Real-Time Search-Driven Caching for Sensing Data in Vehicular Networks

Mingliu Liu, *Member, IEEE*, Deshi Li, Huaqing Wu, *Member, IEEE*, Feng Lyu, *Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Real-time search is essential for accessing specific sensing data (SD) in vehicular networks to support safe, efficient, and intelligent road services. Considering the tremendous data volume, the SD search process should be carefully devised to avoid excessive retrieval and transmission delay. To alleviate the communication and computational burden for sensing devices and the cloud server, roadside edges are adopted to cache the SD in advance. Given a short lifetime of SD, the caching scheme is required to be efficient in facilitating both the search process and uplink/downlink transmission, which is quite challenging due to the coupling of resource allocation decisions. To guarantee the search efficacy and enhance the caching resource utilization, we propose a real-time search-driven caching (RSC) paradigm to enable the cooperation among storage-constrained edges. A hierarchical indexing framework is first introduced for cached data, based on which we then devise a search utility model to quantify the expected data freshness and response delay. With the objective of maximizing the long-term search reward, the RSC problem is formulated by jointly considering the search requests and utility model. A deep-reinforcement-learning-based caching (DRLC) method is proposed to solve the problem. Specifically, an action transition module is introduced to lower the computational complexity via reducing the selection space of caching actions. Extensive simulations are carried out based on the real trace data in Creteil, France, and results show that the intelligent DRLC method can improve the real-time search performance significantly comparing to the benchmark methods.**

Manuscript received July 28, 2021; revised November 9, 2021; accepted November 29, 2021. Date of publication December 14, 2021; date of current version July 7, 2022. This work was supported in part by the China Postdoctoral Science Foundation under Grant 2020M672412; in part by the National Natural Science Foundation of China under Grant 62002389; in part by the Natural Science Foundation of Hunan Province, China, under Grant 2021JJ20079; in part by the Young Elite Scientists Sponsorship Program by CAST under Grant YESS20200238; in part by the Young Talents Plan of Hunan Province of China under Grant 2021RC3004; and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. This article was presented in part at the IEEE International Conference on Communications (ICC) 2021. *(Corresponding author: Deshi Li.)*

Mingliu Liu is with the School of Electronic Information, Wuhan University, Wuhan 430072, China, and also with the Electric Power Research Institute, State Grid Hubei Electric Power Research Institute, Wuhan 430077, Hubei, China (e-mail: liumingliu@whu.edu.cn).

Deshi Li is with the School of Electronic Information, Wuhan University, Wuhan 430072, China (e-mail: dsli@whu.edu.cn).

Huaqing Wu is with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada (e-mail: wu482@mcmaster.ca).

Feng Lyu is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: fenglyu@csu.edu.cn).

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

Digital Object Identifier 10.1109/JIOT.2021.3134964

***Index Terms*—Cooperative caching, real-time search, sensing data (SD) caching, vehicular networks.**

I. Introduction

With the breakthrough developments in connection and information technologies, tremendous smart sensors are adopted in vehicular networks to characterize the road traffic and environmental surroundings by the continuously generated sensing data (SD). Although a large number of sensors are installed on the roadside infrastructures, the high maintenance cost and information access right limit the extent to which the sensing information can be shared among individual users [2]. Therefore, on-board sensors that can exploit the high vehicle mobility to provide wide-coverage and fine grained sensing services are becoming important sources of sensing data [1], [3]–[6]. To stupendously benefit the secure and efficient road services, such as autonomous driving, remote fleet management, and intersection collision warning, vehicles need to get access to specific SD in real time [5], [7], [8]. Studies show that the overall sensor market will become the fastest growing segment in automotive componentry [9], which will lead to a rapid growth of the SD volume. The data traffic is estimated to be roughly 0.6 EB in every month of 2020 and will increase to 9.4 EB by 2030 [10]. Given the tremendous data volume, a real-time search that aims at retrieving requested data promptly [11] is crucial to reduce response time for drivers and enhance the traffic safety [12].

Due to the dynamic road environments and busy daily trips, there are massive connections and search interactions in vehicular networks, which can impose excessive pressure on backhaul links and degrade the service quality significantly [13]. To this end, edge caching, which is a well-recognized solution to relieve the traffic burden for the cloud server, has been explored to support the real-time search by deploying contents in close proximity to target users [14]. Nowadays, the majority of vehicles are equipped with onboard sensors to gather the surrounding information [15], but the remote traffic status used for navigation needs to be requested online. With various travel destinations or moving behaviors, there will be a huge difference among the requests. Due to the growing data volume and limited edge caching size, the caching-assisted search can still sustain data missing and response failures for some requests. To this end, a real-time search-driven caching (RSC) paradigm is designed in this article, where the caching edges are considered to work

cooperatively for enhancing the search efficiency and resource utilization.

In the literature, extensive studies are conducted on the edge caching to improve data access efficacy [13], [14], [16], [17]. The users can be served by one or multiple cooperative nodes [14], [16], and the cooperative caches are normally assumed as neighboring vehicles or roadside units (RSUs) [18], [19]. To minimize the data downloading delay, most caching schemes are determined on the basis of given popularity. However, as the popularity is predefined or needs to be predicted with sufficient historical requests [20], [21], the popularity of SD that have various types and application cases can hardly be practically achieved. Therefore, moderate decoupling of caching decisions and popularity information is critical for data search scenarios. On the other hand, the data need to be uploaded from distributed sensing devices before cached in the edges. Given various content preferences, enormous queries, and data volume, the data cannot always be ensured to be cached in the proximity of target users. Hence, traditional traversal mechanisms, where the search query will be forwarded to the local edge, neighbor cache servers, and content provider in turn, are not applicable for requesting SD. To support the distributed caching decisions and search queries, the cloud server is introduced in our work to manage an index for accelerating the retrieval process.

Compared with infotainment contents, such as video and game profiles, SD is time sensitive and only valid within a certain time window [21]–[24]. To this end, to satisfy real-time search requirements for the temporally variant data, the cooperative caching scheme needs to be carefully designed for ensuring data freshness and prompt response at the same time, which is quite challenging due to the following reasons. First, with limited knowledge of the search requesting features, the double requirements in "real time" include the optimization of both the validity of cached data and response delay, which renders the problem complicated. Second, for the search-driven caching, SD has to be uploaded to edge servers in advance, then be retrieved and downloaded for requesting users. Thus, the delay performance can be affected by both the searching and uplink/downlink transmissions during the process, which should be deeply explored in the scheme design. Third, with limited edge cache size and the dynamic wireless channels [5], when and where to cache the up-to-date generated data is nontrivial because 1) immediate data transmission cannot always be guaranteed especially when the communication channels are weak and 2) the expected retrieval time could vary with different caching locations for each data item. Finally, the caching decisions are generally optimized based on content popularity, while in vehicular networks, the historical search requests cannot always be collected due to privacy protection. Hence, the caching edges should be designed to deal with different cases adaptively.

In this work, we delve into the SD caching design to address the above-mentioned challenges via fully unleashing the potential of cooperative edge caching. Our objective is to increase the result validity and search speed for the widely distributed SD requesters. Therefore, the association between SD and interested users is first analyzed in this article, based on which we propose a hierarchical search framework for the search service in vehicular networks. With the objective of enabling delay-sensitive data search, a utility model is then devised to quantify the expected response delay and freshness for cached data, as these two factors are crucial in searching performance evaluation. Next, the RSC problem is formulated within the constraints of transmission capability and cache storage budget, aiming at maximizing the long-term search reward of large-scaled SD. To study and utilize the potential data searching rules for optimizing caching decisions, we propose a deep-reinforcement-learning-based caching (DRLC) method to solve the problem. In specific, an action transition module is designed to reduce the computational complexity by decreasing the caching selection space. Finally, extensive simulations are conducted to demonstrate the efficacy of the proposed caching method, and results show that it can perfectly support the real-time search requirements as expected.

Our major contributions can be highlighted as follows.

1) We propose an RSC paradigm based on cooperative edges. The RSC paradigm can facilitate the efficient search of SD, which is essential for future vehicular networks but rarely considered in the literature.
2) We propose a hierarchical search framework for cooperative edges in vehicular networks, in which the caching servers are indexed according to their spatial distribution, and the cloud can quickly forward and search the queries accordingly.
3) We devise a search utility (SU) model to quantify the contribution of caching decisions for real-time data search. Then, the DRLC method is proposed to solve the RSC problem in an intelligent and low-complexity way.

The remainder of this article is organized as follows. Section II briefly reviews the related work of content request features and caching problems in vehicular networks. The system model, search framework, and utility model are illustrated in Section III. The RSC problem and DRLC method are elaborated in Section IV. In Section V, the proposed methods are evaluated through abundant simulations, and finally, we conclude this article and direct our future work in Section VI.

II. Related Work

We review the related work in two categories, i.e., real-time data request and cooperative edge caching.

