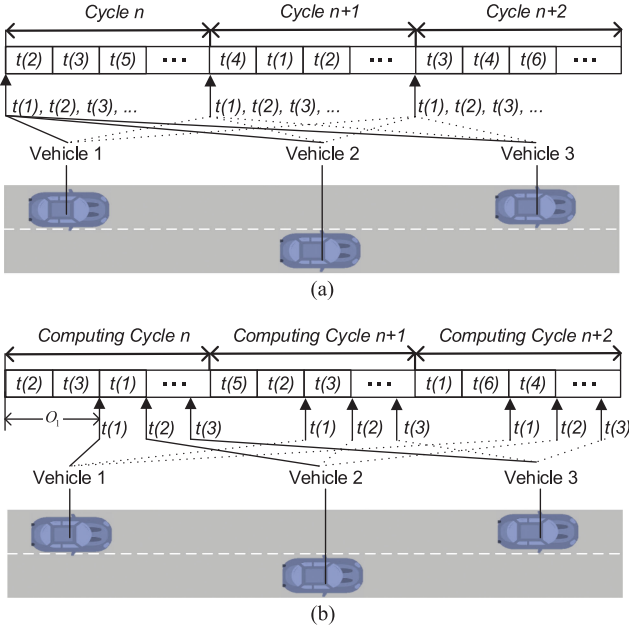


Fig. 2. An illustration of the scheduling scenarios, where  $t(i)$  represents the computing task from vehicle  $i$ . (a) Synchronous offloading. (b) Asynchronous offloading.

The edge server decides the processing order of the tasks received at the beginning of the cycle.

- *Asynchronous offloading*: In this scenario, vehicles offload their observations to the edge server at arbitrary instants. This is illustrated in Fig. 2(b). The time duration from the beginning of the computing cycle to the offloading time of vehicle  $i$  is denoted by  $O_i$ . For simplicity, we assume that a computing cycle at the edge server is the same in length as the sensing cycle in this scenario as well. However, the proposed scheme can be readily extended even if the assumption does not hold. Since the sensing and computing cycles are not aligned, a vehicle may have two tasks from adjacent sensing cycles scheduled in the same computing cycle, one offloaded in the previous computing cycle and the other offloaded in the current computing cycle.

In the remainder of the paper, we mainly focus on the synchronous offloading. In Section VI, we extend the scheduling scheme to the scenario of asynchronous offloading.

### C. Age of Computing Results

The evolution of AoR of a vehicle in the synchronous offloading scenario is shown in Fig. 3. The real-time AoR for vehicle  $i$  in cycle  $n$  and computing slot  $k$  is denoted by  $R_i(n, k)$ . Let  $d_{i,n}$  denote the traveled distance of the vehicle in cycle  $n$ . As the example shown in Fig. 3, vehicle  $i$  is scheduled in the cycles 1, 2, and 5. In these scheduled cycles, when the task computing is completed, the AoR drops to the level of the dashed lines, which represent the traveled distance from the starting of this cycle. Otherwise, if the task is not scheduled in the corresponding cycle, the AoR accumulates due to adding the traveled distance within this cycle.

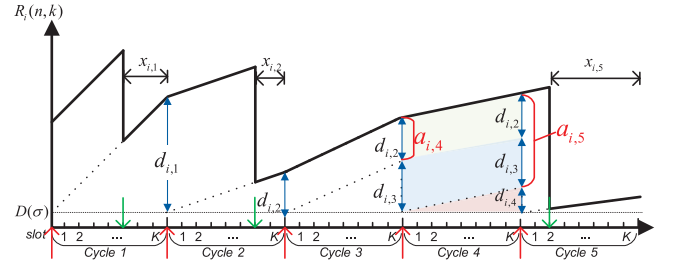


Fig. 3. The evolution of AoR of vehicle  $i$  in five computing cycles, where red arrows represent the task offloading completion instant of a vehicle, and green arrows represent result delivery instant.

The real-time AoR at the beginning of the  $n$ -th cycle,  $R_i(n, 0)$ , can be written as

$$R_i(n, 0) = a_{i,n} + d_{i,n-1} + D(\sigma), \quad (1)$$

where  $d_{i,n-1}$  is the traveled distance in the previous cycle  $n-1$ ,  $D(\sigma)$  is the traveled distance in the offloading time, and the non-negative variable  $a_{i,n}$  is a carryover component contributed to AoR. It can be written as

$$a_{i,n+1} = \begin{cases} a_{i,n} + d_{i,n-1}, & \text{if } i \in \{\mathcal{V}_n \setminus \mathcal{S}_n\}, \\ 0, & \text{if } i \in \mathcal{S}_n, \end{cases} \quad (2)$$

where  $\mathcal{V}_n$  denotes the set of the vehicles that offloaded their observations in computing cycle  $n$ . The sets  $\mathcal{S}_n$  and  $\{\mathcal{V}_n \setminus \mathcal{S}_n\}$  denote the set of vehicles that are scheduled and not scheduled in cycle  $n$ , respectively. Equation (2) shows that if the computing is scheduled in the  $n$ -th cycle, the carryover component will be reset as zero; otherwise, the carryover component accumulates due to adding the vehicle traveled distance in the previous cycle. In the example shown in Fig. 3,  $a_{i,2}$  and  $a_{i,3}$  are zeros since the tasks are scheduled and processed in cycles 1 and 2, respectively. However, the AoR starts accumulating since cycle 3. Thus,  $a_{i,4} = d_{i,2}$ , and  $a_{i,5} = d_{i,2} + d_{i,3}$ . Furthermore, the number of time slots from the task finishing instant to the end of the cycle is denoted as  $x_{i,n}$ , where  $x_{i,n} = 0$  if  $i \in \{\mathcal{V}_n \setminus \mathcal{S}_n\}$  and  $x_{i,n} \geq 0$  otherwise.

The time average AoR of vehicle  $i$  in this process can be represented by the area under age  $R_i(n, k)$  in the age evolution graph, as shown in Fig. 3, normalized by the overall time length. The area under age  $R_i(n, k)$  in cycle  $n$  is denoted by  $Q_{i,n}$  and can be represented as follows:

$$Q_{i,n} = (a_{i,n} + d_{i,n-1})(T - x_{i,n}\omega) + \frac{1}{2}d_{i,n}T + D(\sigma)T. \quad (3)$$

The expected AoR (EAoR) for all vehicles in the communication range of the edge server is:

$$\begin{aligned} E_R &= \lim_{N \rightarrow \infty} \frac{1}{TN} \sum_{n=1}^N \frac{1}{|\mathcal{V}_n|} \sum_{i \in \mathcal{V}_n} Q_{i,n}, \\ &= \lim_{N \rightarrow \infty} \frac{1}{TN} \sum_{n=1}^N \frac{1}{|\mathcal{V}_n|} \sum_{i \in \mathcal{V}_n} (a_{i,n} + d_{i,n-1})(T - x_{i,n}\omega) + \Lambda_{i,n}, \end{aligned} \quad (4)$$

where  $E_R$  denotes the value of EAoR, and  $\Lambda_{i,n}$  represents the constant term of the AoR for vehicle  $i$  in cycle  $n$ , where

$$\Lambda_{i,n} = \frac{1}{2}d_{i,n}T + D(\sigma)T.$$

Neglecting the constant component in (4), the optimization problem for minimizing EAoR can be represented as follows:

$$\begin{aligned} \min_{\pi \in \Pi} \mathbb{E}_{\pi} \left[ \frac{1}{TN} \sum_{n=1}^N \frac{1}{|\mathcal{V}_n|} \sum_{i \in \mathcal{V}_n} (a_{i,n} + d_{i,n-1})(T - x_{i,n}\omega) \right], \\ \text{s.t. } |\mathcal{S}_n| \leq K, \forall n, \\ 0 \leq x_{i,n} < K, x_{i,n} \in \mathbb{Z}, \forall i, n, \end{aligned} \quad (5)$$

where  $\pi$  and  $\Pi$  represent the optimal policy and the feasible policy set, respectively. Since the length of a computing cycle is much shorter than the time for a vehicle leaving the service coverage of an edge server, we apply the long-term policy for vehicle  $i \in \mathcal{V}_n$ . To solve the scheduling problem, we first determine which tasks should be scheduled in the cycle when  $K < |\mathcal{V}_n|$ , *i.e.*, finding  $\mathcal{S}_n$ , based on the maximum AoR by setting the computing slot at the very end of the corresponding computing cycle ( $x_{i,n} = 0$ ). The problem is a long-term optimization problem since the age of a vehicle will accumulate if the tasks are not scheduled properly. After the tasks to be processed in the cycle are selected, we determine when to process the selected tasks, *i.e.*, finding  $x_{i,n}$ , which only affects the instantaneous age in the corresponding cycle.

