Cooperative Caching and Transmission in CoMP-Integrated Cellular Networks Using Reinforcement Learning

Peng Lin, *Member, IEEE*, Qingyang Song, *Senior Member, IEEE*, Jing Song, *Student Member, IEEE*, Abbas Jamalipour, *Fellow, IEEE*, and F. Richard Yu, *Fellow, IEEE*

***Abstract*—Mobile network caching (MNC) and Coordinated MultiPoint (CoMP) joint transmission (JT) techniques are expected to play a significant role in future networks. Yet, when considering caching in a CoMP-integrated network where both single transmission (ST) and JT are permitted, some challenges need to be tackled. In this paper, we propose a cooperative online caching strategy to realize autonomous transmission-aware content caching and updating for video content delivery. Different from existing works, this paper considers practical time-varying user request patterns, and exploits storage-level and transmission-level cooperation to jointly optimize content caching under dynamic user demands for JT and ST. The former helps reduce caching redundancy and the latter helps BSs create CoMP-JT opportunities to improve cell-edge throughput. We formulate the content caching problem to be a Markov decision process (MDP), in which the system reward is defined as the delay reduction after performing a caching action. The goal is to maximize the system reward. Then, we develop a reinforcement learning (RL)-based online learning algorithm to search the optimal caching policy. To overcome the "curse of dimensionality," the designed RL-based algorithm is extended with linear approximation. The system performance in terms of delay and cache hit rate is evaluated in the simulations, and the results demonstrate the effectiveness of the proposed strategy.**

***Index Terms*—CoMP, delay, caching, reinforcement learning.**

I. Introduction

EXPLOSION of wireless data traffic has posed a great burden to current mobile networks. According to [1], 55 percent of these data traffic will be hot video/audio. To support the continuous increase of video/audio streaming without the problems of delay and lack of capacity, *Coordinate Multipoint* (CoMP) transmission technology is proposed and studied to improve network coverage and data rate [2]. Nevertheless, in mobile networks, the backhaul availability and capacity may become the performance bottleneck. Fortunately, *mobile network caching* (MNC) has been proposed as a promising solution for the backhaul bottleneck, due to its ability to reduce duplicate content transmissions [3]–[5]. Hence, these trends prompt network operators to consider the performance improvement profited from the integration of MNC and CoMP.

MNC, extended from the in-network caching, has the potential to reduce backhaul access latency and traffic load by caching contents at base stations (BSs) [6]. The benefits of caching mainly come from reducing the number of hops from the content source to mobile terminals (MTs). To properly combine caching and wireless networks, significant studies have been done, e.g., the least recently used (LRU), the most popular content (MPC) [7], the cooperative caching and scheduling [8]–[12], integrated heterogeneous caching [13], etc. These works study a coupled problem of what and where to cache in cellular networks.

CoMP is one of the promising inter-cell cooperation technologies, and has been standardized in 3GPP for the LTE-advanced system to reduce inter-cell interference and increase cell-edge throughput [2]. Different from traditional single transmission (ST) scenarios where each MT can only be served by a single BS at any given time, the BSs in CoMP-integrated networks can collaboratively transmit a content to the MT by forming a multi-input single-output (MISO) channel between BSs and the MT. This technique is referred to as CoMP Joint Transmission (CoMP-JT, also referred to as JT). In a CoMP-integrated cellular network (CICN), both joint transmission (JT) and ST are possible choices for MTs. MTs at cell edges can utilize JT to improve signal quality, receiving downlink data from multiple neighboring BSs. The success of CoMP-JT will depend on whether the related BSs can provide the contents requested by the potential JT-employed MTs.

As network operators are committed to providing users with low-latency services, the CoMP and MNC technologies appear as promising candidates to enhance network performance in terms of delay. Considering the potentials of MNC and CoMP, it

Manuscript received December 2, 2019; revised February 10, 2020; accepted March 4, 2020. Date of publication March 18, 2020; date of current version May 14, 2020. This work was supported in part by the National Nature Science Foundation of China under Grants 61775033 and 61771120 and in part by the Fundamental Research Funds for the Central Universities under Grant N171602002. The review of this article was coordinated by Dr. B. Mao. *(Corresponding author: Qingyang Song.)*

Peng Lin is with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China, and also with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: penglin11@foxmail.com).

Qingyang Song and Jing Song are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: songqy@cqupt.edu.cn; songjingchn@foxmail.com).

Abbas Jamalipour is with the School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2006, Australia (e-mail: a.jamalipour@ieee.org).

F. Richard Yu is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: richard.yu@carleton.ca).

Digital Object Identifier 10.1109/TVT.2020.2981657

TABLE I
NOTATIONS

Symbol	definition
$\mathcal{M}$;	Set of BSs in the network
$\mathcal{U}_i$	Set of MTs associated with BS b_i
$\mathcal{N}_i$;	Set of neighboring BSs of BS b_i
$\mathcal{F}$;	Set of contents in the network
C_i;	Cache capacity of BS b_i
s_k;	Size of content f_k
$\mathcal{P}$;	State set of content popularity profile
$p_{i,k}(t)$;	Popularity of content f_k in BS b_i at time slot t
p_d;	Probability of each MT departing its current location
p_t;	Probability of each cell-edge MT leaving a cell
p_r;	Probability of each MT requesting a content
$\mathcal{D}$;	State set of request distribution coefficients
$\eta_i(t)$;	Request distribution coefficient at time slot t
$N_i^{cr}(t)$	Number of requests from cell-core MTs at time slot t
$N_i^{er}(t)$	Number of requests from cell-edge MTs at time slot t
$\Psi(\vartheta(t), \chi(t))$	Reward function for taking an action at time slot t
π	Policy function that maps states to actions
$\mathcal{S}$;	Set of possible system states
$\mathcal{A}$;	Set of possible caching actions
$\vartheta_i(t)$;	State of BS i at time slot t
$\chi_i(t)$;	Caching action of BS b_i at time slot t
$\overline{\mathcal{R}}(\vartheta, \pi(\vartheta))$;	Mean system reward
$V^{\pi}(\vartheta_i(t))$;	Cumulative state-value function at time slot t