A. Real-Time Data Request

Information exchange among vehicles, communication infrastructures, and the Internet is the basis of providing emerging services in future vehicular networks, ranging from collision avoidance, remote vehicle diagnosis in self-driving to 3-D environment modeling in navigation [5]. In the literature, there are two main vehicular communications modes: 1) vehicle-to-infrastructure (V2I) and 2) vehicle-to-vehicle (V2V). As one of the most important restrictions to measure the Quality of Service (QoS) [11], the transmission delay is affected by the dynamic communication environment significantly [17], thus it needs to be modeled and analyzed

according to different scenarios. For example, Yang *et al.* [18] considered a hybrid data dissemination model to acquire the demanded data from the edge or nearby neighboring vehicles for automated driving assistance, and the deadlines of acquiring data were adopted to analyze the delay in algorithm design. Liu *et al.* [25] proposed a distributed algorithm to minimize the average download delay, addressing the content placement, and transmission problem jointly. A similar joint problem was discussed in [26], where the user's requests can be forwarded either to the edge caches or congested backend server directly. Hence, two delay models were constructed to assess the different transmitting associations with traffic congestion. Nevertheless, these works mostly focused on the downlink data transmission delay, which is insufficient to reflect the freshness of time-sensitive sensing information.

A myriad of current studies assumed that the request patterns follow the Poisson process or Markov process, which however, is not applicable to real traffics [21]. Therefore, the real-world content popularity requires further investigation since it is an indispensable factor for request abstraction and content deployment. In existing studies, content popularity is widely considered for entertainment files such as music or movies, owing to the abundant data resources [22], [27], while the proposed models for them may not be suitable to characterize the temporal SD. For example, the generated location is always included as a dimension of sensing features, thus the request distance from the data has a stronger impact on the interest than that from the caching place of entertainment content. Due to the dynamic nature of traffic systems, existing studies are concerned more about the temporal analysis for SD in vehicular networks, such as traffic information and location-based service information. In [7], to jointly enhance the data quality and delivery ratio, a temporal data update and dissemination problem was formulated. In [28], a data validity model was devised at first, then the authors exploited the tradeoff between traffic load and data validity for caching SD at network routers. However, these analyses were conducted under given requests, while seldom considering the highly dynamic and geographically distributed characteristics of requests in data search applications.

B. Cooperative Edge Caching

Cooperative caching has been studied recently to enlarge the set of cached data [13], [14], [16], [17]. Most of these works focused on minimizing downloading delay of cached files, and a user can be served by either one or multiple nodes together [14], [16]. Accordingly, Liu *et al.* [25] considered multiple candidate transmission schemes for different users and designed a cache placement strategy to minimize the average downloading delay. Aiming at optimizing the total reduced backhaul traffic in the long term, Lyu *et al.* [13] formulated a WiFi caching gain maximization problem. Zhang *et al.* [14] considered a user-centric mobile network and implemented cooperative edge caching by solving two fundamental problems, namely, content placement and edge clustering. Hui *et al.* [17] studied a heterogeneous vehicular network consisting of cellular base stations and roadside units. In specific, the base station was considered as the cloud server, and the content delivery problem was studied to improve utilities of participants (i.e., cellular base station, vehicles, etc) in heterogeneous vehicular networks. Apart from data downloading, the request forwarding is also an essential part of the content retrieval process for time-sensitive data, which is designed differently in various cooperative architectures. In the network model proposed by Xia *et al.* [16], the fog access points (F-AP) were clustered to cooperate with each other. If the requested content was not cached in the serving F-AP, the request would then be transmitted to the cluster head. The cluster head can identify the F-AP which had cached the requested content and coordinate the cooperation between two F-APs. Dehghan *et al.* [26] assumed that when the requested content was not cached in the edge cache, it needed to be retrieved and downloaded from the backend server, resulting in a long content access delay. Therefore, to facilitate real-time data search services, the caching placement and request forwarding problems should be jointly designed under a cooperative architecture. Given the large volume and distributed generation locations of temporal SD, balancing the tradeoff between data updating and downloading delay becomes crucial in the delay analysis, and the impact of data retrieving pattern should be well studied.

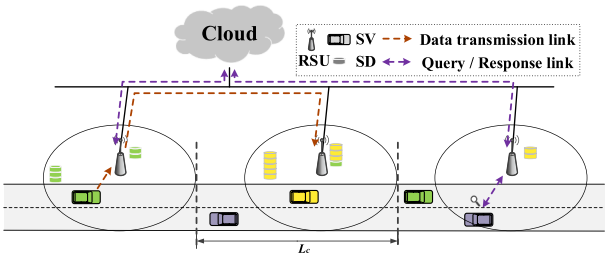

Fig. 1. Illustration for cooperative SD caching.

III. System Description

In this section, we present the system model of caching SD in vehicular networks and describe the proposed SU model. The key notations used in this article are listed in Table I.

A. Network Model

To facilitate intelligent services, such as accident detection and road maintenance, modern vehicles are usually equipped with various sensors (speedometer, radar, pressure sensor, etc.) to collect vehicular and environmental information. Fig. 1 illustrates a traffic monitoring scenario, where SD (traffic status, road condition, noise decibel, etc.) can be gathered by the onboard sensors, and we name the vehicles as sensing vehicles (SVs). RSUs are regarded as edge nodes, which can provide both network and caching services for vehicles. In specific, the RSU is named as intended cache for data prepared to be uploaded in. In this scenario, the gathered SD should be transmitted to intended cache via V2I communication. Let $\mathbb{R}$ be the set of all cache nodes, and the capacity of each cache server is a preset value, denoted by $\mathcal{C}_i$ of node R_i. Particularly, the cloud server, denoted by R_0, is introduced as the backup cache, providing storage for contents that surpass the edge cache size to avoid data omission.

TABLE I
NOTATION SUMMARY

Notation	Description
$\mathbb{R}$	the set of all cache nodes
R_i	the i_{th} cache node
$R_i^{\mathcal{N}}$	neighboring set of the ith cache
$\mathcal{C}_i$	caching capacity of cache i
$\mathbb{S}$	the set of all sensing data items
$\mathbb{S}_i$	sensing data set gathered around R_i
M_i	the number of sensing data items in $\mathbb{S}_i$
c_d	the size of a sensing data item
s_{ij}	sensing item of segment j indexed by R_i
s_{ij}^k	caching item s_{ij} at R_k
N_s	total number of data items in $\mathbb{S}$
Δ	data updating interval
S_{ij}^k	the SU value of s_{ij}^k
$\mathcal{F}_{ij}^k$	the freshness of s_{ij}^k
C_{ij}^k	the expected search space of s_{ij}^k
D_{ij}^k	delay for caching s_{ij} at R_k
$D_{H_{ij}}^k$	holding delay for caching s_{ij}^k
$D_{T_{ij}}^k$	upload transmission delay for caching s_{ij}^k
L_c	the covered road length for an RSU
$\overline{r_i}$	the average transmission rate of R_i
p_{ij}^q	the probability of searching s_{ij} at cache q
$\mathcal{S}_\tau$	the data search state at time slot τ
$\mathcal{A}_\tau$	the set of feasible caching actions at τ
$\mathcal{A}_\tau^h$	the hidden action set
$\varphi(\cdot)$	the expected reward value
$V(\cdot)$	the expected long-term search reward
$Q(\cdot)$	the Q-value function

Considering the scenario with insufficient edge caching size, N RSU nodes work cooperatively with the cloud server, where the allocated caching size is $\mathcal{C} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_N\}$. Deployed along the road, RSUs are connected with the backbone network and can communicate through backhaul links with the cloud, i.e., the RSU-to-Cloud (R2C) link. Without loss of generality, we assume that RSUs can periodically broadcast heartbeat messages to discover vehicles passing-by, then serve them through the Vehicle-to-RSU (V2R) links.

When the queried data is not cached in the serving edge, extra route search and request forwarding delay will increase dramatically if the same queries are triggered repeatedly at different locations. Caching the most popular content in multiple edges is an intuitive solution to address this issue, which, however, is prohibitive due to the insufficient edge caching storage and laborious exploration of time-varying popularity of SD. Therefore, to reduce excessive interaction overhead, in our proposed cooperative caching model, the cloud takes charge of the forwarding of data requests initiated from multiple locations via a cache index to speed up the search of request forwarding path, as discussed in the following section.

B. Searching Framework

A unique feature of SD search is that the searched contents are highly correlated to the spatial and temporal information. To devise an effective search framework, we first investigate the spatiotemporal characteristics of search preferences. Fig. 2(a) and (b) shows that people tend to pay more attention to surrounding information in the most recent years. According to statistics in Fig. 2(c), the popularity decreases with the distance between search queries and special events, i.e., events are cared more by local residents. As SD is the most direct reflection of specific events, it is reasonable to assume that the data is queried more by surrounding users based on these observations.