#### IV. RESTLESS MULTI-ARMED BANDIT FORMULATION AND INDEX POLICY

To solve the optimization problem in (5), a myopic policy is to schedule the vehicle with the highest AoR in each computing cycle. As proved in [37], in a symmetric network, in which all vehicles always have the same mobility, such policy follows a round-robin pattern and attains the minimum AoR. However, the optimality of the myopic policy would be lost if vehicles have diversified and time-variant mobility. Therefore, considering the mobility dynamics, we reformulate the AoR minimization problem into an RMAB problem. We then propose an index-based scheduling scheme, which assigns an index to each vehicle to be scheduled for measuring the value to activate an arm at a particular state. A Whittle's index policy, an optimal policy to a relaxation of the RMAB problem [38], [39], is utilized to schedule the processing order for the synchronous offloading scenario.

##### A. Restless Multi-Armed Bandit Formulation

Different from classic MAB, RMAB considers a generalized bandit process, in which the states of arms can evolve over time even when the arms are not activated. Moreover, instead of selecting only one arm in a decision step in MAB, RMAB can activate multiple arms in a step. In our case, a cycle is a decision step and can schedule  $K$  tasks in each decision step. To determine which tasks to schedule in a computing cycle, *i.e.*, the set  $\mathcal{S}_n$ , we assume  $x_{i,n} = 0$  at first, in which the maximum EAoR is minimized in this step.

The state of a vehicle consists of two parts: the current carryover component, *i.e.*,  $a_{i,n}$  and the past vehicle mobility profile. We use the traveled distance of the vehicle for each cycle in past  $W$  cycles, *i.e.*,  $\delta_{i,n} = \{d_{i,n-W}, \dots, d_{i,n-1}\}$ ,  $\forall i \in |\mathcal{V}_n|$ , as the past mobility profile to predict vehicle future mobility. Denote  $\delta_{i,n}^{-1}$  as the vehicle traveled distance in past  $W - 1$  cycles, *i.e.*,  $\delta_{i,n}^{-1} = \{d_{i,n-W+1}, \dots, d_{i,n-1}\}$ . Let a binary variable  $u_{i,n}$  indicate the action of the RMAB problem for vehicle  $i$  in cycle  $n$ . If the task is scheduled in the cycle, *i.e.*,  $i \in \mathcal{S}_n$ , the arm  $i$  is activated, and  $u_{i,n} = 1$ . Otherwise, the arm  $i$  is not activated, and  $u_{i,n} = 0$ .

According to the evolution of the carryover component  $a_{i,n}$  in (4), when action  $u_{i,n} = 1$ , the state transition probability of vehicle  $i$  can be obtained as follows:

$$P(a_{i,n+1} = 0, \{\delta_{i,n}^{-1}; d_{i,n}\} | a_{i,n}, \delta_{i,n}) = p(d_{i,n} | \delta_{i,n}), \quad (6)$$

where  $p(d_{i,n} | \delta_{i,n})$  is the probability that the vehicle travels  $d_{i,n}$  distance in cycle  $n$ , given past mobility profile  $\delta_{i,n}$ . On the other hand, when action  $u_{i,n} = 0$ , the state transition probability of vehicle  $i$  can be obtained as follows

$$P(a_{i,n+1} = a_{i,n} + d_{i,n-1}, \{\delta_{i,n}^{-1}; d_{i,n}\} | a_{i,n}, \delta_{i,n}) = p(d_{i,n} | \delta_{i,n}). \quad (7)$$

The Whittle index policy introduces a service charge whenever an arm makes active action. Let  $C(\delta_{i,n})$  represent the service charge for vehicle  $i$  given the mobility profile  $\delta_{i,n}$ . The objective of introducing the service charge is to make the passive and active actions equal in a long-term cost [38]. Although there is no instantaneous cost when the arm is passive in a step, it would result in an accumulation of age and a service charge in the future.

##### B. Indexability and Index Policy

In the Whittle index policy, a service charge is the Lagrange multiplier of the RMAB constraint. The original RMAB problem is relaxed and decoupled into subproblems for individual arms such that the service charge for each arm can be evaluated separately. The subproblem for each arm is referred to as decoupled model, and the service charge obtained by the decoupled model only depends on the characteristic of the arm itself [38]. Although problem decomposition would relax the original RMAB problem, a near-optimal solution can be obtained by such decoupled model [37], [38]. However, the service charge cannot be regarded as the Whittle index unless the considered RMAB problem is indexable.

Denote the set of all policies for scheduling vehicle  $i$  by  $\Pi(i)$ . To determine whether the task should be scheduled, we minimize the upper bound performance of the AoR achieved by the selection. Decoupling the objective function (5) for vehicle  $i$ , the new objective function for minimizing the AoR for this individual vehicle is given by

$$\begin{aligned} \min_{\pi(i) \in \Pi(i)} \mathbb{E}_{\pi(i)} [J_i] \\ \text{s.t. } J_i = \frac{1}{TN} \sum_{n=1}^N (a_{i,n} + d_{i,n-1})T + C(\delta_{i,n})u_{i,n}, \end{aligned} \quad (8)$$

To simplify the notation, we neglect the vehicle index  $i$  when we evaluate the problem of the decouple model in (8).

Let  $J_n(a_n, \delta_n)$  be the value function representing the minimum EAoR for problem (8) when the state is  $(a_n, \delta_n)$ , which can be formulated to the following dynamic programming:

$$J_n(a_n, \delta_n) = (a_n + d_{n-1})T + \min_{u_n \in \{0,1\}} \left\{ C(\delta_n)u_n + \sum_{d_n} J_{n+1}(a_{n+1}, \{\delta_n^{-1}; d_n\})p(d_n|\delta_n) \right\}. \quad (9)$$

According to [40], the value function of the optimal policy in a finite-horizon average cost minimization problem can be represented as

$$J^*(a_n, \delta_n) \approx h(a_n, \delta_n) + \lambda TN + o(N) = (a_n + d_{n-1})T + \min_{u_n \in \{0,1\}} \left\{ C(\delta_n)u_n + \sum_{d_n} p(d_n|\delta_n) [h(a_{n+1}, \{\delta_n^{-1}; d_n\}) + \lambda T(N-1) + o(N)] \right\}, \quad (10)$$

where  $\lambda$  is the optimal average cost normalized by the number of computing slots,  $h(a_n, \delta_n)$  is the differential cost function representing the cost incurred when the state transits from  $(a_n, \delta_n)$  to  $(0, \delta_{n+1})$  for the first time, and  $o(N)$  is the cost caused by not completing the execution at the end of the time horizon. All states communicate with one another, which implies that  $h(0, \delta_n) = 0, \forall n$ . Thus, we have differential cost function  $h(a_n, \delta_n)$  as follows:

$$h(a_n, \delta_n) = (a_n + d_{n-1})T - \lambda T + \min_{u_n \in \{0,1\}} \left\{ C(\delta_n)u_n + \sum_{d_n} p(d_n|\delta_n)h(a_{n+1}, \{\delta_n^{-1}; d_n\}) \right\}. \quad (11)$$

By solving the Bellman equation (11), the optimal service charge  $C(a, \delta)$  for vehicle  $i$  can be obtained, given Proposition 1 below.

**Proposition 1:** The service charge  $C(a, \delta)$  for a vehicle in state  $(a, \delta_n)$  is

$$C(a, \delta_n) = \left\{ 2a + d_{n-1} + \mathbb{E} \left[ \sum_{t=0}^{\infty} (a - D_{n+t}) \mathbf{1}(D_{n+t} \leq a) | \delta_n \right] \right\} T, \quad (12)$$

where  $D_{n+t} = \sum_{x=0}^t d_{n+x}$ , and function  $\mathbf{1}(x)$  indicates whether  $x$  is true or not, i.e.,  $x$  is true if  $\mathbf{1}(x) = 1$ , and  $x$  is false if  $\mathbf{1}(x) = 0$ .

*Proof:* See Appendix A.  $\blacksquare$

The purpose of the service charge is to measure the value of activating an arm. A high service charge indicates that the cost of making passive action on the arm is high. If the RMAB problem is indexable, the service charge can be regarded as the Whittle index, and the Whittle index policy is to schedule the tasks with  $K$  highest service charges in a cycle. We denote set  $\mathcal{P}(C)$  as the set of state  $(a, \delta)$  for which the arm is passive according to the

service charge  $C(a, \delta)$ . To prove the indexability of the arm, the following condition should be satisfied:

**Definition 1:** If an arm is indexable, set  $\mathcal{P}(C)$  of the corresponding single-armed bandit process increases monotonically from the empty set  $\emptyset$  to the whole state space as charge  $C$  increases from  $-\infty$  to  $+\infty$  [38], [41].