is beneficial to combine them together. However, the integration of MNC and CoMP in CICNs is not well addressed as several challenges are induced by this combination. First, in the CICN where both JT and ST are permitted, the content placement should be transmission-aware, which means that content provisioning at BSs should be based on MTs' demands for JT and ST. The reason is that the ST-prioritized caching provides the MTs with a lower transmission rate than JT, while the JT-prioritized caching sacrifices storage space and causes high cache miss for ST-employed MTs. Second, content caching and updating should be proactive according to the dynamics of user request pattern, i.e., the time-varying content popularity and users' location distribution; however existing caching strategies simply take historical request records as the predicted user demands for the next time instant. Third, as BSs are densely deployed with topological associations, content sharing between BSs should be encouraged to improve the storage utilization.

Thus, to address these issues, in this paper, we consider practical time-varying user request patterns, and study the joint optimization problem of content caching for JT and ST in CICNs. We aim to design an autonomous caching strategy for video content streaming to realize fully automatic transmission-aware content placing and updating. To achieve this goal, we propose a reinforcement-learning (RL) based cooperative caching strategy, which begins from knowing nothing about the system and then learns to make caching decisions based on users' demands for JT and ST without using prior knowledge of content popularity and users' location distribution. Specifically, the main technical contributions of this paper are as follows:

1) We design a cooperative caching strategy considering time-varying user request patterns in CICNs. The strategy holds storage-level and transmission-level cooperation. The former helps to reduce caching redundancy and the latter helps BSs create JT opportunities to improve cell-edge throughput.
2) Under the proposed caching strategy, we define a reward function which calculates the saved delay in transmitting contents with ST and JT, respectively. Then, based on the dynamic user demands for ST and JT, the content caching problem is formulated as a Markov Decision Process (MDP) to maximize the long-term system reward.
3) We develop an RL-based algorithm (e.g., Q-learning) to find the optimal caching policy. In this way, the content placement is updated in an online fashion, and the optimal caching policy is obtained gradually. To overcome the "curse of dimensionality," the designed Q-learning algorithm is extended with linear approximation, so that it can be applied in practical environments with a large number of contents.
4) In the simulations, the convergence of the Q-learning based caching strategies for a small number of contents and a large number of contents are evaluated, respectively. Meanwhile, the delay performance of the proposed strategy is evaluated and compared with existing caching strategies. The results show the advantages of our method.

The remainder of this paper is organized as follows. The related work is presented in Section II. The network model and the modeling of user request patterns are presented in Section III. In Section IV, we formulate the content caching problem to be an MDP. In Section V, we propose the RL-based caching strategy. Section VI presents the simulation settings. The simulation results and analysis are given in Section VII. Section VIII concludes the paper.

II. RELATED WORK

In recent years, some studies have focused on content caching at BSs in mobile networks. Traditional isolated caching strategies, such as the LRU and MPC [7] strategies, have good performance in sparse networks. However, in the network where BSs are densely deployed and hold close topological associations, the isolated caching strategies will lead to content redundancy. To overcome the problem, cooperative caching was proposed from different perspectives. For example, the studies in [8]–[10] proposed collaborative content caching and scheduling at small BSs to offload the traffics from core networks. The studies in [11] and [12] proposed cooperative multitier caching architectures, in which popular contents were cooperatively placed in heterogeneous networks, and the caching performance in terms of delay, hit rate, and outage rate was improved. The integrated caching in heterogeneous networks was studied in [13], where three types of cooperation among caching nodes were exploited to reduce content delivery delay. Joint optimization of caching and computing in 5 G IoVs was studied in [14], where the best trade-off between the QoE of users and the profits of MNOs can be acquired by designing a novel task assignment and resource allocation scheme. All these works focus on caching strategy design in traditional ST networks.

Content caching in CoMP-integrated networks, where both ST and JT are permitted, has also been studied in recent years.

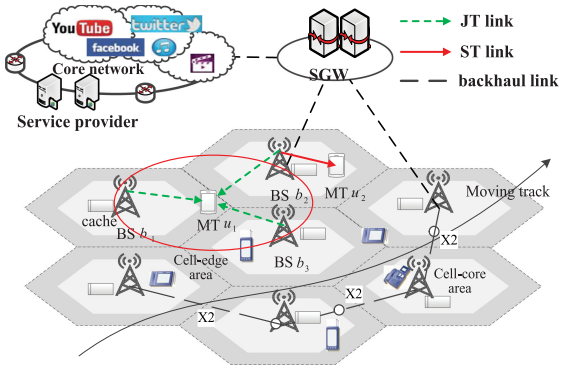


Fig. 1. Caching and transmission in a CoMP-integrated cellular network.