Generally, SD should be transmitted to a nearest edge for reducing the upload delay and ensuring data freshness. To this end, the monitoring space will be divided as shown in Fig. 1, where each road segment with a length of L_c will be assigned with a caching edge. If all data items are cached according to space segmentation, then they can be retrieved from the assigned edges directly. However, this can hardly be achieved due to the limited caching storage and highly dynamic data requests. Therefore, the cooperative caching is proposed where some SD can be uploaded to the other edges through the cloud. Specifically, to avoid data omission and alleviate the traffic burden for backhaul links at the same time, the cloud server will provide caching service as a supplement for the edge nodes, and only hold the uploaded omitting data.

For data not cached in the assigned edges, a hierarchical search framework is proposed to speed up data retrieval. In the existing literature, the request will be first forwarded to the nearest serving edge, when it is not satisfied by the local cache, it will be forwarded to the neighboring edges, then be traversed among all caches by the cloud or resort to the Internet for content access, as shown in Fig. 3(a). However, as there are distributed search requests, the data cannot be ensured to be cached near all serving edges of the requesters. On the contrary, the data should be cached close to the sensing locations for enhancing data freshness. Therefore, the cloud will maintain a cache index of the space segmentation rule in this article and take charge of the index and data management. As such, when the serving cache does not contain the requested data, it will next be forwarded to the cloud server. In this way, the retrieving space and request forwarding paths can be reduced under the direction of the index. Fig. 3(b) illustrates the following hierarchical search process: the cloud will first forward requests to the indexed cache, i.e., the assigned edge for SD according to space segmentation, then search the neighbors of indexed cache if the data are not cached as planned. For the traffic monitoring in vehicular networks, we consider the two adjacent nodes of R_i as its neighbors, denoted by $R_i^{\mathcal{N}} = \{R_{i-1}, R_{i+1}\}$. Particularly, in a noncircular scenario, we have $R_1^{\mathcal{N}} = \{R_2\}$, $R_N^{\mathcal{N}} = \{R_{N-1}\}$. Next, the request will be broadcasted to all the other caches and transmitted back to the cloud server for searching the results.

C. Utility Model

To facilitate an effective caching scheme for the monitoring information, the search gain expectation of caching solutions with different caching locations should be evaluated first. Accordingly, the sensing space is divided into several segments as shown in Fig. 4. When a vehicle passes by a segment,

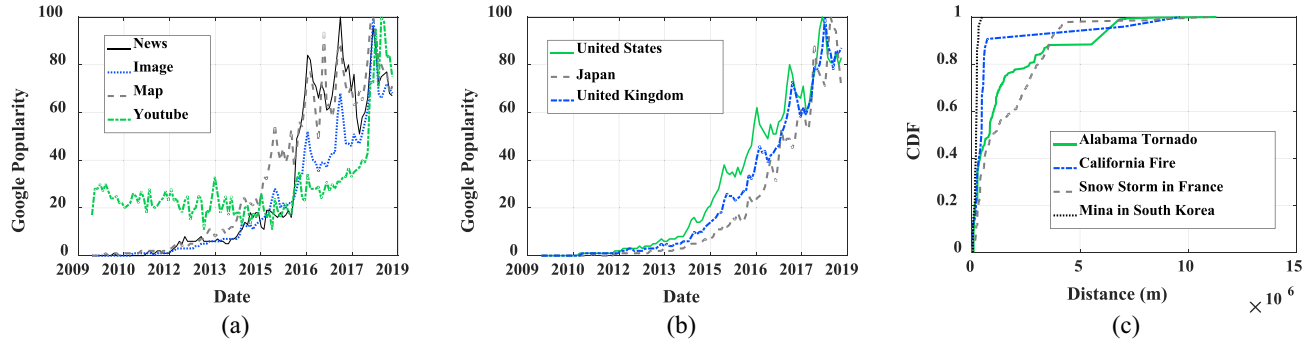

Fig. 2. Relationship between the search popularity and request locations in Google. (a) Popularity of "nearby" search queries in different applications. (b) Popularity of nearby search queries in different countries. (c) CDF for popularity of emergent events versus distance between search queries and event location.

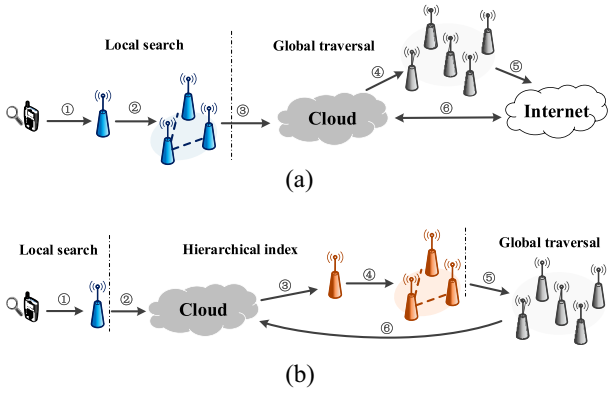

Fig. 3. Data retrieval in cooperative edge caching. (a) Existing search method: ① search at the serving edge; ② search at neighbors of the serving edge; ③ forward to the cloud server; ④ traverse all the edges for searching; ⑤ go through the Internet; and ⑥ search and retrieve data from the cloud server. (b) Proposed hierarchical search framework: ① search at the serving edge; ② forward to the cloud server; ③ search at the indexed cache; ④ search at neighbors of the indexed edge; ⑤ traverse all the other edges for searching; and ⑥ search the cloud server.

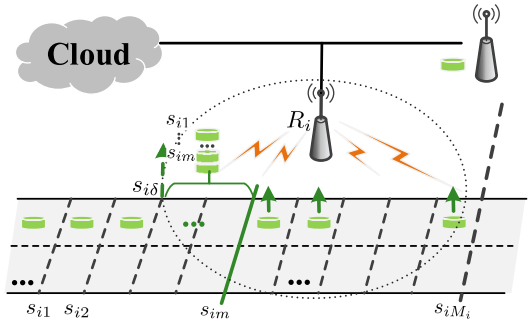

Fig. 4. Data uploading in the sensing area indexed by R_i.

it can generate a data item with the equipped sensors. Let the number of data items indexed by RSU R_i be M_i, then the total number of SD would be $N_s = \sum_{i=1}^{N} M_i$. s_{ij} denotes the SD gathered at segment j indexed by R_i. If it is transmitted to cache k, s_{ij} will be marked as s_{ij}^k. Denoting the set of all SD as ground data set $\mathbb{S}$, which can be partitioned into N disjoint sets, i.e., $\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_N$, where $\mathbb{S}_i$ represents the set of SD gathered around R_i, and we have $\mathbb{S}_i = \{s_{i1}, s_{i2}, \ldots, s_{iM_i}\}$.

Based on the proposed search framework, there are four levels of caches in the cooperative edge caching system, including the indexed edge, neighbors of the indexed edge, other edges, and the cloud server. To adopt the hierarchical index in assisting real-time information search, we define the following evaluation model.

Definition 1 (Search Utility): The gain expectation for real-time search services when caching SD at a specific location.

To respond with fresh data in a short time, the data freshness and search space become two crucial restrictions to quantify the SU. The former value can be measured by the freshness loss as

$$\mathcal{F} = \frac{D}{\Delta} \tag{1}$$

where D is the total delay between data generation and the time when it is accessed by request users, and $\Delta = l/v$ refers to the data updating interval. l is the distance between SVs of the same sensing category, and v represents the moving speed.

As one of the most widely adopted index structures in mainstream database system, B-tree is a self-balancing data structure that generalizes binary search tree, maintains sorted data, and allows search in logarithmic time. The search complexity of B-tree-based retrieval can be measured by the expected search space, i.e., $O(\log C)$, where C represents the space size. Therefore, the SU can be expressed as

$$S_{ij}^k = \frac{1}{\mathcal{F}_{ij}^k \log_2\left(1 + \mathcal{C}_{ij}^k\right)} \tag{2}$$

where the formation of data freshness $\mathcal{F}_{ij}^k$ and search space $\mathcal{C}_{ij}^k$ change with s_{ij}^k. Based on the hierarchical indexing framework, the search space can be measured by the total capacity of all retrieved caches, i.e.,

$$\mathcal{C}_{ij}^k = \begin{cases} \mathcal{C}_i, & k = i \\ \mathcal{C}_i + \sum_{r \in R_i^{\mathcal{N}}} \mathcal{C}_r, & k \in R_i^{\mathcal{N}} \\ \sum_{r \in \mathbb{R}} \mathcal{C}_r - \mathcal{C}_0, & k > 0 \text{ and } k \in \mathbb{R} \backslash R_i^{\mathcal{N}} \\ \sum_{r \in \mathbb{R}} \mathcal{C}_r, & k = 0. \end{cases} \tag{3}$$

For search requesters, the data access time mainly depends on the transmission delay and request generation time. However, the search requests could be generated randomly and arrive in the caches at any time, thus the delay between the generation of s_{ij}^k and the time when it arrives the cache is used to measure the expected data caching freshness, denoted

by D_{ij}^k. SD transmission will be initiated only when 1) the vehicles are covered by an RSU and 2) the data gathered before have all been uploaded. Accordingly, several data items need to be held for a while, thus D_{ij}^k can be calculated as

$$D_{ij}^k = D_{H_{ij}}^k + D_{T_{ij}}^k \tag{4}$$

where $D_{H_{ij}}^k$ and $D_{T_{ij}}^k$ refer to the holding delay and SD upload transmission delay, respectively.