The condition on indexability indicates that the optimal action of an arm can never switch from passive action to active action with an increase of  $C$  [41]. The RMAB problem is indexable only if all arms are indexable. According to the above definition, we prove the indexability of the considered RMAB problem.

**Theorem 1:** In computing cycle  $n$ , vehicle  $i$  is indexable, and the Whittle index of vehicle  $i$  is identical with the service charge of vehicle  $i$ , where

$$C_i(a_{i,n}, \delta_{i,n}) = \left\{ 2a_{i,n} + d_{i,n-1} + \mathbb{E} \left[ \sum_{t=0}^{\infty} (a_{i,n} - D_{i,n+t}) \times \mathbf{1}(D_{i,n+t} \leq a_{i,n}) | \delta_{i,n} \right] \right\} T. \quad (13)$$

*Proof:* Since both  $a_{i,n}$  and  $d_{i,n}$  are non-negative, there is no such state to make the service charge as a negative number. Therefore,  $\mathcal{P}(C)$  is an empty set when  $C_i = -\infty$ . Moreover, when  $\delta_{i,n}$  is fixed,  $C_i$  increases monotonically with  $a_{i,n}$ . An arm in state  $(a_{i,n}, \delta_{i,n})$  will not switch from passive to active action with the increase of  $C_i$ . If  $C_i$  is  $+\infty$ ,  $a_{i,n}$  can only be  $+\infty$  since  $d_{i,n} < +\infty$ . In such a case,  $\mathcal{P}(C)$  is the whole state set. Therefore, the RMAB problem is indexable, and the Whittle index for the arm is the corresponding service charge.  $\blacksquare$

As shown in the Whittle index, as carryover component  $a_{i,n}$  and traveled distance in the past cycle  $d_{i,n-1}$  increase, the value for activating the arm also increases, which is similar compared to the myopic policy. Furthermore, given the same  $a_{i,n}$  and  $d_{i,n-1}$ , a vehicle with high mobility in the subsequent cycles may have a lower index compared to the one with low mobility. This is because, if the vehicle with high mobility is scheduled, the age of the result would increase fast in the future, and the delivered computing results would be outdated quickly. Given the limited computing resource, the edge server will select the one that is most efficient to reduce the overall AoR in the long term.

According to the closed-form Whittle index in (12), we can schedule tasks in each cycle according to the Whittle index policy. The index implies the value of scheduling a vehicle. From a long-term perspective, scheduling the vehicle with a higher index value leads to a lower AoR of the network. Recall the model for the synchronous offloading scenario. The edge server can collect the Whittle indexes of vehicles and schedule  $K$  vehicles with the highest index values at the beginning of each computing cycle. After selecting tasks, we allocate the computing slot for those tasks. As mentioned above, the slot allocation when set  $\mathcal{S}_n$  is given only affects the instantaneous AoR within a cycle. To obtain  $x_{i,n}$ , we formulate the following optimization problem:

$$\min_{\{x_{i,n}, i \in \mathcal{V}_n\}} \sum_{i \in \mathcal{V}_n} (a_{i,n} + d_{i,n-1})(T - x_{i,n}\omega). \quad (14)$$

As  $(a_{i,n} + d_{i,n-1})$  is constant, the optimal solution of the linear programming (14) is to allocate the slot with the highest  $x_{i,n}$  to the vehicle with the highest carryover component  $a_{i,n} + d_{i,n-1}$ . The policy in this step is sorting those tasks according to  $(a_{i,n} + d_{i,n-1})$  in a decreasing order and scheduling them in sequence.

Although we have obtained the closed-form of the Whittle index and found the Whittle index policy for the synchronous scenario, the future vehicle mobility is unknown. Next, we develop a DRL-assisted method to learn the vehicle mobility in the future from the vehicle mobility profile in the past in Section V.

## V. DRL-ASSISTED SCHEDULING

We denote the term related to the future mobility information in the Whittle index by  $V(a_{i,n}, \delta_{i,n})$ , where

$$V(a_{i,n}, \delta_{i,n}) = \mathbb{E} \left[ \sum_{t=0}^{\infty} (a_{i,n} - D_{i,n+t}) \mathbf{1}(D_{i,n+t} \leq a_{i,n}) | \delta_{i,n} \right]. \quad (15)$$

We refer to  $V(a_{i,n}, \delta_{i,n})$  as the value function for state  $(a_{i,n}, \delta_{i,n})$ . Given the same carryover component  $a_{i,n}$ , the value function for state  $(a_{i,n}, \{\delta_{i,n}^{-1}; d_{i,n}\})$  can be obtained by

$$V(a_{i,n}, \{\delta_{i,n}^{-1}; d_{i,n}\}) = \mathbb{E} \left[ \sum_{t=1}^{\infty} (a_{i,n} - D_{i,n+t} + d_{i,n}) \times \mathbf{1}(D_{i,n+t} - d_{i,n} \leq a_{i,n}) | \{\delta_{i,n}^{-1}; d_{i,n}\} \right]. \quad (16)$$

We then observe that the value function  $V(a_{i,n}, \delta_{i,n})$  can be represented in an alternative way as

$$V(a_{i,n}, \delta_{i,n}) = \mathbb{E} [(a_{i,n} - d_{i,n}) \mathbf{1}(d_{i,n} \leq a_{i,n}) + V(a_{i,n} - d_{i,n}, \{\delta_{i,n}^{-1}; d_{i,n}\}) | \delta_{i,n}]. \quad (17)$$

Given a known traveled distance  $d_{i,n}$ , equation (17) is a Bellman equation and can be solved by dynamic programming. Equation (17) also shows that  $V(0, \delta_{i,n}) = 0$ . To adapt the real-time vehicle mobility, we use a DRL method to approximate  $V(a_{i,n}, \delta_{i,n})$  iteratively, which gets the vehicle mobility from the environment, *i.e.*, the transportation network, and updates the value function based on the current estimation.

Different from other reinforcement learning methods like Deep Q Network (DQN), the considered learning process only focus on the state transition and the value function approximation. The state for the DRL-learning problem is  $s_{i,n} = (a_{i,n}, \delta_{i,n})$ . The reward for each decision step is:

$$r_{i,n} = (a_{i,n} - d_{i,n}) \mathbf{1}(d_{i,n} \leq a_{i,n}),$$

which can be observed from the vehicle mobility in the environment. Instead of observing the vehicle state of the next decision step in traditional reinforcement learning methods, the next state is known if  $d_{i,n}$  is obtained. According to (17), the next state

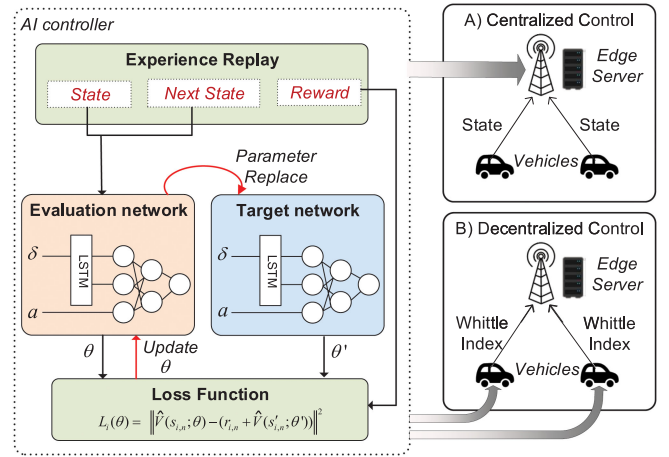


Fig. 4. Deep reinforcement learning structure.

$s'_{i,n}$  is:

$$s'_{i,n} = (a_{i,n} - d_{i,n}, \{\delta_{i,n}^{-1}; d_{i,n}\}).$$

Furthermore, since the state space of the considered problem is large, we adopt deep neural networks to learn the value function  $V(a_{i,n}, \delta_{i,n})$ . Inspired by DQN, we use two neural networks, *i.e.*, the evaluation network and the target network, to learn the value function. As shown in Fig. 4, the evaluation network is trained in an online manner according to a loss function, which is shown as follows:

$$L_i(\theta) = \|\hat{V}(s_{i,n}; \theta) - (r_{i,n} + \hat{V}(s'_{i,n}; \theta'))\|_2^2, \quad (18)$$

where  $\hat{V}(s; \theta)$  represents the estimated value function obtained by weights  $\theta$  in the evaluation network, and  $\hat{V}(s; \theta')$  represents the estimated value function obtained by weights  $\theta'$  in the target network. The weights in the target network are periodically replaced by the weights in the evaluation network.