The studies in [15] originally proposed a cache-enabled opportunistic CoMP framework for wireless video streaming. In this framework, the relays and BSs cached a portion of video files, and then opportunistically employed CoMP to achieve MIMO cooperation gain and reduce the backhaul load. The popularity-based combinational caching in CICNs was introduced by [16], where the storage space of each BS was partitioned to store the most popular contents and less popular contents, respectively. The JT clustering problem with the consideration of caches at base stations was studied by [17], [18]. Some studies took into account the opportunities of using JT and ST when exploring caching [19], [20]. However, these studies just considered static user demands for ST and JT instead of real-time mobile network status.

To help mobile network operator manage their networks more intelligently, the application of machine learning in communication systems was studied by [21]–[25]. An RL-based algorithm can provide a viable solution to dynamic resource optimization problems, since RL can make decisions from past experiences by a trial-and-error process [26]. Deep-learning-based intelligent architectures and algorithms for traffic control were introduced in [21]. The intelligent frameworks for 5G-enabled communications were studied by [27], [28]. In these works, the resource allocation efficiency between vehicles and macrocells was improved by developing distributed RL-based optimization methods, which has great significance for future network management. The resource optimization problem in dynamic channel environments was studied in [29], where the deep RL algorithm was used to optimize bandwidth allocation and buffer management. However, the existing dynamic optimization methods have not well addressed the problem of content caching with real-time user demands.

III. System Description and Modeling

A. Network Model

We consider a CoMP-integrated cellular network (CICN) as shown in Fig. 1. In the network, there are M BSs, denoted by $\mathcal{M} = \{b_1, \ldots, b_i, \ldots, b_M\}$, under the centralized control of a service gateway (SGW). Each BS supports ST and JT to serve MTs. The service providers (SPs) (e.g., YouTube, Facebook, and so on) provide a set of contents throughout the whole network. The content set is denoted by $\mathcal{F} = \{f_1, \ldots f_k, \ldots, f_F\}$, in which the size of content f_k is s_k. The cache size of BS b_i is C_i. The BSs are interconnected by the X2 links. We denote $\mathcal{N}_i$ as a set of neighboring BSs of b_i, which is given by

$$\mathcal{N}_i = \{b_r \in \mathcal{M} : \delta_{ir} = 1, b_i \neq b_r\} \tag{1}$$

where $\delta_{ir} \in \{0, 1\}$ indicates whether BSs b_i and b_r are connected.

In addition, a dynamic set of MTs associated with BS b_i is denoted by $\mathcal{U}_i = \{u_1, \ldots, u_j, \ldots, u_{N_i}\}$, where N_i is the number of MTs. The *long-term averaged* signal to interference plus noise ratio (SINR) $\overline{g}_{b,u}$ [30] is used to evaluate the signal quality of MTs. Correspondingly, an MT can be marked as a cell-core MT when $\overline{g}_{b,u} \geq \tau$, where τ is an SINR threshold. Otherwise, it will be marked as a cell-edge MT. To improve cell-edge throughput, the neighboring BSs are grouped into CoMP-JT clusters to collaboratively serve the cell-edge MTs with JT. Because CoMP-JT contributes limited improvement in terms of throughput for cell-core MTs [31], we assume that only the cell-edge MTs can obtain JT service.

Moreover, we consider a BS association strategy in which each MT in CoMP cluster is only associated with the BS which has the biggest SINR, and the BS is called as *anchored-BS*. For a cell-edge MT, the other BSs in the CoMP cluster are called *assisted-BSs*. The assisted-BSs coordinate with the anchored-BS to perform JT, e.g., in Fig. 1, for MT u_1, BS b_1 is an anchored-BS and BSs b_2 and b_3 are assisted-BSs. Traditionally, in the ST-only network, when MT u_1 in the cell of BS b_1 sends a content request, BS b_1 checks the local content set and transmits the requested content to MT u_1 if the content has been cached. However, in the CoMP-integrated network, to perform JT, BS b_1 firstly forwards the request to the assisted-BSs b_2 and b_3, then BSs b_1, b_2, and b_3 collaboratively transmit the content to MT u_1 if the content has been cached at all the three BSs. Otherwise, MT u_1 will be served by BS b_1 with ST.

B. Time-Varying Content Popularity Profile

In the considered system, we divide time into discrete time slots. Content caching and transmission in this system are carried out in a slotted fashion over time slot $t = 1, 2, \ldots, \infty$. We assume that the time slot gap is large enough (e.g., several seconds) so that the retrieval and delivery of a content can be accomplished within one time slot. At the end of each time slot, BS b_i updates its cached contents for possible transmissions in subsequent time slots. Content popularity profile indicates users' temporal demands for different contents during a certain period of time, which plays an important role for BSs to make caching decisions. In practice, the content popularity is time-varying during one day and its values at different time instances are correlated. At the beginning of a day (for example at 1 AM∼5 AM), when the network load is at an off-peak period, there are less content requests in the network, which leads to a more flattened head of the popularity distribution in content set $\mathcal{F}$. As users generate content requests gradually and continuously during the daytime, some popular contents appear and account for a majority of requests. The content popularity follows the Zipf distribution

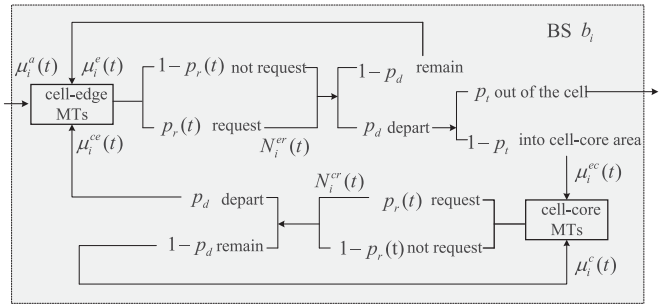


Fig. 2. Process of MTs requesting and moving in cell of BS b_i.

with a Zipf function $\zeta : o_l = 1/(l^\alpha \sum_{n=1}^{F} n^{-\alpha})$, where l is the rank of occurrence frequency and α is the skewness factor [32].