Basically, the transmission rate depends on the distance between the vehicle and the receiving RSU [29]. In general, the coverage of an RSU is divided into several zones based on 802.11p, where vehicles have different transmission rates in different zones. In this article, we divide the coverage of R_i into O zones and denote the set of zones by $\mathcal{Z}_i = \{z_1, z_2, \ldots, z_O\}$. For zone $z_\alpha (z_\alpha \in \mathcal{Z}_i)$, the transmission rate of the V2R link is $r_{z\alpha}$. Denoting the average transmission rate under the coverage of R_i as $\overline{r_i}$, we have

$$\overline{r_i} = \sum_{z_\alpha \in \mathcal{Z}_i} \frac{d_{z_\alpha} r_{z_\alpha}}{2Rc_i} \tag{5}$$

where Rc_i is the coverage radius of RSU i and d_{z_α} is the length of zone z_α.

Supposing that the vehicle begins to transmit data to the serving edge once it enters the covered area, then the total transmitted data size is $C = \overline{r_i} t$, where t refers to the communication time duration. s_{ij}^k can be transmitted only if the data packets gathered before have already been sent out. Let the data size generated at a single segment be c_d, then before transmitting s_{ij}^k, the total transmitted data size is $(j-1)c_d$. Supposing that the vehicle begins to enter the communication coverage after generating δ items, then the total uploaded data size before transmitting s_{ij}^k can also be expressed as

$$C = \overline{r_i}\left(D_{H_{ij}}^k + t_{ij} - t_{i\delta}\right) \tag{6}$$

where t_{ij} and $t_{i\delta}$ represent the generating time of s_{ij} and $s_{i\delta}$, respectively. Notice that when $C > (j-1)c_d$, the SD can be uploaded in real time, which means that it will no longer need to be held. As shown in Fig. 4, let the first real-time uploading data be s_{im}, then we have $t_{im} = [(m-1)c_d]/(\overline{r_i}) + t_{i\delta}$, and the holding delay can be expressed as

$$D_{H_{ij}}^k = \begin{cases} \frac{(j-1)c_d}{\overline{r_i}} + t_{i\delta} - t_{ij}, & j \le m \\ 0, & j > m. \end{cases} \tag{7}$$

Let r_{ci} be the transmission rate between the cloud server and R_i, thus the data transmission delay can be expressed as

$$D_{T_{ij}}^k = \begin{cases} c_d\left(\frac{1}{\overline{r_i}} + \frac{1}{r_{ci}}\right), & k = 0 \\ \frac{c_d}{\overline{r_i}}, & i = k \\ c_d\left(\frac{1}{\overline{r_i}} + \sum_{p\in\{i,k\}} \frac{1}{r_{cp}}\right), & \text{otherwise.} \end{cases} \tag{8}$$

IV. Problem Formulation and Solution

In this section, we will first formulate the RSC problem, then model the problem to be a Markov decision process (MDP) and design the DRLC method for solving it.

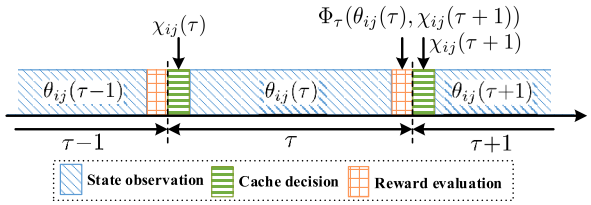

Fig. 5. Working process of caches across time.

A. RSC Problem

Generally, SD is generated continuously and can be requested repeatedly by users at different locations, thus cooperative caching is proposed to alleviate the traffic burden and enlarge the cache size. In this way, the RSC problem can be cast as follows.

Definition 2 (Real-Time Search-Driven Caching): Given a set of cooperative caches and a set of SD that would be generated in the near future, how to allocate caches for the data items so that the expected response delay of searching fresh data can be minimized?

Intuitively, the solution to the RSC problem aims to provide a caching decision that fulfills the real-time search requirement. Therefore, prefetching the real-time distribution of search requests is critical in caching determination, and the historical request features should first be learned. In general, the search history of a data item s_{ij} can be described by the request distribution during a given time period, which can be measured by

$$p_{ij}^q = \frac{Nr_{ij}^q}{\sum_{R_q \in \mathbb{R}} Nr_{ij}^q} \tag{9}$$

where Nr_{ij}^q is the requested times of s_{ij} at cache q, and p_{ij}^q is the corresponding probability.

To study the requesting features from real-time information and provide online caching decisions, the RSC problem is modeled as an MDP, which can be defined by a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \Phi(\theta, \chi)\}$.

1) $\mathcal{S}$ is the state set, which is expressed by the historical request information for all SD items, thus we have $\mathcal{S} := \{\theta_{ij} | \theta_{ij} = [q, p_{ij}^q] \ \forall s_{ij} \in \mathbb{S}\}$, in special, $q = \arg\max p_{ij}^q$.
2) $\mathcal{A} := \{\chi_{ij} | \chi_{ij} \in \{0,1\}^N, \|\chi_{ij}\|_1 \le 1 \ \forall s_{ij} \in \mathbb{S}\}$ is the set of feasible caching actions, where χ_{ij} is an $N \times 1$ binary vector indicating the caching action of s_{ij}. For clarification, we use x_{ij}^q to denote the qth entry of χ_{ij}, which indicates whether R_q decides to cache data s_{ij}.
3) $\varphi(\theta, \chi)$ is the reward function that captures the expected gain of supporting real-time search if taking action χ under state θ.

The process for a cache edge to work across time is described in Fig. 5, where the working period can be divided into three phases, i.e., *state observation*, *cache decision*, and *reward evaluation*. In this way, a cache can take actions based on the experienced states to improve the search efficiency in the next time slot. Denote $\mathcal{S}_\tau$ as the set of request state at time slot τ, and $\theta_{ij}(\tau) = [q, p_{ij}^q(\tau)]$ as the state of s_{ij}, upon

taking action $\chi_{ij}(\tau)$ at the cache decision phase, the reward $\varphi_\tau(\theta_{ij}(\tau-1), \chi_{ij}(\tau)|\theta_{ij}(\tau))$ can then be obtained.

Denote $\pi : \mathcal{S} \to \mathcal{A}$ as the policy function, which maps any state $\theta \in \mathcal{S}$ to action $\chi \in \mathcal{A}$. Under policy π, the caching decision at $\tau+1$ will be carried out via $\mathcal{A}_{\tau+1} = \pi(\mathcal{S}_\tau)$. In the MDP framework, the reward $\Phi(\mathcal{S}, \mathcal{A}) = \sum_{s_{ij}\in\mathbb{S}} \varphi_{ij}(\theta_{ij}, \chi_{ij})$ is used to quantify the real-time search effectiveness of a caching scheme.

As the edge caching is driven by real-time SD search, the search response delay and expected utility are jointly considered to formulate the cache reward function. However, the prediction of accurate and real-time request distribution is quite difficult. Due to the computational time and discretized request state, we can hardly get the exact request timestamp or retrieval delay. Hence, for s_{ij} cached at R_k and requested at cache R_q, where $p_{ij}^q(\tau)$ is the maximum requested probability of $\theta_{ij}(\tau)$, the response delay is chosen to evaluate the search efficacy. Specifically, the delay is defined as the data downloading time from the cache to the requester, and denoted by $D_{Q_{ij}}^{kq}$ here. Since the downlink and uplink transmission rates are considered to be the same in this article, we have $D_{Q_{ij}}^{kq} = D_{T_{qj}}^{k}$. But for the requests generated at other locations, the cache gain would be evaluated by S_{ij}^q. To further unify the expression, the total cache reward of action $\mathcal{A}_\tau$ can be calculated as

$$\Phi_\tau(\mathcal{S}_{\tau-1}, \mathcal{A}_\tau|\mathcal{S}_\tau) = \sum_{s_{ij}\in\mathbb{S}} p_{ij}^q(\tau)\frac{\Delta}{D_{Q_{ij}}^{kq}} + \Big(1 - p_{ij}^q(\tau)\Big)S_{ij}^q. \quad (10)$$