The algorithm of the proposed DRL-assisted index policy is shown in Algorithm 1. At the beginning of each computing cycle, the Whittle indexes of all vehicles are calculated using (12). The edge server gathers the indexes and schedules the  $K$  vehicles with the highest index values according to the Whittle index policy, which is presented in Line 5 of Algorithm 1. To improve the learning efficiency, in experience replay, we select the state-reward tuples in the memory by weighted sampling in Line 8. Via a positive parameter  $\eta$ , the state with non-zero  $a_{i,n}$  has a higher probability to be selected. As shown in Line 13, a soft parameter replacement scheme for the target network is adopted, where  $\tau$  is a number less than one. Furthermore, we add an long short-term memory (LSTM) layer in the neural network to analyze the time correlation of the vehicle traveling mobility profile in the state. As illustrated in Fig. 4, there are two options to implement the proposed policy:

- Centralized control: An AI controller is deployed at the edge server. Vehicles report their current state, and the edge server stores all the state-reward tuples obtained in this decision step in the memory. The neural network is trained by the aggregated mobility profiles reported by the



**Algorithm 1: Learning-Assisted Whittle Index Policy.**


---

```

1: Initialize the weights of the evaluation and target
   networks ( $\theta$  and  $\theta'$ ), respectively.
2: Initialize the experience replay buffer.
3: for each cycle  $n$  do
   % Schedule tasks with the Whittle index policy.
4: Obtain the initial Whittle index for vehicle  $i$ 
   using (13) corresponding to the estimated value
   function, i.e.,  $\hat{V}(a_{i,n}, \delta_{i,n})$ .
5: Edge server gathers the Whittle indexes from all
   vehicles and schedules the  $K$  vehicles with the
   highest index values.
   % Train the deep neural network.
6: Observe the traveled distance of vehicles, i.e.,
    $\{d_{i,n}, \forall i\}$ .
7: Store state-reward tuple  $(s_{i,n}, s'_{i,n}, r_{i,n})$  in the
   experience replay buffer. Delete the oldest transition
   set if the buffer is full.
8: Sample a mini-batch of  $N$  samples using a weighted
   sampling, where the weight for a state with carryover
   component  $a$  is  $w = \mathbf{1}(a > 0) + \eta$ , and  $\eta > 0$ .
9: for state  $s_0$  with  $a = 0$  in the mini-batch do
10:  $\hat{V}(s_0; \theta') = 0$ .
11: end for
12: Update the weights in the evaluation network by
   minimizing the loss function (18).
13: Update the target network:  $\theta' = \tau(1 - \tau)\theta + \tau\theta'$ .
14: end for

```

---

vehicles. In this case, the edge server can learn the mobility feature of a group of vehicles.

- Decentralized control: At each vehicle, an AI controller is deployed to learn the vehicle driving behavior. Since the Whittle index can be generated independently by each vehicle, vehicles only need to update their Whittle indexes to the server for scheduling. Compared to reporting the state in each cycle in the centralized control, the communication overhead is much lower via the decentralized control.

Either option can be selected, depending on the characteristics of the communication and transportation network scenarios. For example, in an urban area, the centralized method would be preferred since the geometric features in traffic profiles can be learned by the edge server. On the other hand, in areas with smooth traffic flow or limited communication resources, the decentralized control would be preferred to reduce communication overhead.

## VI. SCHEDULING SCHEME FOR ASYNCHRONOUS OFFLOADING

So far, we have discussed the scheduling scheme for the synchronous offloading scenario. In this section, the index-based scheduling scheme is extended to the asynchronous scenario, in which vehicles sense and offload their observation to the edge server at arbitrary instants. As mentioned in the system model, all vehicles have the same sensing cycle, while the proposed

scheme can be extended to adapt the case with different sensing cycle lengths.

Compared to the synchronous scenario, in which a vehicle only has one task to be scheduled in a computing cycle, up to two tasks offloaded by a vehicle could be scheduled in a computing cycle in the asynchronous scenario. For the edge server, vehicle  $i$  with offset  $O_i$  divides a computing cycle into two parts. In the set of the first several computing slots, denoted by  $\mathcal{T}_i^1 = \{1, \dots, \lceil O_i/\omega \rceil\}$ , the edge server can schedule a task that is offloaded in the previous computing cycle by vehicle  $i$  but not scheduled yet. Such task is denoted by task  $n^{(1)}$ . In the set of the rest of computing slots in the cycle, denoted by  $\mathcal{T}_i^2 = \{\lceil O_i/\omega \rceil + 1, \dots, K\}$ , the edge server can schedule a task that is offloaded by vehicle  $i$  in the current computing cycle. Such task is denoted by task  $n^{(2)}$ . Therefore, three indexes can be calculated for this vehicle: The first index is the service charge for scheduling task  $n^{(1)}$  in  $\mathcal{T}_i^1$ , which is denoted by  $C_{i,n}^1$ . The value of  $C_{i,n}^1$  inherits the index of the task in the previous cycle. The second and third indexes are for scheduling task  $n^{(2)}$  in  $\mathcal{T}_i^2$ . If task  $n^{(1)}$  is not scheduled, carryover component  $a_{i,n}$  will increase as given in (2). The corresponding index for scheduling task  $n^{(2)}$  is denoted by  $C_{i,n}^2$ . Otherwise, if task  $n^{(1)}$  is scheduled, carryover component  $a_{i,n}$  will be reset as zero. The corresponding index for scheduling task  $n^{(2)}$  is denoted by  $C_{i,n}^3$ .

Extended from the Whittle index policy for the synchronous scenario, we propose a scheduling scheme for asynchronous offloading (SSA) algorithm, which is shown in Algorithm 2. Instead of evaluating the Whittle index for all vehicles at the beginning of a cycle in the synchronous scenario, the index values are compared in each of computing slots in the asynchronous scenario. The idea behind Algorithm 2 is similar to the Whittle index policy, in which the server schedules the vehicle with the highest Whittle index for each computing slot. At the beginning of the algorithm, the initial Whittle indexes of tasks  $n^{(1)}$  and  $n^{(2)}$  are obtained via Algorithm 1 without considering asynchronous offloading. We use two indicator vectors  $I_{i,n}^{(1)}$  and  $I_{i,n}^{(2)}$  to represent whether vehicle  $i$  is scheduled in part  $\mathcal{T}_i^1$  and part  $\mathcal{T}_i^2$  of the computing cycle, respectively. If the vehicle is scheduled in part  $\mathcal{T}_i^1$ ,  $I_{i,n}^{(1)} = 0$ ; otherwise,  $I_{i,n}^{(1)} = 1$ . Similarly, if the vehicle is scheduled in part  $\mathcal{T}_i^2$ ,  $I_{i,n}^{(2)} = 0$ ; otherwise,  $I_{i,n}^{(2)} = 1$ . In Lines 6 to 13, the vehicle with the highest index is scheduled for each slot iteratively. As shown in Lines 8 and 9 in Algorithm 2, we define variables  $F_{i,n}^{(1)}$  and  $F_{i,n}^{(2)}$  to represent the initial index for  $n^{(1)}$  and  $n^{(2)}$  of vehicle  $i$  in a computing cycle. To avoid scheduling a task twice in different computing cycles, the values of  $F_{i,n}^{(1)}$  and  $F_{i,n}^{(2)}$  will be -1 if the corresponding task is scheduled. Furthermore, the index for task  $n^{(2)}$  will change if the computing policy for task  $n^{(1)}$  changes. The approximated Whittle index for the vehicle in slot  $k$  is defined as follows:

$$C_{i,k} = F_{i,n}^{(1)} \mathbf{1}(k \in \mathcal{T}_i^1) + F_{i,n}^{(2)} \mathbf{1}(k \in \mathcal{T}_i^2).$$

The vehicle with the highest approximated index is selected to be scheduled in the slot as presented in Line 9. Lines 10 to 12 update the indicator according to the policy made by Line 9.