In this paper, to formulate the dynamics of the content popularity, we first construct the Zipf function with different skewness factors α. Then, we make the content popularity profile follow a finite-state Markov process with multiple popularity states. Denote the local content popularity profile at time slot t in the cell of BS b_i by an $F \times 1$ vector $\mathrm{P}_i(t)$, where the k-th entity $[\mathrm{P}_i(t)]_k = p_{i,k}(t)$ indicates the expected demand for content f_k at time slot t. As a consequence, the popularity profile follows an underlining Markov process with a state set $\mathcal{P} = \{\mathrm{P}^1(\alpha_1), \ldots, \mathrm{P}^j(\alpha_j), \ldots \mathrm{P}^G(\alpha_G)\}$, where $\mathrm{P}^j(\alpha_j)$ is the j-th state of content popularity profile constructed by the Zipf function ζ with skewness factors α_j, and G is the number of states. To obtain the user request preference, i.e., $\mathrm{P}_i(t) \in \mathcal{P}$, BS b_i can determine the popularity of each content by observing the real-time content requests as follows:

$$p_{i,k}(t) = \frac{\text{Number of requests for } f_k \text{ at } b_i \text{ at time slot } t}{\text{Number of total requests at } b_i \text{ at time slot } t} \quad (2)$$

When the caching system starts to work, a learning agent in the BS calculates the number of content requests and determines the popularity state in $\mathcal{P}$ according to (2). The observed dynamic content popularity is considered in the following design of an RL-based caching optimization scheme, which is formulated in Section IV. Note that, although the popularity states can be observed by calculating the number of user requests periodically, the state transition probability of the Markov process is not available in practice.

C. Time-Varying Request Distribution

Request distribution expresses the spatial composition of content requests from MTs, and it is influenced by the distribution of MTs' locations. In this paper, we use a request distribution coefficient to denote the ratio of the number of cell-edge requests to the number of cell-core requests. To describe the dynamics of request distribution, we analyze the movements of MTs over time slots in the cell of BS b_i. As shown in Fig. 2, at the beginning of time slot t, multiple MTs randomly enter the cell-edge area of BS b_i and become the cell-edge MTs. At the end of the time slot, a moving MT may depart its current location with probability p_d. Since the system is sampled every several seconds, it is reasonable to assume that each cell-edge MT is at one of two moving states within one time slot: (i) A cell-edge MT leaves its current cell and enters the cell-edge area of a neighboring BS with probability p_t. (ii) A cell-edge MT leaves its current location and enters the cell-core area with probability $1 - p_t$. For a cell-core MT, it can only become a cell-edge MT by entering the cell-edge area of the current cell.

The mobility model of the MTs in the cell of b_i is described in Fig. 2. In this system, each MT generates a content request independently with probability p_r, and an MT can generate multiple independent requests for different contents. During time slot t, new MTs that enter into the cell-edge area of BS b_i from other cells follow the Poisson process with average arrival rate λ, and the number of them at time slot t is denoted by $\mu_i^a(t)$. The number of remaining cell-edge MTs in the cell of BS b_i at time slot t is denoted by $\mu_i^e(t)$, which can be obtained by

$$\mu_i^e(t) = \left[\mu_i^a(t) + \mu_i^{ce}(t-1) + \mu_i^e(t-1)\right](1 - p_d), \quad (3)$$

where $\mu_i^{ce}(t-1)$ is the number of MTs that leave from the cell-core area and go into the cell-edge area of BS b_i during time slot $t-1$, and its presentation at time slot t is written as

$$\mu_i^{ce}(t) = \left[\mu_i^{ec}(t-1) + \mu_i^c(t-1)\right] p_d, \quad (4)$$

where $\mu_i^{ec}(t-1)$ is the number of cell-edge MTs changing into the cell-core MTs at time slot $t-1$, and determined by probabilities p_d and p_t. $\mu_i^c(t-1)$ is the number of remaining cell-core MTs at time slot $t-1$. Both $\mu_i^{ec}(t-1)$ and $\mu_i^c(t-1)$ can be obtained in a recursive manner similar to (3) and (4).

Then, we can obtain the number of requests generated by the cell-edge MTs at time slot t as

$$N_i^{er}(t) = \left(\mu_i^a(t) + \mu_i^e(t) + \mu_i^{ce}(t)\right) p_r. \quad (5)$$

Similarly, the number of requests generated by the cell-core MTs during time slot t can be obtained as

$$N_i^{cr}(t) = \left(\mu_i^{ec}(t) + \mu_i^c(t)\right) p_r. \quad (6)$$

Recalling the definition of request distribution coefficient, we denote $\eta_i(t)$ as the request distribution coefficient at BS b_i, which can be obtained as

$$\eta_i(t) = \frac{N_i^{er}(t)}{N_i^{er}(t) + N_i^{cr}(t)}. \quad (7)$$

Therefore, the state transition of $\eta_i(t)$ follows a first-order Markov chain as below

$$P\left(\eta_i(t) | \eta_i(1), \ldots, \eta_i(t-1)\right) = P\left(\eta_i(t) | \eta_i(t-1)\right) \quad (8)$$

We divide the value space of $\eta(t)$ into discrete levels, i.e., η_1, if $0 \le \eta_i(t) < \eta_1^*$; η_2, if $\eta_1^* \le \eta_i(t) < \eta_2^*$; $\ldots$; η_E, if $\eta_{E-1}^* \le \eta_i(t) < \eta_E^*$, where $\eta_1^*, \ldots, \eta_E^*$ are the state value bounds. Then, the state space of $\eta(t)$ can be defined as $\mathcal{D} = \{\eta_1, \eta_2, \ldots, \eta_E\}$. Based on the underlying state transition probability, coefficient $\eta(t)$ varies from one state to another as time slot elapses.