Then, the expected long-term reward can be expressed by

$$V(\mathcal{S}_\tau) = \mathbb{E}\left[\sum_{\tau=1}^{\infty} \gamma^{\tau-1}\Phi_\tau(\mathcal{S}_\tau, \mathcal{A}_{\tau+1})\right] \quad (11)$$

where $\gamma \in [0, 1]$ is a discount factor that determines the impact of the current action on future rewards. Considering the conditional independent characteristic among the dynamically distributed and time-varying requests, the long-term reward is adopted to quantify the caching efficacy of real-time data search. To this end, the RSC problem aims to find an optimal policy π^* shown as

$$\pi^* = \arg\max_{\pi\in\Pi} V(\mathcal{S}_\tau) \quad (12)$$

where Π denotes the set of all possible policies. Particularly, data copies are not considered in this scenario to improve the monitoring coverage and data diversity, where each data item is assumed to be cached only once. Therefore, combining the restrictions of limited cache space and data updating frequency, the RSC problem is expressed as

$$\max \ \ V(\mathcal{S}_\tau) \ \ \forall\tau \quad (13)$$

$$\text{s.t.} \ \ \left\|\chi^k(\tau)\right\|_1 \le C_k/c_d \ \ \forall k \in \mathbb{R} \ \ \forall\tau \quad (13a)$$

$$\left\|\chi_{ij}(\tau)\right\|_1 \le 1 \ \ \forall s_{ij} \in \mathbb{S} \ \ \forall\tau \quad (13b)$$

$$x_{ij}^k(\tau)D_{ij}^k \le \Delta \ \ \forall s_{ij} \in \mathbb{S} \ \ \forall k \in \mathbb{R} \ \ \forall\tau \quad (13c)$$

$$\pi \in \Pi \quad (13d)$$

where χ^k is the set of all indicators for data actions in R_k. In particular, (13a) and (13b) ensure that the caching action is limited by the storage capacity, (13c) guarantees the validity of cached data, and the above problem is a sequential decision-making problem with time-varying parameters.

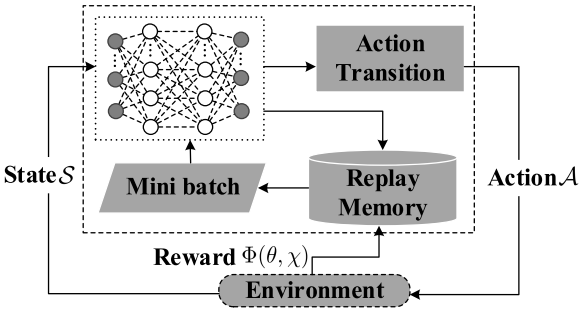

Fig. 6. Illustration of the DRLC training process.

Intuitively, the feasibility of a given policy can be verified in polynomial time by checking whether it satisfies the constraints in (13), which means that the RSC problem is NP. Following the demonstration of [30, Th. 1], the RSC problem can be proved to be NP-hard by using a reduction from the multiple-choice knapsack problem, which is known to be NP-complete and usually comes with an exponential computational complexity solution. As an important evaluation criterion of practical system, especially for the short lifetime SD, computational complexity is a primary concern that needs to be reduced.

B. DRLC Method

On the basis of the MDP formulation, the intelligent DRLC method is designed to lower the computational complexity of solving the RSC problem. One of the most critical components in DRLC is the Q-value table $Q(\mathcal{S}, \mathcal{A})$, the dimension of which is $|\mathcal{S}| \times |\mathcal{A}|$. Hence, when the number of edges and sensing items are large-scale in the network scenario, the dimension of state space will be very high. To address this problem, we introduce the hidden action $\mathcal{A}^h := \{\chi_{ij}^h | \chi_{ij}^h \in \{0, 1\}, \|\chi_{ij}^h\|_1 \le 1 \ \forall s_{ij} \in \mathbb{S} \ \forall\tau\}$ into DRLC, reducing the dimension to $|\mathcal{S}| \times 2$. In particular, $\chi_{ij}^h = 1$ indicates that the s_{ij} should be cached at R_q, where $q = \arg\max p_{ij}^q$. Otherwise, the greedy-based algorithm should be adopted to decide the caching place for s_{ij}. As shown in Fig. 6, the greedy-based algorithm is implemented within the action transition module and transforms the hidden action $\mathcal{A}_\tau^h$ into $\mathcal{A}_\tau$.

Accordingly, the Q-value function will be expressed through the relationship between value function and hidden-action function

$$Q\Big(\mathcal{S}_\tau, \mathcal{A}_\tau^h; \omega_\tau\Big) = Q\Big(\mathcal{S}_\tau, \mathcal{A}_\tau^h; \omega_\tau\Big) + \alpha\Bigg(\Phi_\tau + \gamma \max_{\mathcal{A}^h} Q\Big(\mathcal{S}_{\tau+1}, \mathcal{A}_{\tau+1}^h; \widehat{\omega_\tau}\Big) - Q\Big(\mathcal{S}_\tau, \mathcal{A}_\tau^h; \omega_\tau\Big)\Bigg) \quad (14)$$

where $\alpha \in [0, 1)$ is an introduced parameter, ω_τ is a parameter of the evaluation network, and $\widehat{\omega_\tau}$ is a parameter of the target network. Two networks are adopted to prevent overfitting.

Algorithm 1 DRLC Training Process

Initialize $E, I_n, \epsilon, \mathcal{S}_0 \coloneqq \{\theta_{ij}|\theta_{ij} = [0,0], \forall s_{ij} \in \mathbb{S}\}$,
$\mathcal{A}^h \coloneqq \{\chi_{ij}^h|\chi_{ij}^h \in \{0\}, \|\chi_{ij}^h\|_1 \leq 1, \forall s_{ij} \in \mathbb{S}, \forall \tau\}$,
$\mathcal{A} \coloneqq \{\chi_{ij}|\chi_{ij} = \{0\}^N, \forall s_{ij} \in \mathbb{S}, \forall \tau\}$.
1: **For** episode $e = 1$ to E **do**
2: **For** iteration $i = 1$ to I_n **do**
3: Collect the requests and transform into state $\mathcal{S}_\tau$.
4: Choose a random probability ξ.
5: **If** $\xi < \epsilon$
6: Randomly select an action $\chi^h(\tau) \in \mathcal{A}^h$.
7: **Else**
8: Select action with $\chi^h(\tau) = \arg\max_{\chi \in \mathcal{A}^h} Q(\theta(\tau-1), \chi; \omega_{\tau-1})$.
9: **End if**
****** Action Transition Start ******
10: $\mathbb{S}^{ht} \coloneqq \{(k_{mq}, s_{ij})|x_{ij}^h = 1, \forall x_{ij}^h \in \chi^h(\tau)\}$, $\mathbb{S}^{hf} \coloneqq \{s_{ij}|x_{ij}^h = 0, \forall x_{ij}^h \in \chi^h(\tau)\}$.
11: **Do** $s_x^y = \arg\max_{s_{ij}^k \in \mathbb{S}^{ht}} S_{ij}^k$
12: **If** $D_{ij}^k \leq \Delta$ **and** $\|\chi^y(\tau)\|_1 + 1 \leq \mathcal{C}_k/c_d$
13: $x_{ij}^{k_{mq}} = 1$,
14: $\mathbb{S}^{ht} \leftarrow \mathbb{S}^{ht} \backslash \{s_{ij}^{k_{mq}}\}$.
15: **Until** $S^{ht} = \varnothing$
16: **Do** $s_x^y = \arg\max_{s_{ij}^k \in \mathbb{S}^{hf}} S_{ij}^k$
17: **If** $D_{ij}^k \leq \Delta$ **and** $\|\chi^y(\tau)\|_1 + 1 \leq \mathcal{C}_k/c_d$
18: $x_{ij}^k = 1$,
19: $\mathbb{S}^{hf} \leftarrow \mathbb{S}^{hf} \backslash \{s_{ij}^m : \forall R_m \in \mathbb{R}\}$.
20: **Until** $S^{hf} = \varnothing$
****** Action Transition End ******
21: Execute action $\chi(\tau)$ and observe request state θ_τ.
22: Calculate the reward $\Phi_\tau(\theta(\tau-1), \chi(\tau)|\theta(\tau))$.
23: Add $[\mathcal{S}_\tau, \mathcal{A}_\tau, \Phi_\tau, \mathcal{S}_{\tau+1}]$ into the tuple $\mathcal{M}_\tau$.
24: Randomly sample a mini-batch $\widetilde{\mathcal{M}_\tau}$ from $\mathcal{M}_\tau$.
25: Update the Q network ω_τ with $\nabla(F(\omega_\tau))$.
26: **End for**
27: **End for**

Specifically, the maximum objective can be temporarily fixed in the target network, thus the relevance between action selection and model training can be reduced. The gradient descent method is utilized to update the parameters, where the loss function is expressed as

$$F(\omega_\tau) = \sum_{(\mathcal{S}_\tau, \mathcal{A}_\tau) \in \widetilde{\mathcal{M}_\tau}} \left(y_i - Q\left(\mathcal{S}_i, \mathcal{A}_i^h; \omega_i\right)\right)^2 \tag{15}$$

in special, we have $y_i = \Phi_i + \gamma \max_{\mathcal{A}^h} Q(\mathcal{S}_{i+1}, \mathcal{A}_{i+1}^h; \widehat{\omega_i})$, and $\widetilde{\mathcal{M}_\tau}$ is a mini-batch of $\mathcal{M}_\tau$. Therefore, the update of ω_τ can be expressed as

$$\omega_{\tau+1} = \omega_\tau - \eta_\tau \nabla(F(\omega_\tau)) \tag{16}$$

where η_τ is the learning rate.