---

**Algorithm 2:** Scheduling Scheme for Asynchronous Offloading (SSA).

---

```

1: Initialize indicator vectors  $I_{i,n}^{(1)} = I_{i,n-1}^{(2)}$ , and  $I_{i,n}^{(2)} = 0$ ,  $\forall i$ .
2: Set the indicator vectors  $J_{i,n}^{(1)} = I_{i,n}^{(1)}$ , and  $J_{i,n}^{(2)} = I_{i,n}^{(2)}$ ,  $\forall i$ .
3: Obtain the initial Whittle indexes,  $C_{i,n}^1$ ,  $C_{i,n}^2$ , and  $C_{i,n}^3$ , using Algorithm 1.
4: Initialize counter  $e = 1$ .
5: Initialize assignment vector  $\mathbf{W} = \mathbf{0}$ 
6: for  $k = \{1, \dots, K\}$  do
7:    $F_{i,n}^{(1)} = C_{i,n}^1 J_{i,n}^{(1)} - (1 - J_{i,n}^{(1)})$ ,  $\forall i$ .
8:    $F_{i,n}^{(2)} = [C_{i,n}^2 J_{i,n}^{(1)} + C_{i,n}^3 (1 - J_{i,n}^{(1)})] J_{i,n}^{(2)} - (1 - J_{i,n}^{(2)})$ ,  $\forall i$ .
9:   For slot  $k$ ,  $i^* = \operatorname{argmax}_i C_{i,k}$ 
10:  If  $C_{i,k} < 0$ :  $W_k = \emptyset$ .
11:  Else if  $\mathbf{1}(k \in \mathcal{T}_i^1) == 1$ :  $J_{i^*,n}^{(1)} = 0$ ,  $W_k = i^*$ .
12:  Else if  $\mathbf{1}(k \in \mathcal{T}_i^2) == 0$ :  $J_{i^*,n}^{(2)} = 0$ ,  $W_k = i^*$ .
13: end for
14: if  $J_{i,n}^{(1)} == I_{i,n}^{(1)}$ ,  $J_{i,n}^{(2)} == I_{i,n}^{(2)}$ ,  $\forall i$  then
15:    $C_{i,n+1}^1 = [C_{i,n}^2 J_{i,n}^{(1)} + C_{i,n}^3 (1 - J_{i,n}^{(1)})] J_{i,n}^{(2)} - (1 - J_{i,n}^{(2)})$ .
16:   Assignment is completed.
17: else
18:    $I_{i,n}^{(1)} = J_{i,n}^{(1)}$ ,  $I_{i,n}^{(2)} = J_{i,n}^{(2)}$ ,  $\forall i$ .
19:    $e = e + 1$ .
20:   If  $e < e_{max}$ : Return to Line 5
21:   Else: Return to Line 15
22: end if

```

---

Since the task allocated to a slot with a small number has a short queuing time, the algorithm allocates the tasks to slots 1 to  $K$  in a cycle consecutively, and the computing slot with a small number has priority in selecting a task. Since the changed policy made in the first part of a computing cycle, *i.e.*,  $\mathcal{T}_i^1$ , for vehicle  $i$  results in a different approximated index in the second part of the computing cycle, *i.e.*,  $\mathcal{T}_i^2$ , the indicators may change as the policy changes. The SSA algorithm iteratively updates the computing policy until the indicator vectors no longer change or the iteration number reaches  $e_{max}$ .

The time complexity of the algorithm primarily depends on the number of iterations of the outer loop in Algorithm 2, which checks the convergence of the indicator vectors. In each iteration, the time complexity is the same as the selective sorting algorithm if the time complexity on solving the Whittle indexes using equation (13) is neglected. Since a vehicle has two indexes during a computing cycle, the time complexity of comparing index values at all vehicles is  $2K|\mathcal{V}_n|$ . Furthermore, since the indicators may change when the policy changes, the convergence of the algorithm may not be guaranteed. The algorithm will be terminated after  $e_{max}$  iterations if the algorithm cannot converge. Thus, the time complexity of Algorithm 2 is  $O(2K|\mathcal{V}_n|e_{max})$  in the worst case.

TABLE I  
VEHICLE SPEED DISTRIBUTION IN A COMPUTING CYCLE (KM/H)

	Group 1		Group 2		Group 3		Group 4		Group 5	
	$\mu$	$\xi$	$\mu$	$\xi$	$\mu$	$\xi$	$\mu$	$\xi$	$\mu$	$\xi$
<b>P1</b>	20	20	40	20	60	20	80	40	60	40
<b>P2</b>	20	0	40	0	60	0	80	0	60	0

## VII. SIMULATION RESULTS

In this section, we first present the numerical results of Whittle index policy that is found by the proposed scheduling scheme and compare the performance with two other benchmarks: *highest-AoR-first* and *round-robin* scheduling policies. The highest-AoR-first policy is a myopic policy that always schedules the task with the highest AoR first, and the round-robin computing policy schedules vehicles in a circular order. We also compare the performance of the proposed scheme with a *SSA-only* scheme. In the SSA-only scheme, we schedule the vehicles by the proposed scheduling scheme shown in Algorithm 2. Different from the proposed scheme, the SSA-only scheme uses vehicles' AoR to guide the task scheduling rather than the Whittle index. We simulate our computing scheduling scheme and the learning approach based on a real-world vehicle mobility dataset.<sup>2</sup> The centralized learning model is considered in the simulation.

### A. Numerical Results

In this section, we generate vehicle mobility profiles and simulate the proposed index-based scheduling scheme. In this numerical example, the edge server schedules the vehicles' tasks according to the vehicle mobility profiles. We consider a total of 150 vehicles traveling under the service coverage of the edge server. According to the different speed distributions, they are divided into five groups with the same number of 30 vehicles in a group. We consider two mobility profiles, *i.e.* P1 and P2, which is summarized in Table I. The speed of vehicles in P1 follows normal distributions, where  $\mu$  and  $\xi$  represent mean and standard deviation, respectively. In terms of profile P2, all vehicles travel at constant speeds. For both mobility profiles, we evaluate the performance of EAoR for both synchronous and asynchronous scenarios, where the offset of vehicles in the asynchronous scenario is selected randomly from the interval  $[0, T)$ . The length of a computing cycle  $T$  is 2 seconds. The policies are used in 500 cycles to obtain the average AoR for all vehicles. We demonstrate the performance of the computing policies versus the number of slots in each cycle, *i.e.*,  $K$ . The term *sync* in the legend represents the synchronous offloading scenario, and *async* represents the asynchronous offloading scenario. The performance of EAoR with mobility profile P1 is shown in Fig. 5 with the number of slots in a cycle, *i.e.*,  $K$ , changing from 20 to 150. In both the synchronous and asynchronous scenarios, our proposed scheme has the lowest AoR among all scheduling policies. In the asynchronous scenario, when  $K$  is 80, the proposed scheme can reduce the EAoR by up to 40% and 30% compared to

<sup>2</sup>The data came from the Didi Gaia Data Opening Plan: <https://gaia.didichuxing.com>

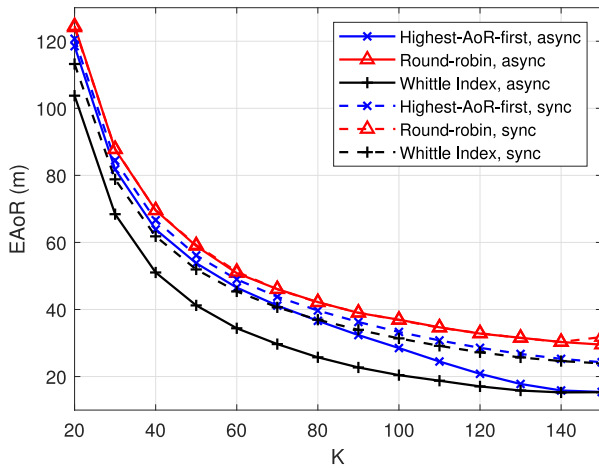


Fig. 5. EAoR versus the number of computing slots in a cycle with mobility profile P1.