D. Cooperative Caching Strategy

In this part, we introduce a cooperative caching strategy. Given the user request pattern determined by $\mathrm{P}_i(t)$ and $\eta_i(t)$, BS b_i can cache contents to serve MTs with low-delay transmissions. In the CoMP-integrated system, a cell-edge MT can obtain

the desired content either from its anchored-BS by ST or from both anchored-BS and assisted-BSs in the CoMP-JT cluster by JT, based on the caching state of the desired content. To make the caching deployment adaptive to the hybrid transmission, we consider an integrated caching strategy with the following two types of cooperation:

Storage-level cooperation: The BSs in the network are connected with each other through high-capacity X2 links. To achieve efficient cooperation between BSs, the caching information and the cached contents are shared between adjacent BSs via the X2 links. Accordingly, for a certain MT served with ST, (i) the desired content can be fetched from the anchored-BS directly if the content has been cached; (ii) the desired content can be fetched from a neighboring BS (that is inaccessible for the MT but has the content cached) indirectly through the X2 link; (iii) the desired content will be fetched from the SP if all the neighboring BSs do not cache the content. In order to achieve high cache hit rate for both cell-core MTs and cell-edge MTs, we use this type of cooperation to increase content diversity in MTs' accessible storage space so that more different popular contents are cached.

Transmission-level cooperation: When a cell-edge MT requests a content, the BSs in CoMP-JT cluster jointly determine the transmission method (i.e., ST or JT) depending on the caching state of the content. If the desired content is cached in both the anchored-BS and assisted-BSs in the CoMP-JT cluster, the MT will be served with JT. Otherwise, i.e., the content is only cached by the anchored-BS or an assisted-BS, the content will be transmitted by ST. To improve cell-edge throughput, BSs are encouraged to cache more same popular contents to facilitate more JT opportunities. In this way, content delivery delay can be reduced with JT especially when most requests are from the cell-edge area. Note that, another opportunity for obtaining JT service is to wait for all the BSs (in the CoMP cluster) that do not have the desired content to download it from a cooperative BS or the SP. However, compared with the case in which the content has been cached and ready for JT, it triggers extra delay and backhaul load. So we do not consider it as an option.

IV. Problem Formulation

A. System State and Action

Our goal is to design an online caching strategy which can realize joint content placement and update for JT and ST under the time-varying user request pattern. To achieve this goal, we model the caching problem to be an MDP. The MDP framework can be defined by a tuple $\{\mathcal{S}, \mathcal{A}, \Psi(\vartheta, \chi)\}$. Specifically,

- $\mathcal{S}$ is the set of possible states for BSs, and $\mathcal{S} := \{\vartheta_i | \vartheta_i = [\mathrm{P}_i, \eta_i], \forall b_i \in \mathcal{M}\}$, where $\mathrm{P}_i \in \mathcal{P}$ and $\eta_i \in \mathcal{D}$.
- $\mathcal{A} := \{\chi_i | \chi_i \in \{0,1\}^F, \|\chi_i\|_1 = N_{C_i}, \forall b_i \in \mathcal{M}\}$ is the set of feasible caching actions, where N_{C_i} is the number of contents can be cached with storage size C_i. χ_i is a $F \times 1$ binary vector indicating the caching actions of BS b_i. The k-th entry $\chi_{i,k} = 1$ indicates that BS b_i decides to cache content f_k, and $\chi_{i,k} = 0$ otherwise.
- $\Psi(\vartheta, \chi)$ is the reward function that captures the expected saved delivery delay if taking action χ under state ϑ.

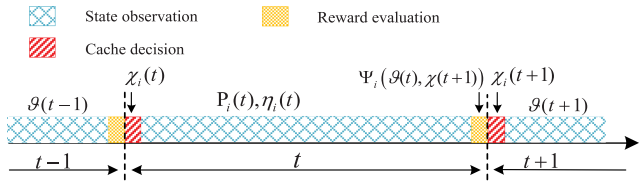


Fig. 3. Schematic of BS b_i working across time slot t.

The process for BS b_i to work over time slot t is described in Fig. 3. In this system, a time slot is divided into three phases, i.e., *cache decision*, *state observation*, and *reward evaluation*. Each BS proactively takes caching actions based on the experienced system state so as to reduce the content delivery delay in the next time slot. Denote $\vartheta_i(t) = [\mathrm{P}_i(t), \eta_i(t)]$ as the system state of BS b_i at time slot t. Upon taking action $\chi_i(t)$ at the cache decision phases, the system reward $\Psi_t(\vartheta_i(t-1), \chi_i(t) | \vartheta_i(t))$ can be obtained at the reward evaluation phase of time slot t. The MDP for BS b_i over time slot t is described as follows:

1) Based on state $\vartheta_i(t-1)$, BS b_i takes action $\chi_i(t) \in \mathcal{A}$ at the caching decision phase of time slot t.
2) During the state observation phase, BS b_i senses current user request information, and reaches state $\vartheta_i(t)$.
3) Based on state $\vartheta_i(t)$, BS b_i computes reward $\Psi_t(\vartheta_i(t-1), \chi_i(t) | \vartheta_i(t))$ for taking action $\chi_i(t)$ at the reward evaluation phase.