The whole training process of DRLC is described in Algorithm 1. For initialization, the training episode time E, iteration time of each episode I_n, ξ-greedy learning parameter, and the actions will first be set, in particular, all elements in $\mathcal{A}^h$ and $\mathcal{A}$ will be set as 0. In each training episode, the location and content of search requests will first be collected and transformed into state $\mathcal{S}_\tau$. Then, a hidden action will be selected via the evaluation Q-network (lines 4–9), and the caching action will be generated after action transition (lines 10–20). According to different values of the hidden actions, the SD will be separated into two candidate sets, i.e., $\mathbb{S}^{ht}$ and $\mathbb{S}^{hf}$. The former packs all the candidate SD with the target cache together, while $\mathbb{S}^{hf}$ contains all the other data items. However, constrained by the restrictions in (13), we need to decide which items in $\mathbb{S}^{ht}$ are able to be uploaded and cached. Then, if the caching storage is not fully occupied, for items in $\mathbb{S}^{hf}$, we need to determine which of them can be uploaded and where should they be cached. Accordingly, a greedy-based algorithm will be utilized to determine the final caching actions for these two candidate sets in turn. Next, the learning agent can calculate the immediate reward and obtain the next state, and store it into the replay memory buffer (lines 21–23). Finally, a random mini-batch $\mathcal{M}_\tau$ will be sampled to update the evaluation Q-network.

The training complexity of the standard deep Q network is $O(EI_n\bar{n}\bar{n}\bar{m})$, where $\bar{n}$ is the total unit number in the training network [31], [32]. $\bar{m}$ depends on the state and action size, and we have $\bar{m} = |\widetilde{\mathcal{M}_\tau}|$. Without the action transition module, the feasible space of actions is $O(N^{N_s})$, where N_s denotes the total size of the SD set. Obviously, this tremendously high value is unacceptable for practical implementation. Through the design of hidden actions, the computational complexity includes two main parts. The feasible space is decreased to $O(2^{N_s})$. As $N_s = \sum_{i=1}^{N} M_i$, the space can be further decreased by dividing the data into several categories. For example, the data can be distinguished by whether it is covered by an RSU, and the computational space would be $O(2^{2N})$. Since N is a given constant, the complexity will be acceptable. Then, through the greedy-based algorithm, the final action can be selected within $O(N_t \log N_t + N_f \log N_f)$. In particular, N_t and N_f are the sizes of $\mathbb{S}^{ht}$ and $\mathbb{S}^{hf}$, respectively.

V. Performance Evaluation

In this section, we will elaborate simulation parameter settings, explore the training performance of DRLC, and evaluate caching effectiveness for the proposed method.

A. Simulation Settings

The proposed caching method will be verified through three criteria in this section, including the response delay, result freshness, and cache hit ratio (CHR). The vehicular mobility trace of Europarc roundabout, Creteil, France [33] is adopted to construct the simulation scenario. In particular, SVs are randomly selected from the data set, which are assumed to generate data about the surrounding traffics with onboard sensors. Other vehicles are supposed to be search requesters and need to search for the real-time road status that they will pass by in the near future. In other words, we assume that the road status should be accessed in real time for the sake of navigation, thus the search requests and caching actions will be updated in every other 10 s. In this way, the SD needs to be well cached in the RSUs to support real-time search from distributed vehicles. There are six entrances/exits in the roundabout, and one caching edge is assigned for each of them. In specific, 1 km road length of each entrance/exit charged by an

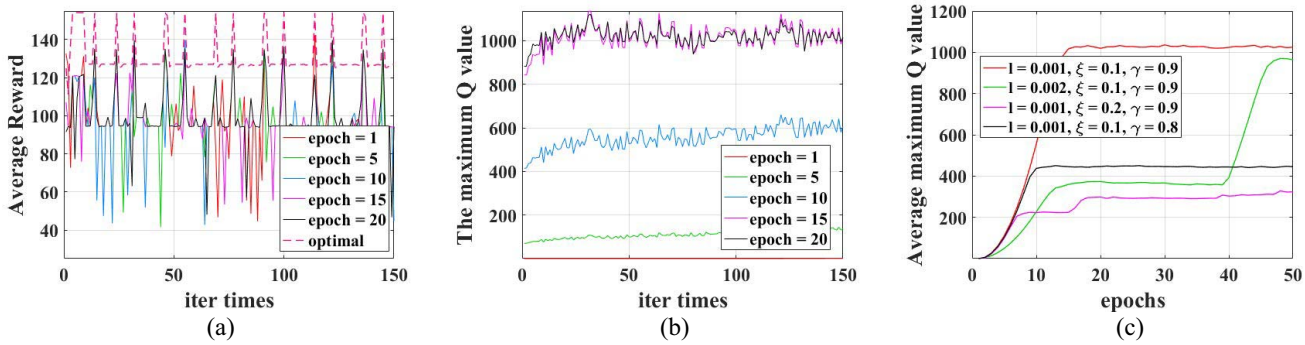

Fig. 7. Training performance of DRLC. (a) Average reward versus iteration time. (b) Maximum Q value versus iteration time. (c) Average maximum Q value versus epoch.

TABLE II
SIMULATION SETTINGS

Cases	No. 1	No. 2	No. 3
N	4	6	6
c_d	$500kb$	$500kb$	$1Mb$
M_i	100		
L_c	$1km$		
r_{z_α}	$\{1, 2, 5.5, 11, 5.5, 2, 1\}$ *Mbps*		

RSU edge is equally segmented into ten parts, thus each SD item reflects the road status in a range of 100 m. Due to the different distances between the cloud server and each RSU, the transmission rates for R2C links are randomly assigned, following a uniform distribution in the range of [30, 100] Mb/s. Similarly, the assigned cache capacity for each RSU follows a uniform distribution in the range of [0.1, 3] Gb. The transmission rate of the fronthaul link, i.e., the V2R link is set with seven different values as in [17]. To evaluate the caching performance, the cooperative edge number and SD size will be changed in three different simulation cases, specifically, the parameter settings are presented in Table II.

To verify the search efficacy of the DRLC method, the response and caching performance are compared for the caching decisions provided by the following benchmark methods.

1) *Greedy SU:* The cached SD will be selected with an objective of maximizing the SU value, where the caching decisions would be the same under all search requests.
2) *Greedy Reward:* SD prepared for search requests in the next time slot will be cached with the objective of maximizing the reward value, i.e., Φ_τ, which is calculated based on current requests.
3) *Local Prior Caching (LPC):* SD will be cached locally according to their generating sequence. Only when the local cache capacity is insufficient will the data select an available cache from the candidate edge sets randomly.
4) *Multiple Copies for Least Recently Used (M-LRU):* Each data item will be cached with multiple copies. Similarly, they would be cached according to the generation time, and the cached copies will be selected based on the greedy algorithm. When the cache size is insufficient, previously cached data that is used least recently will be deleted to make room for new caching data. For simplicity, the number of sensing copies is set as 2 in the simulation.
5) *Multiple Copies for Least Search Utility (M-LSU):* Each SD item will be cached with multiple copies as the same as M-LRU. Differently, cached data with the least SU will be deleted when the capacity is insufficient.

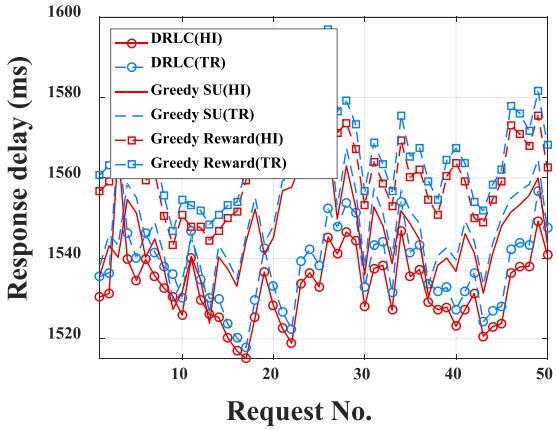

Fig. 8. Comparison for the response delay with different search frameworks.

Generally, the cloud server provides higher reliability for search-driven caching system, but it requires extra construction and interaction costs. Hence, to demonstrate the effectiveness of the hierarchical caching architecture, we design two different cooperative modes.