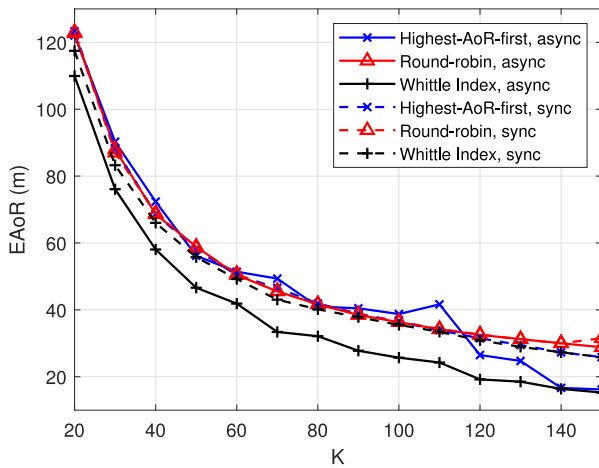


Fig. 6. EAoR versus the number of computing slots in a cycle with mobility profile P2.

the round-robin and highest-AoR-first policies, respectively. In the synchronous scenario, vehicles in the Whittle index policy experience a higher AoR compared to the asynchronous scenario, and the performance gaps between the proposed scheme and the other two policies are smaller. This is because all requests are arrived at the same time in the synchronous scenario. Some tasks are inevitably being allocated to the computing slots at the end of the computing cycle, which will increase the AoR significantly. Although the same could also happen using the highest-AoR-first policy, when the computing slots are insufficient, the tasks with the highest AoR are likely about to outdated. Processing the outdated tasks rather than the newly arrived tasks results in a high EAoR when  $K$  is small. Therefore, the AoR of the highest-AoR-first policy with the asynchronous offloading is not reduced significantly compared to that with the synchronous offloading, especially when computing slots are insufficient. The performance of EAoR with mobility profile P2 is shown in Fig. 6, in which all vehicles travel at constant speed. In such a case, the major advantage of the Whittle index policy,

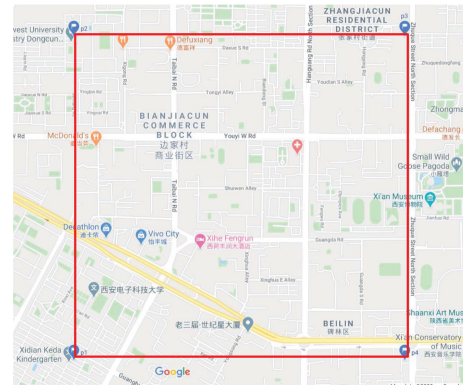


Fig. 7. A snap shot of the simulation region.

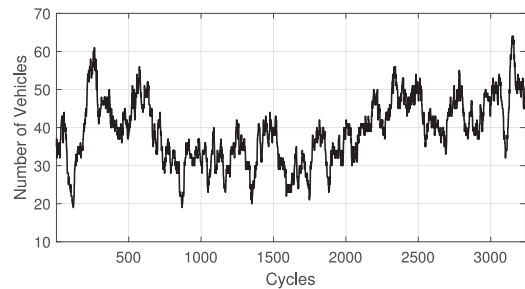


Fig. 8. The number of vehicles in the simulation time window.

which schedules the tasks according to the mobility dynamics, has less influence on the scheduling performance. Therefore, the Whittle index policy is close in performance with the other two policies in the synchronous scenario. Moreover, as proved in [37], when all vehicles have the same speed at the same time, the highest-AoR-first policy is equivalent to the round-robin policy and provides the optimal scheduling solution. Since vehicles have similar mobility profiles, the three policies would all achieve near-optimal performance. On the other hand, for the asynchronous scenario, the Whittle-index policy has better performance since it always schedules the tasks considering the age.

### B. Simulation in a Real Dataset

In this subsection, we simulate the proposed index-based scheduling scheme and the learning approach based on the taxi driving trace data collected by Didi Gaia Data Opening Plan in Xi'an, China. We investigate the vehicle trajectories from 16:30 to 18:20 on Oct. 1st, 2016 in a  $2 \text{ km} \times 2 \text{ km}$  area as shown in Fig. 7. The attributes of a set of data include timestamp, vehicle ID, and vehicle location (longitude and latitude). The length of a computing cycle is 2 seconds. Thus, there are 3300 computing cycles in the simulation. We select the vehicles which travel in the  $2 \text{ km} \times 2 \text{ km}$  area consistently in the prediction window  $W$ , where  $W$  is 80 in our simulation. Under such condition, there are 126,376 sets of data to be analyzed, and the number of vehicles counted in each computing cycle is shown in Fig. 8. The settings of the DRL algorithm are summarized as follows: The learning rate is 0.001, the soft replacement parameter  $\tau$  is

TABLE II  
NEURAL NETWORK STRUCTURE

Layers	Number of neurons	Activation function
LSTM	40	relu
Fully connected 1	120 (50% dropout)	relu
Fully connected 2	60	relu
Fully connected 3	30	relu
Fully connected 4	15	elu
Fully connected 5	1	elu

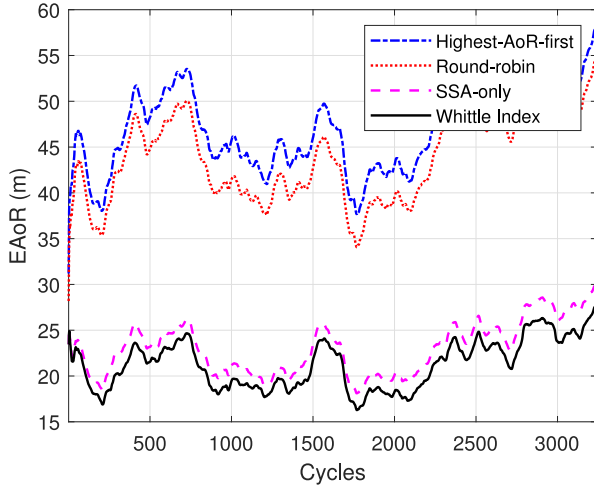


Fig. 9. EAO<sub>R</sub> when the number of computing slots in each cycle, *i.e.*,  $K$ , is 10.

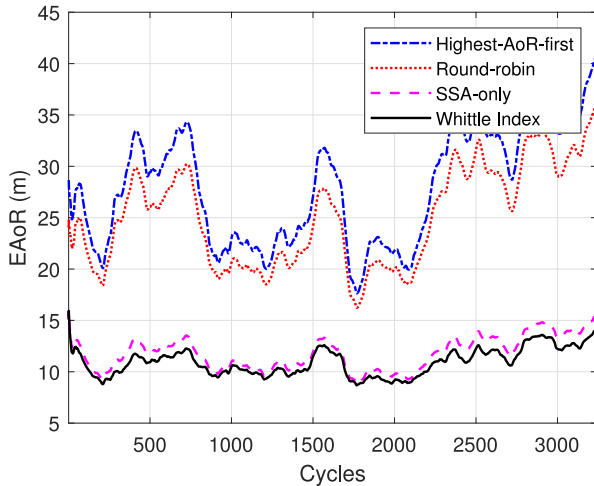


Fig. 10. EAO<sub>R</sub> when the number of computing slots in each cycle, *i.e.*,  $K$ , is 20.

0.8, the memory can store 800 state-reward tuples, and the batch size is 48. The structure of the neural network is presented in Table II. We apply a moving average to measure the EAO<sub>R</sub> value for the vehicles in an online manner, where the window size is 200 cycles.

The EAO<sub>R</sub> performances in the simulation time window are shown in Figs. 9 and 10, where the number of computing slots in a cycle is 10 and 20, respectively. For both scenarios, the proposed scheme reduces the Ao<sub>R</sub> significantly. Although the computing resource is limited, the proposed scheme maintains the Ao<sub>R</sub> around 20 m and 10 m when  $K$  is 10 and 20,

TABLE III  
AVERAGE ALGORITHM RUNNING TIME (IN SECOND)

Number of slots	Highest-AoR-first	Round-robin	SSA-only	Whittle Index
$K = 10$	0.0009	0.0003	0.0321	0.1176
$K = 20$	0.0022	0.0005	0.0626	0.1550

respectively. Moreover, compared to the SSA-only scheme which schedules the tasks without considering future mobility, our proposed scheme can adapt to the time-variant vehicle mobility in real-time and further reduce the EAO<sub>R</sub>. As the number of slots increases, the performance gap between the proposed scheme and SSA-only scheme also increases since the influence of future mobility on Ao<sub>R</sub> performance is significant when computing slots are insufficient.

The average running time of the proposed and benchmark schemes is presented in Table III. The simulation is performed on a machine with Intel i7-9750H CPU. Since the proposed indexed-based scheduling scheme obtains the Whittle indexes from the neural network and attains scheduling policy iteratively, the running time is the highest compared to other benchmarks. However, the time consumption for running the proposed scheme is still on the millisecond level and can be acceptable to real-time applications. Moreover, the time consumed on the neural network dominates the overall running time of the proposed scheme, which can be seen from the time difference between SSA-only and the proposed scheme. Such running time can be further decreased with the advancement of GPUs.

## VIII. CONCLUSION

In this paper, we have proposed a proactive indexed-based scheduling scheme based on predicted future vehicle mobility dynamics for the edge server to process real-time computing tasks offloaded by autonomous vehicles. We have adopted a novel DRL approach to estimate the mobility of the vehicles in the future and assist the evaluation of scheduling indexes. As a result, the proposed scheduling scheme can adapt to real-time vehicle mobility and support safety-related computing services with low computing complexity and communication overhead. A potential future research direction is to analyze how parameters such as sensing cycle length impact the vehicles' Ao<sub>R</sub> and design a computing scheduling scheme according to diversified service requirements of the vehicles.