Denote $\pi : \mathcal{S} \to \mathcal{A}$ as the policy function, which maps any state $\vartheta \in \mathcal{S}$ to action $\chi \in \mathcal{A}$. Under policy π, the caching decision at time slot $t+1$ for BS b_i will be carried out via $\chi_i(t+1) = \pi(\vartheta_i(t))$. In the MDP framework, each BS uses system reward $\Psi(\vartheta, \chi)$ to quantify the delay reduction of taking a caching action compared with no action. The details of calculating $\Psi(\vartheta, \chi)$ are provided in the following subsection.

B. Reward Calculation

As this study aims to reduce content delivery delay, we choose the expected delay reduction under a preset caching decision as the system reward. The content delivery delays experienced by the MTs depend on the content caching states at the BSs and the transmission methods used by the BSs. The involved delays are intracell delay, intercell delay, and outer-cell delay. For BS b_i, the intracell delay can be distinguished into transmission delays using ST and JT, which are influenced by the channel state information. It should be noted that the high dynamics of the wireless channels are beyond the consideration in this paper, and the transmission delays using ST and JT can be calculated based on the average transmission rates r_i^{ST} and r_i^{JT}, which can be obtained using the Shannon bound with the long-term averaged SINRs between BSs and MTs. The intercell delay appears when BS b_i fetches content f_k from one of the neighboring BSs, and can be calculated based on the average transmission rate r_i^{NE} between the two BSs. Similarly, the outer-cell delay can be calculated based on the average backhaul transmission rate, r_i^{SP}, from the SP to BS b_i.

We take BS b_i in a CoMP cluster as an example to calculate the system reward. For the MTs in the cell of BS b_i, the system

reward for taking action $\chi_i(t)$ under state $\vartheta_i(t-1)$ can be expressed as the conditional reward $\Psi_t(\vartheta_i(t-1), \chi_i(t)|\vartheta_i(t))$ as follows:

$$\Psi_t(\vartheta_i(t-1), \chi_i(t)|\vartheta_i(t)) = R_{i,c}(t) + R_{i,e}(t), \tag{9}$$

where the direct rewards $R_{i,c}(t)$ and $R_{i,e}(t)$ are defined as the saved delivery delay for cell-core and cell-edge MTs, respectively. For the cell-core MTs, the action to cache content f_k will reduce content delivery either from the CP or from a neighboring BS to BS b_i, depending on the caching state of the content. Denote $x_n^k(t) \in \{0,1\}$ as the caching state of content f_k at neighboring BS b_n, which can be obtained with real-time message exchange between BSs. If content f_k has been cached at any neighboring BSs of BS b_i, i.e., $\bigcup_{b_n \in \mathcal{N}_i} x_n^k(t) = 1$, the reduced delay is s_k / r_i^{NE}; otherwise, i.e., $\bigcup_{b_n \in \mathcal{N}_i} x_n^k(t) = 0$, the reduced delay is s_k / r_i^{CP}. Accordingly, the direct reward $R_{i,c}(t)$ can be presented as

$$R_{i,c}(t) = \sum_{k=1}^{F} \chi_{i,k}(t) p_{i,k}(t)(1-\eta_i(t))\zeta_{i,k}(t), \tag{10}$$

where

$$\zeta_{i,k}(t) = \begin{cases} \dfrac{s_k}{r_i^{\mathrm{NE}}}, & \bigcup\limits_{b_n \in \mathcal{N}_i} x_n^k(t) = 1 \\ \dfrac{s_k}{r_i^{\mathrm{SP}}}, & \bigcup\limits_{b_n \in \mathcal{N}_i} x_n^k(t) = 0 \end{cases} \tag{11}$$

is the reduced delay of cell-core MTs retrieving content f_k through BS b_i. For the cell-edge MTs, in addition to the delay saving in the backhaul network, the action to cache content f_k also brings delay performance gaining by performing JT on the premises that each BS in the CoMP cluster has cached content f_k, i.e., $\prod_{b_n \in \mathcal{C}_i} x_n^k(t) = 1$. Therefore, the reward $R_{i,e}(t)$ can be calculated as

$$R_{i,e}(t) = \sum_{k=1}^{F} \chi_{i,k}(t) p_{i,k}(t) \eta_i(t) \xi_{i,k}(t), \tag{12}$$

where

$$\xi_i(t) = \begin{cases} \dfrac{s_k}{r_i^{\mathrm{SP}}}, & \bigcup\limits_{b_n \in \mathcal{N}_i} x_n^k(t) = 0 \\ \dfrac{s_k}{r_i^{\mathrm{NE}}}, & 0 < \sum\limits_{b_n \in \mathcal{C}_i} x_n^k(t) < |\mathcal{C}_i| \\ \dfrac{s_k}{r_i^{\mathrm{ST}}} - \dfrac{s_k}{r_i^{\mathrm{JT}}}, & \prod\limits_{b_n \in \mathcal{C}_i} x_n^k(t) = 1 \end{cases} \tag{13}$$

is the reduced delay of cell-edge MTs retrieving content f_k through BS b_i. $\mathcal{C}_i$ is the CoMP cluster which contains all the necessary assisted-BSs to perform JT with BS b_i. Based on the above analysis, given the surrounding content placement of BS b_i, the system reward $\Psi(\vartheta_i(t-1), \chi_i(t)|\vartheta_i(t))$ induced by taking action $\chi_i(t)$ can be obtained after observing the system state $\vartheta_i(t)$, i.e., $\mathrm{P}_i(t), \eta_i(t)$.