1) *Edge-Cloud Caching (ecc):* The cloud server will be able to provide caching services for the uploaded SD, where the cloud caching size is assumed to be large enough to cache all the data items.
2) *Edge Caching (ec):* The caching for SD can only be served by edge nodes cooperatively. The cloud server will be in charge of forwarding search requests and managing the whole system.

In particular, the *Greedy SU* and *Greedy Reward* adopt both ec and ecc as our proposed *DRLC* method, while the other benchmark methods are in the ec cooperative mode. Meanwhile, the two search frameworks presented in Fig. 3 are compared to demonstrate the necessity of adopting an index for data searching, where the proposed hierarchical framework is marked as *HI*, and the traditional one is *TR*.

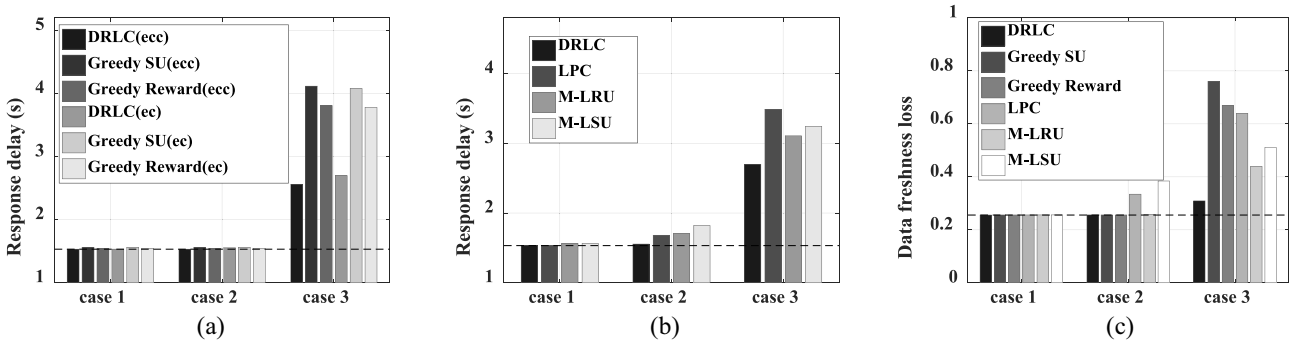

Fig. 9. Comparison for the search efficiency. (a) Response delay with different cooperative modes. (b) Response delay with different caching methods. (c) Data freshness with different caching methods.

B. Convergence Analysis

To analyze the convergence performance of the DRLC, the training process of case 2 is first studied. Fig. 7(a) and (b) shows the change of reward and the maximum Q value for several training epochs, respectively. In particular, the learning rate l is set as 0.001, ξ in the ξ-greedy training process is 0.1, and for the discount factor of reward, we have $\gamma = 0.9$. The optimal reward values are calculated via *Greedy Reward*, where the maximum Q value is obtained with the requests during each time slot. Although the reward values are mostly smaller than the optimal values, the dynamic characteristics are quite close when the epoch reaches 20. In Fig. 7(b), results show that the maximum Q values are similar to each other after the 15th epoch. The phenomenon demonstrates that DRLC converges within about 20 training epochs.

To further verify the effectiveness of parameter selection, we then compare the performance of Q values with different training settings. In specific, the average maximum Q value for all iterations in each epoch is presented. As shown in Fig. 7(c), the training converges within 20 epochs in all settings. Even though the intelligent algorithm will converge faster with a larger learning rate, the increasing of the maximum Q value shows that overfitting emerges with larger epoch number. Similarly, larger ξ values also cause the overfitting. For different discount factors, they mainly affect the converged maximum Q value, which is reasonable according to (14).

C. Real-Time Search Performance

Since the proposed SU model and DRLC method are based on the hierarchical indexing structure, the necessity of adopting this structure for searching should be demonstrated. We present the response delay for 50 search requests in Fig. 8. The data are cached according to solutions provided by *DRLC*, *Greedy SU*, and *Greedy Reward* in the ec mode. Results show that the caching solutions searched with our proposed framework achieve lower delay than that with *TR*. The main reason is that *HI* can locate the caching edge for requests faster via the index, and reduce the data search space. Specifically, the response delay of *DRLC* is the lowest for most requests, which proves the efficacy of our proposed caching method.

Based on the above analysis, *HI* will be used in the following simulations to search the cached SD. To verify the real-time search effectiveness of *DRLC*, the average response delay and data freshness are chosen as two major evaluation metrics. Since data requests can be generated by vehicles at different time slots from different locations, the freshness of the SD is calculated based on the uploading delay. According to (1), data freshness is measured by the freshness loss, the worst value of which is 1. The average statistics for the real-time search performance are presented in Fig. 9, where the caching decisions are determined step by step for requests in each time slot. Fig. 9(a) shows that *DRLC* costs lower delay than the *Greedy SU* and *Greedy Reward* in all simulation cases. But for all the caching methods, the response delay of the two cooperative modes is quite close to each other, which means that a supplement of the cloud caching size is unused in most cases. This is because that only a small group of SD can be transmitted successfully during the limited communication period between SV and the serving edge. Compared with LPC, M-LRU, and M-LSU, Fig. 9(b) shows that *DRLC* costs lower delay in all three cases, and the advantage is more obvious with more cooperative edges and larger data size. Specifically, the computational search space of case 2 and case 3 is larger with more caches. Meanwhile, the amount of uploading and caching data decreases with a bigger SD size, thus there are less feasible solutions in case 3. This phenomenon proves the effectiveness of *DRLC* in choosing the optimal solution. Similarly, *DRLC* performs better in the latter two cases in Fig. 9(c). Therefore, the result demonstrates that *DRLC* helps the edges to obtain SD faster, which is critical to provide an earlier response and support the real-time search.

D. Cache Hit Ratio

Fig. 10 shows the comparison of the cumulative distribution function (CDF) statistics of CHR. The best situation is that all data are cached in the target places, which leads to a vertical line at CHR = 100%. Hence, the closer the statistical curve to the point located at the bottom right of the coordinate axis, the better the performance. The results of the previously defined three cases are presented in three different subfigures, respectively. Specifically, all the caching methods present a vertical line in Fig. 10(a), which means that there is no data missing in case 1. Fig. 10(b) shows that all data are well cached except for M-LSU and LPC. Although sporadic data missing occurs in *DRLC(ec)*, the delay and freshness performance can still prove

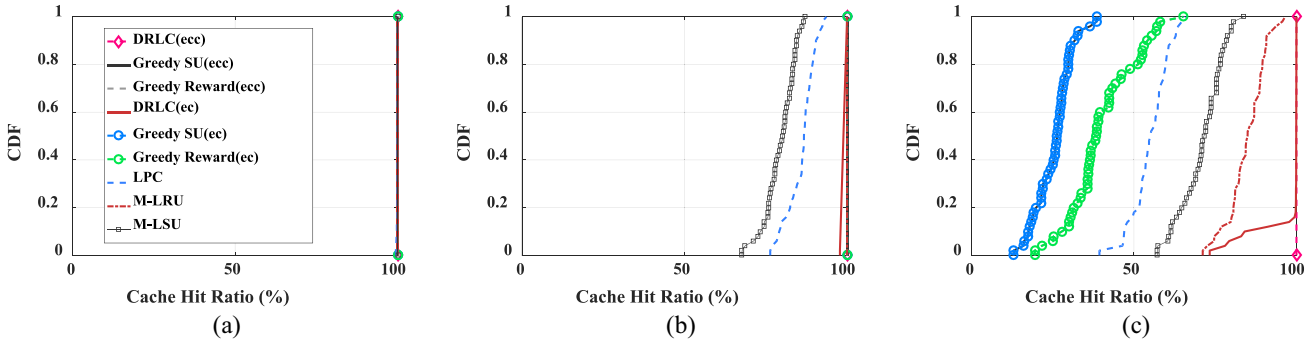

Fig. 10. Comparison for the CHR. (a) Case 1. (b) Case 2. (c) Case 3.

the real-time search effectiveness for our proposed caching method. With the increasing of SD size, only *DRLC* in the ecc mode can cache all the requested data, as shown in Fig. 10(c). On the other hand, even without the extra caching space provided by the cloud server, *DRLC(ec)* performs better than all the other methods. Intuitively, a larger data size means that less data can be uploaded and cached for searching. Hence, the performance in case 3 provides more powerful evidence for the effectiveness of the *DRLC* in supporting real-time search.

VI. Conclusion

To enable a real-time search for SD in vehicular networks, we have proposed a cooperative edge caching paradigm in this article. In particular, a hierarchical indexing framework and an SU model have been devised to measure the expected freshness of cached data. Then, on the basis of long-term search reward and historical requests, the DRLC method has been proposed to solve the RSC problem. The cooperative edge caching method provides a promising solution to improve data freshness and support real-time data search, and the proposed search structure and utility model can be valuable for future studies on data access and analysis for SD. For future work, we will focus on the caching of SD that is generated dynamically through various kinds of sensing devices, such as noise sensor, infrared detector, vibration sensor, and webcam. The application scenarios can be further expanded as the cache edges may need to work cooperatively to feed fused sensing information. In addition, when there are multiple matched results under a large-scale search area, we will delve into the joint optimization of result selection and transmission.