## APPENDIX

### A. Proof of Proposition 1

To obtain the service charge, similar to the approach in [37], we assume a threshold policy at first. The threshold is denoted by  $A$ . The arm is passive when the carryover component  $0 \leq a_n \leq A$  and active when  $a_n > A$ .

According to the threshold policy,  $u_n = 1$  when  $a_n > A$ . The following condition has to be satisfied:

$$C(\delta_n) > \sum_{d_n} h(a_n + d_{n-1}, \{\delta_n^{-1}; d_n\}) p(d_n | \delta_n),$$

when  $a_n > A$ . (22)



Let  $j^+$  be a positive number. The differential cost in state  $(A + j^+, \delta_n)$  for the case  $a_n > A$  can be simplified as

$$h(A + j^+, \delta_n) = C(\delta_n) + (A + j^+ + d_{n-1} - \lambda)T. \quad (23)$$

The equation shows that  $h(A + j^+, \delta_n)$  monotonically increases with  $j^+$ .

On the other hand,  $u_n = 0$  when  $0 \leq a_n \leq A$ . Assume that the vehicle speed precision level is high, and existing a state  $(A, \delta_n)$  has a service charge as follows:

$$C(A, \delta_n) = \sum_{d_n} h(A + d_{n-1}, \{\delta_n^{-1}; d_n\})p(d_n|\delta_n). \quad (24)$$

Let  $j^-$  be a negative number that greater than  $-A$ . Then, the differential cost in state  $(A + j^-, \delta_n)$  for the case  $0 \leq a_n \leq A$  is

$$h(A + j^-, \delta_n) = (A + j^- + d_{n-1} - \lambda)T + \sum_{d_n} h(A + j^- + d_{n-1}, \{\delta_n^{-1}; d_n\})p(d_n|\delta_n). \quad (25)$$

Similarly, equation (25) shows that  $h(A + j^-, \delta_n)$  monotonically increases with  $j^-$ . Given the threshold policy, the term of  $h(A + j^- + d_{n-1}, \{\delta_n^{-1}; d_n\})$  can be determined according to the value of the term of  $(A + j^- + d_{n-1})$ . The differential cost in (25) in could be further derived to (19), shown at the bottom of this page, where  $B_{n+t} = \sum_{x=-1}^t d_{n+x}$  shown at the bottom of this page, shown at the bottom of this page. In a similar manner, we then evaluate the differential cost for the next states recursively, which could be summarized as (20).

Next, to simplify the complex form in (20), the term of  $h(A + d_{n-1}, \{\delta_n^{-1}; d_n\})p(d_n|\delta_n)$  in (24) is expanded according to the threshold policy, we have

$$C(\delta_n) = (A + d_{n-1} - \lambda)T + \mathbb{E}[d_n|\delta_n]T + \sum_{d_n} C(\{\delta_n^{-1}; d_n\}) \times p(d_n|\delta_n)[P(d_{n-1} > -j^-) + P(d_{n-1} \leq -j^-)]. \quad (26)$$

Then, we recursively apply (26) to represent service charge  $C(\delta_n)$ . Finally, we obtain the relation in (21) shown at the bottom of this page. Combining (20) and (21) together, the differential cost on state  $(A + j^-, \delta_n)$  is obtained as follows:

$$h(A + j^-, \delta_n) = (A + d_{n-1} - \lambda)T + 2j^- + C(\delta_n) + \mathbb{E} \left[ \sum_{t=-1}^{\infty} B_{n+t} \mathbf{1}(B_{n+t} \leq -j^-) | \delta_n \right] T + j^- T P(B_{n+t} \leq -j^- | \delta_n). \quad (27)$$

Utilizing the property of  $h(0, \delta_n) = 0$  and letting  $j^- = -A$  in (27), the service charge  $C(\delta_n)$  is:

$$C(\delta_n) = (A + \lambda - d_{n-1})T + \mathbb{E} \left[ \sum_{t=-1}^{\infty} (A - B_{n+t}) \mathbf{1}(B_{n+t} \leq A) | \delta_n \right] T. \quad (28)$$

$$h(A + j^-, \delta_n) = (A + j^- + d_{n-1} - \lambda)T + P(d_{n-1} \leq -j^- | \delta_n) \sum_{d_n} \left[ (A + j^+ + d_{n-1} + d_n - \lambda)T + \sum_{d_{n+1}} h(A + j^- + d_{n-1} + d_n, \{\delta_n^{-2}; d_n; d_{n+1}\})p(d_{n+1} | \{\delta_n^{-1}; d_n\}) \right] p(d_n|\delta_n) + P(d_{n-1} > -j^- | \delta_n) \sum_{d_n} [C(\{\delta_n^{-1}; d_n\}) + (A + j^- + d_{n-1} + d_n - \lambda)T] p(d_n|\delta_n) \quad (19)$$

$$h(A + j^-, \delta_n) = 2(A + j^- + d_{n-1} - \lambda)T + \mathbb{E}[d_n|\delta_n]T + (A + j^- - \lambda)T \sum_{t=-1}^{\infty} P(B_{n+t} \leq -j^- | \delta_n) + \mathbb{E} \left[ \sum_{t=-1}^{\infty} B_{n+t+2} \mathbf{1}(B_{n+t} \leq -j^-) | \delta_n \right] T + \mathbb{E} \left[ \sum_{t=0}^{\infty} C(\hat{\delta}_{n+t+2}) \mathbf{1}(B_{n+t} > -j^-, B_{n+t-1} \leq -j^-) | \delta_n \right] + \sum_{d_n} C(\{\delta_n^{-1}; d_n\})p(d_n|\delta_n)P(d_{n-1} > -j^- | \delta_n), \text{ where } \hat{\delta}_{n+t} = \{\delta_n^{-t}; d_n; \dots; d_{n+t-1}\} \quad (20)$$

$$C(\delta_n) = (A + j^- + d_{n-1} - \lambda)T + \mathbb{E}[d_n|\delta_n]T + \mathbb{E} \left[ \sum_{t=0}^{\infty} C(\hat{\delta}_{n+t+2}) \mathbf{1}(B_{n+t} > -j^-, B_{n+t-1} \leq -j^-) | \delta_n \right] + \sum_{d_n} C(\{\delta_n^{-1}; d_n\})p(d_n|\delta_n)P(d_{n-1} > -j^- | \delta_n) + (A - \lambda)T \left[ \sum_{t=-1}^{\infty} P(B_{n+t} \leq -j^- | \delta_n) \right] + \mathbb{E} \left[ \sum_{t=-1}^{\infty} (d_{n+t+2} + d_{n+t+1}) \mathbf{1}(B_{n+t} \leq -j^-) | \delta_n \right] T \quad (21)$$

According to equation (28), we further obtain the service charge for state  $\{\delta_n^{-1}; d_n\}$ , where

$$C(\{\delta_n^{-1}; d_n\}) = (A + \lambda - d_n)T + \mathbb{E} \left[ \sum_{t=-1}^{\infty} (A - B_{n+t+1} + d_{n-1}) \times \mathbf{1}(B_{n+t+1} - d_{n-1} \leq A) | \delta_n \right] T. \quad (29)$$

Substituting (29) into (26) yields:

$$\begin{aligned} C(A, \delta_n) &= (2A + d_{n-1})T \\ &+ \mathbb{E} \left[ \sum_{t=-1}^{\infty} (A - B_{n+t+1} + d_{n-1}) \mathbf{1}(B_{n+t+1} - d_{n-1} \leq A) | \delta_n \right] T \\ &= (2A + d_{n-1})T + \mathbb{E} \left[ \sum_{t=0}^{\infty} (A - D_{n+t}) \mathbf{1}(D_{n+t} \leq A) | \delta_n \right] T. \end{aligned} \quad (30)$$

From (23) and (27), we see that the differential cost  $h(a, \delta_n)$  increases with carryover component  $a$ . Therefore, the solution of the Bellman equation (11) follows a threshold policy. To evaluate the service charge of state  $(a, \delta)$ , we substitute  $A$  to  $a$ , and the corresponding service charge is presented in equation (30).

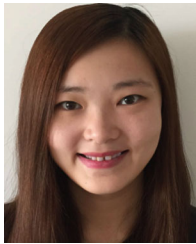
#### ACKNOWLEDGMENT

The authors would like to thank Conghao Zhou and Lechuan Peng for their help with simulation data processing. The simulation dataset is from the Didi Chuxing GAIA Initiative.