In order to find the optimal policy π^*, we define $\varphi_i(t)$ to evaluate the long-term system reward, which is given as

$$\varphi_i(t) = \sum_{\tau=t}^{\infty} \beta^{\tau-t} \Psi(\vartheta_i(\tau), \pi(\vartheta_i(\tau))), \tag{14}$$

where $0 \le \beta \le 1$ is a discount factor that determines the impact of the current caching action on future rewards. Accordingly, given a caching policy π, the caching performance can be measured by a cumulative state-value function $V^\pi(\vartheta_i(t))$, which calculates the expected long-term system reward as follows

$$\begin{aligned} V^\pi(\vartheta_i(t)) &= \mathbb{E}\Big[\varphi_i(t)\Big] \\ &= \mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t} \Psi(\vartheta_i(\tau), \pi(\vartheta_i(\tau))\right]. \end{aligned} \tag{15}$$

Our goal is to find the optimal policy π^* so as to maximize the long-term system reward of any state, i.e.,

$$\pi^* = \arg\max_{\pi \in \Pi} V^\pi(\vartheta_i(t)), \tag{16}$$

where Π denotes the set of possible policies. Then, the caching policy optimization problem can be formulated to maximize the expected long-term system reward.

Problem (I)

$$\max \quad V^\pi(\vartheta_i(t)), \qquad \forall t \tag{17a}$$

$$\text{s.t.} \quad \sum_{k=1}^{F} s_k \chi_{i,k}(t) \le N_{C_i}, \quad \forall b_i, \forall t \tag{17b}$$

$$\pi \in \Pi \tag{17c}$$

where (17b) ensures that the caching action is limited by the storage capacity of BS b_i. Problem (I) is a sequential decision-making problem with time-varying parameters. In the ensuing section, we will present the problem in Bellman equations, and resort to RL to obtain the optimal policy.

V. Q-Learning Based Cooperative Caching

A. Problem Transformation

Bellman equation is a dynamic programming equation. It provides necessary conditions for obtaining the optimal policy in sequential decision-making problems. Denote the transition probability of the system going form current state ϑ to the next state ϑ' by $P_{\vartheta\vartheta'}$, which is given as

$$P_{\vartheta\vartheta'} = \Pr\{\vartheta(t) = \vartheta' \mid \vartheta(t-1) = \vartheta, \pi(\vartheta) = \chi\}. \tag{18}$$

For simplicity, we drop the BS index i and the symbol of time slot t. Then, the objective function of Problem (I) can be written into the Bellman equation in an iterative fashion as

$$V^\pi(\vartheta) = \overline{\mathcal{R}}(\vartheta, \pi(\vartheta)) + \beta \sum_{\vartheta' := [\mathrm{P}', \eta']} P_{\vartheta\vartheta'} V^\pi(\vartheta'), \tag{19}$$

where $\overline{\mathcal{R}}(\vartheta, \pi(\vartheta))$ is the mean of the current system reward $\Psi(\vartheta, \pi(\vartheta))$ in (9). The second item in (19) is the discounted

version of future state-value function under a given policy π. The mean reward $\overline{\mathcal{R}}(\vartheta, \pi(\vartheta))$ can be written as

$$\begin{aligned}\overline{\mathcal{R}}(\vartheta, \pi(\vartheta)) &= \mathbb{E}^{\pi}_{\mathrm{P}',\eta'}\left[\Psi(\vartheta, \pi(\vartheta)) \mid \mathrm{P}', \eta'\right] \\ &= \sum_{\vartheta':=[\mathrm{P}',\eta']} P_{\vartheta\vartheta'}\left[\Psi(\vartheta, \pi(\vartheta)) \mid \mathrm{P}', \eta'\right]. \end{aligned} \tag{20}$$

With $P_{\vartheta\vartheta'}$, we can obtain $V^{\pi}(\vartheta)$ by solving (19), and eventually obtain the optimal policy π^* through sequential policy iteration. Specifically, the evaluation of the caching action under policy π is controlled by the Q-function [26],

$$Q_{\pi}(\vartheta, \chi') = \overline{\mathcal{R}}(\vartheta, \chi') + \beta \sum_{\vartheta' \in \mathcal{S}} P_{\vartheta\vartheta'} V^{\pi}(\vartheta'). \tag{21}$$

Then, policy π can be updated by calculating the optimal Q-value for each experienced state in the iterative process. The policy iteration process initializes with π_0, and then proceeds with policy update at the n-th iteration as follows:

1) Compute $V^{\pi_n}(\vartheta)$ for each state $\vartheta \in \mathcal{S}_n$ under policy π_n, according to the Bellman equation in (19), where

$$\mathcal{S}_n := \{\vartheta \in \mathcal{S} : \pi_n(\vartheta) \neq \pi_{n-1}(\vartheta)\} \tag{22}$$

2) Let $V^{\pi_n}(\vartheta) \to Q_{\pi_n}(\vartheta, \chi)$, and update the policy as:

$$\pi_{n+1}(\vartheta) = \arg\max_{\chi} Q_{\pi_n}(\vartheta, \chi), \quad \forall \vartheta \in \mathcal{S}. \tag{23}$$

3) Terminate the process if $\pi_{n+1}(\vartheta) = \pi_n(\vartheta), \forall \vartheta \in \mathcal{S}$.