References

[1] M. Liu, D. Li, H. Wu, F. Lyu, and X. S. Shen, "Cooperative edge-cloud caching for real-time sensing big data search in vehicular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, 2021, pp. 1–6.

[2] H. B. Tulay and C. E. Koksal, "Increasing situational awareness in vehicular networks: Passive traffic sensing based on machine learning," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, Antwerp, Belgium, 2020, pp. 1–7.

[3] Q. Yuan, H. Zhou, Z. Liu, J. Li, F. Yang, and X. Shen, "CESense: Cost-effective urban environment sensing in vehicular sensor networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 9, pp. 3235–3246, Sep. 2019.

[4] X. Qin, Y. Xia, H. Li, Z. Feng, and P. Zhang, "Distributed data collection in age-aware vehicular participatory sensing networks," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14501–14513, Oct. 2021.

[5] F. Lyu *et al.*, "Towards rear-end collision avoidance: Adaptive beaconing for connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 1248–1263, Feb. 2021.

[6] "C-V2X use cases: Methodology, examples and service level requirements," Munich, Bavaria, 5G Autom. Assoc., White Paper, 2019.

[7] P. Dai *et al.*, "Temporal information services in large-scale vehicular networks through evolutionary multi-objective optimization," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 218–231, Jan. 2019.

[8] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2783–2797, Dec. 2020.

[9] O. Burkacky, J. Deichmann, and J. P. Stein, *Automotive Software and Electronics 2030: Mapping the Sector's Future Landscape*, McKinsey Company, Munich, Germany, 2019.

[10] P. Lee, D. Stewart, C. Calugar-Pop, and E. Talbot, "Technology, media and telecommunications predictions," Delloitte Touche Tohmatsu Ltd., London, U.K., Rep., 2017. [Online]. Available: https:// www2.deloitte.com/content/dam/Deloitte/global/Documents/Technology-Media-Telecommunications/gxdeloitte- 2017-tmt-predictions.pdf

[11] M. Liu, D. Li, Y. Zeng, W. Huang, K. Meng, and H. Chen, "Combinatorial-oriented feedback for sensor data search in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 284–297, Jan. 2020.

[12] T. X. Tran, D. V. Le, G. Yue, and D. Pompili, "Cooperative hierarchical caching and request scheduling in a cloud radio access network," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2729–2743, Dec. 2018.

[13] F. Lyu *et al.*, "LEAD: Large-Scale edge cache deployment based on spatio-temporal WiFi traffic statistics," *IEEE Trans. Mobile Comput.*, vol. 20, no. 8, pp. 2607–2623, Aug. 2021, doi: 10.1109/TMC.2020.2984261.

[14] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.

[15] S. Hisaka and S. Kamijo, "On-board wireless sensor for collision avoidance: Vehicle and pedestrian detection at intersection," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Washington, DC, USA, 2011, pp. 198–205.

[16] C. Xia, Y. Jiang, M. Peng, F.-C. Zheng, M. Bennis, and X. You, "Cooperative edge caching in fog radio access networks: A pigeon inspired optimization approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1–6.

[17] Y. Hui, Z. Su, and T. H. Luan, "Collaborative content delivery in software-defined heterogeneous vehicular networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 575–587, Apr. 2020.

[18] L. Yang, L. Zhang, Z. He, J. Cao, and W. Wu, "Efficient hybrid data dissemination for edge-assisted automated driving," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 148–159, Jan. 2020.

[19] J. Wang *et al.*, "Dynamic clustering and cooperative scheduling for vehicle-to-vehicle communication in bidirectional road scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1913–1924, Jun. 2018.

[20] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4309–4324, Oct. 2017.

[21] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 3rd Quart., 2019.

[22] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, Apr. 2013.

[23] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, "Caching transient data for Internet of Things: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2074–2083, Apr. 2019.
[24] K. Suto, H. Nishiyama, and N. Kato, "Postdisaster user location maneuvering method for improving the QoE guaranteed service time in energy harvesting small cell networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9410–9420, Oct. 2017.
[25] J. Liu, B. Bai, J. Zhang, and K. B. Letaief, "Content caching at the wireless network edge: A distributed algorithm via belief propagation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.
[26] M. Dehghan *et al.*, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1635–1648, Jun. 2017.
[27] G. Li *et al.*, "Understanding user generated content characteristics: A hot-event perspective," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kyoto, Japan, 2011, pp. 1–5.
[28] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, 2014, pp. 3185–3190.
[29] H. Zhou *et al.*, "ChainCluster: Engineering a cooperative content distribution framework for highway vehicular communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2644–2657, Dec. 2014.
[30] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Caching and operator cooperation policies for layered video content delivery," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, USA, 2016, pp. 1–9.
[31] P. Lin, Q. Song, J. Song, A. Jamalipour, and F. R. Yu, "Cooperative caching and transmission in comp-integrated cellular networks using reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5508–5520, May 2020.
[32] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. M. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 154–169, Jan. 2021.
[33] M. A. Lèbre, F. Le Mouël, and E. Ménard, "Resilient, decentralized V2V online stop-free strategy in a complex roundabout," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC-Spring)*, Nanjing, China, May 2016, pp. 1–5.

Mingliu Liu (Member, IEEE) received the B.E. and Ph.D. degrees from the School of Electronic Information, Wuhan University, Wuhan, China, in 2013 and 2019, respectively.

Since July 2019, she has been working as a Postdoctoral Fellow with the School of Electronic Information, Wuhan University. Her research interests include information search service, Internet of Things, cloud/edge caching, and big data-driven applications.

Dr. Liu is a member of the IEEE Computer and IEEE Communication Society.

Deshi Li received the Ph.D. degree in computer application technology from Wuhan University, Wuhan, China, in 2001.

He was a Visiting Scholar with the Network Lab, University of California at Davis, Davis, CA, USA. He is a Professor with the Electronic Information School, Wuhan University. He has published more than 100 research papers. His recent research projects include National Science and Technology Major Project of China (973 Program), National High Technology Program of China (863 Program), and National Natural Science Foundation of China. His research interests include wireless communication, Internet of Things, intelligence system, and SOC design.

Prof. Li currently serves as a member of the Internet of Things Expert Committee and the Education Committee of Chinese Institute of Electronics, and the Associate Chief Scientist in Space Communication area of Collaborative Innovation Center of Geospatial Technology, also is an Executive Trustee Member of China Cloud System Pioneer Strategic Alliance. He serves as a reviewer for many international academic journals and an expert evaluator for the Ministry of Science and Technology of China, Ministry of Education of China, and NSF China.

Huaqing Wu (Member, IEEE) received the B.E. and M.E. degrees from Beijing University of Posts and Telecommunications, Beijing, China, in 2014 and 2017, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2021.

She is currently a Postdoctoral Research Fellow with McMaster University, Hamilton, ON, Canada. Her current research interests include vehicular networks with emphasis on edge caching, wireless resource management, space–air–ground integrated networks, and application of artificial intelligence for wireless networks.

Dr. Wu received the Best Paper Award at IEEE GLOBECOM 2018 and *Chinese Journal on Internet of Things* 2020. She received the prestigious Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship Award in 2021.

Feng Lyu (Member, IEEE) received the B.S. degree in software engineering from Central South University, Changsha, China, in 2013, and the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018.

From September 2018 to December 2019 and from October 2016 to October 2017, he worked as a Postdoctoral Fellow and was a visiting Ph.D. student with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He is currently a Professor with the School of Computer Science and Engineering, Central South University. His research interests include vehicular networks, beyond 5G networks, big data measurement and application design, and edge computing.

Prof. Lyu is the recipient of the Best Paper Award of IEEE ICC 2019. He currently serves as an Associate Editor for IEEE SYSTEMS JOURNAL and *Peer-to-Peer Networking and Applications*, and served as a TPC member for many international conferences. He is a member of the IEEE Computer Society, Communication Society, and Vehicular Technology Society.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, 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 received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), and the Technical Recognition Award from Wireless Communications Technical Committee in 2019 and AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, and IEEE Globecom'07, and the Chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the President Elect of the IEEE ComSoc. He was the Vice President for Technical and Educational Activities, the Vice President for Publications, a Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a member of the IEEE Fellow Selection Committee of the ComSoc. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*. He is a registered Professional Engineer of Ontario, Canada, a Fellow of the Engineering Institute of Canada, Canadian Academy of Engineering, and Royal Society of Canada, a Foreign Member of the Chinese Academy of Engineering, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.