#### REFERENCES

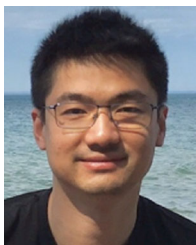
- [1] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.
- [2] S. Zhang, J. Chen, F. Lyu, N. Cheng, W. Shi, and X. Shen, "Vehicular communication networks in the automated driving era," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 26–32, Sep. 2018.
- [3] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–36, Oct. 2019.
- [4] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.
- [5] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.
- [6] L. Li, Y. Li, and R. Hou, "A novel mobile edge computing-based architecture for future cellular vehicular networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.
- [7] Queensland Government's Transport Department, "Stopping distances on wet and dry roads," [Online]. Available: <https://www.qld.gov.au/transport/safety/road-safety/driving-safely/stopping-distances/graph>
- [8] S. Maheshwari, D. Raychaudhuri, I. Sesar, and F. Bronzino, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," in *Proc. IEEE/ACM Symp. Edge Comput.*, 2018, pp. 286–299.
- [9] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.
- [10] J. Cao, L. Yang, and J. Cao, "Revisiting computation partitioning in future 5G-based edge computing environments," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2427–2438, Apr. 2019.
- [11] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.
- [12] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Mar. 2020.
- [13] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.
- [14] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.
- [15] Y. Sun *et al.*, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.
- [16] J. Zhou, D. Tian, Y. Wang, Z. Sheng, X. Duan, and V. C. M. Leung, "Reliability-optimal cooperative communication and computing in connected vehicle systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1216–1232, May 2020.
- [17] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, Sep. 2017.
- [18] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.
- [19] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5 G," *White paper, ETSI*, vol. 11, no. 11, pp. 1–16, 2014.
- [20] F. Giust *et al.*, "Multi-access edge computing: The driver behind the wheel of 5G-connected cars," *IEEE Commun. Standards Mag.*, vol. 2, no. 3, pp. 66–73, Sep. 2018.
- [21] F. Lyu *et al.*, "LEAD: Large-scale edge cache deployment based on spatio-temporal WiFi traffic statistics," *IEEE Trans. Mobile Comput.*, to be published, doi: [10.1109/TMC.2020.2984261](https://doi.org/10.1109/TMC.2020.2984261).
- [22] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Mar. 2020.
- [23] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.
- [24] M. Li, L. Zhao, and H. Liang, "An SMDP-based prioritized channel allocation scheme in cognitive enabled vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 7925–7933, Sep. 2017.
- [25] Y. Liu *et al.*, "Dependency-aware task scheduling in vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4961–4971, Jun. 2020.
- [26] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.
- [27] M. Li, C. Chen, H. Wu, X. Guan, and X. Shen, "Age-of-information aware scheduling for edge-assisted industrial wireless networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5562–5571, Aug. 2021.
- [28] T. D. T. Nguyen, V. D. Nguyen, V. Pham, L. N. T. Huynh, M. D. Hossain, and E. Huh, "Modeling data redundancy and cost-aware task allocation in MEC-enabled internet-of-vehicles applications," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1687–1701, Aug. 2021.
- [29] X. Chen *et al.*, "Age of information aware radio resource management in vehicular networks: A proactive deep reinforcement learning perspective," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2268–2281, Apr. 2020.
- [30] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. 8th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, 2011, pp. 350–358.
- [31] S. Zhang, J. Li, H. Luo, J. Gao, L. Zhao, and X. Shen, "Low-latency and fresh content provision in information-centric vehicular networks," *IEEE Trans. Mobile Comput.*, to be published, doi: [10.1109/TMC.2020.3025201](https://doi.org/10.1109/TMC.2020.3025201).
- [32] C. Li, S. Li, and Y. T. Hou, "A general model for minimizing age of information at network edge," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 118–126.
- [33] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Analysis on computation-intensive status update in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4353–4366, Apr. 2020.
- [34] M. Chen, Y. Xiao, Q. Li, and K. Chen, "Minimizing age-of-information for fog computing-supported vehicular networks with deep q-learning," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

- [35] A. Alabbasi and V. Aggarwal, "Joint information freshness and completion time optimization for vehicular networks," *IEEE Trans. Serv. Comput.*, to be published, doi: [10.1109/TSC.2020.2978063](https://doi.org/10.1109/TSC.2020.2978063).
- [36] B. Yin *et al.*, "Only those requested count: Proactive scheduling policies for minimizing effective age-of-information," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 109–117.
- [37] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2637–2650, Dec. 2018.
- [38] K. Liu and Q. Zhao, "Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access," *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5547–5567, Nov. 2010.
- [39] P. Whittle, "Restless bandits: Activity allocation in a changing world," *J. Appl. Probability*, vol. 25, no. A, pp. 287–298, 1988.
- [40] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. MA, USA: Athena Scientific Belmont, 1995.
- [41] Y. Qian, C. Zhang, B. Krishnamachari, and M. Tambe, "Restless poachers: Handling exploration-exploitation tradeoffs in security domains," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2016, pp. 123–131.



**Mushu Li** (Student Member, IEEE) received the B.Eng. degree from the University of Ontario Institute of Technology (UOIT), Canada, in 2015, and the M.A.Sc. degree from Ryerson University, Canada, in 2017. She is currently working toward the Ph.D. degree in electrical engineering with University of Waterloo, Canada. She was the recipient of Natural Science and Engineering Research Council of Canada Graduate Scholarship in 2018, and Ontario Graduate Scholarship in 2015 and 2016, respectively. Her research interests include the system optimization in

VANETs and machine learning in wireless networks.



**Jie Gao** (Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical and computer engineer from the University of Alberta, Edmonton, AB, Canada, in 2009 and 2014, respectively. He joined the Department of Electrical and Computer Engineering, Marquette University, Milwaukee, WI, USA, as an Assistant Professor in August 2020. He was a Research Associate with the University of Waterloo, Waterloo, ON, Canada, from 2019 to 2020 and a Postdoctoral Fellow with Ryerson University, Toronto, ON, Canada, from 2017 to 2019. His research interests

include machine learning for communications and networking, Internet of Things (IoT) and industrial IoT solutions, and cloud and edge computing. Dr. Gao was the recipient of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY Top Reviewer Award (2018), the Ontario Centres of Excellence TalentEdge Fellowship (2016), and the Natural Science and Engineering Research Council of Canada Postdoctoral Fellowship (2016). He is the Editor for Springer *Peer-to-Peer Networking and Applications* and IEEE ACCESS.



**Lian Zhao** (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2002. She joined the Department of Electrical, Computer, & Biomedical Engineering with Ryerson University, Toronto, Canada, in 2003 and has been a Professor in 2014. Her research interests include the areas of wireless communications, radio resource management, mobile edge computing, caching and communications, and vehicular ad-hoc networks.

She has been selected as an IEEE Communication Society (ComSoc) Distinguished Lecturer (DL) for 2020 and 2021, she was the recipient of the Best Land Transportation Paper Award from IEEE Vehicular Technology Society in 2016, Top 15 Editor Award in 2015 for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, Best Paper Award from the 2013 International Conference on Wireless Communications and Signal Processing (WCSP), and the Canada Foundation for Innovation (CFI) New Opportunity Research Award in 2005.

She has been the Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and IEEE INTERNET OF THINGS JOURNAL. She was the Co-Chair of Wireless Communication Symposium for IEEE Globecom 2020 and IEEE ICC 2018, Local Arrangement Co-Chair for IEEE VTC Fall 2017 and IEEE Infocom 2014, Co-Chair of Communication Theory Symposium for IEEE Globecom 2013. She was the committee member for NSERC (Natural Science and Engineering Research Council of Canada) Discovery Grants Evaluation Group for Electrical and Computer Engineering 2015 to 2018. She is a licensed Professional Engineer in the Province of Ontario, a Senior Member of the IEEE Communication Society and Vehicular Society.



**Xuemin (Sherman) Shen** (Fellow, IEEE) received the B.A.Sc. degree from Dalian Maritime University, China, the M.A.Sc. and Ph.D. degrees in electrical engineering from Rutgers University, New Brunswick, NJ, USA. He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. Dr. Shen is a Registered Professional Engineer of Ontario, Canada,

an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Dr. Shen is the 5th in Canada in 2020 Ranking of Top 1000 Scientists in the field of Computer Science and Electronics, Guide2Research Forum. He was the recipient of the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) presents in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He was also recipient of the Excellent Graduate Supervision Award in 2006 and Outstanding Performance Award 5 times from the University of Waterloo, and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. Dr. Shen was the General Chair for ACM Mobihoc'15, the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. Dr. Shen is the elected IEEE Communications Society Vice President for Technical and Educational Activities, Vice President for Publications, Member-at-Large on the Board of Governors. He was/is the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, *IET Communications*, and *Peer-to-Peer Networking and Applications*.