Based on the policy iteration method, obtaining the optimal policy relies on knowing the prior knowledge of the transition probability $P_{\vartheta\vartheta'}$. Because $P_{\vartheta\vartheta'}$ is not available in practice, the system reward in (19) cannot be obtained even given a certain policy π. Although a big-data based method can provide prediction by centrally analyzing and processing large-scale request information, it is not applicable to our proposed system which holds strict requirements for decentralization and easy implementation. This motivates us to resort to reinforcement-learning based methods to achieve the optimal policy.

RL is an important branch of machine learning. It is defined by characterizing a learning problem and operates based on its own experience of the environmental uncertainty [26]. An RL problem can be described as an optimal control of an MDP. One of the classic RL algorithms is Q-learning algorithm, which is considered to be suitable and powerful for some problems with tough situations and real-world complexity. In the following, we will present how we solve the problem with the Q-learning algorithm.

B. Standard Q-Learning for Small-Scale Caching

The key component of the Q-learning algorithm is an online learning mechanism which can properly determine the Q-value for each state-action pair. The learning task is handled by a learning agent holding a $|\mathcal{S}| \times |\mathcal{A}|$ dimension Q-table $Q[\mathcal{S}, \mathcal{A}]$. The Q-table stores the reward value (i.e., Q-value) for each state-action pair. The size of the Q-table determines the convergence rate of the learning system. To describe how Problem (I) is solved via Q-learning, we firstly consider a network with a small-scale content set (e.g. a network with $F = 10$ and $N_C = 2$).

Algorithm 1: Optimal Caching With Standard Q-Learning.

Initialize: $\vartheta(0) := \{\mathbf{P} \in \mathcal{P}, \eta \in \mathcal{D}\}$, $Q_0(\vartheta, \chi) = 0$, $\forall \vartheta \in \mathcal{S}, \forall \chi \in \mathcal{A}$.
for $t = 1$ to T **do**
  Choose a random probability ξ.
  if $\xi < \epsilon$ **then**
    Randomly select an action $\chi(t) \in \mathcal{A}$.
  else if $(1 - \xi) < \epsilon$ **then**
    Select action $\chi(t) = \arg\max_{\chi \in \mathcal{A}} Q(\vartheta(t-1), \chi)$.
  end if
  Execute action $\chi(t)$, and observe state $\mathrm{P}(t)$ and $\eta(t)$.
  Calculate the reward $\overline{\mathcal{R}}(\vartheta(t-1), \chi(t) | \mathrm{P}(t), \eta(t))$.
  Update the Q-value:
  $Q_t(\vartheta(t-1), \chi(t)) = (1-\gamma) Q_{t-1}(\vartheta(t-1), \chi(t)) + \gamma \left[\overline{\mathcal{R}}(\vartheta(t-1), \chi(t)) + \beta \max_{\chi \in \mathcal{A}} Q_{t-1}(\vartheta(t), \chi) \right]$
  Keep the rest entries of Q-table unchanged.
end for

The goal of Q-learning is to obtain the optimal state-action value table $Q^*[\mathcal{S}, \mathcal{A}]$ which infers the optimal policy π^*. Then policy π^* instructs the agent to take actions under certain states. Given the optimal Q-table $Q^*[\mathcal{S}, \mathcal{A}]$, the optimal policy π^* can be determined by

$$\pi^*(\vartheta) = \arg\max_{\chi \in \mathcal{A}} Q^*(\vartheta, \chi), \quad \forall \vartheta \in \mathcal{S}. \tag{24}$$

And the maximum state value over all possible actions can be determined by

$$V^{\pi^*}(\vartheta) = \max_{\chi \in \mathcal{A}} Q^*(\vartheta, \chi), \quad \forall \vartheta \in \mathcal{S}. \tag{25}$$

Hence, the optimal Q-value can be defined as

$$Q^*(\vartheta, \chi') = \overline{\mathcal{R}}(\vartheta, \chi') + \beta \sum_{\vartheta' \in \mathcal{S}} P_{\vartheta\vartheta'} \max_{\chi \in \mathcal{A}} Q^*(\vartheta', \chi) \tag{26}$$

To find the optimal Q-value function, the learning agent adjusts the Q-table in a recursive manner using an update rule. Given current state ϑ, Q-learning takes action χ' at the decision phase of the next time slot. Then it observes ϑ' and obtains the system reward $\overline{\mathcal{R}}(\vartheta, \chi')$ at state ϑ'. The Q-function is updated using the stochastic gradient descent [26] as

$$\begin{aligned} Q_n(\vartheta, \chi') &= (1-\gamma) Q_{n-1}(\vartheta, \chi') \\ &\quad + \gamma \left[\overline{\mathcal{R}}(\vartheta, \pi(\vartheta')) + \beta \max_{\chi \in \mathcal{A}} Q_{n-1}(\vartheta', \chi) \right], \end{aligned} \tag{27}$$

where γ is a learning parameter. $\gamma = 0$ makes the agent exclusively exploit previous knowledge, and $\gamma = 1$ makes the agent only exploit the latest information.

During the learning process, given a certain state, the learning agent selects an action according to (24), so that the expected system reward is maximized. This process is called "exploitation". Moreover, each action must be explored enough times to avoid the algorithm converging into a local optimum. This process

is called "exploration". The ϵ-greedy approach is commonly used to select actions in Q-learning, where the exploration and exploitation are balanced by an exploration value $\epsilon \in (0,1)$ [23]. ϵ indicates the probability of taking a random action. The Q-learning based caching is described in Algorithm 1.

C. Approximate Q-Learning for Large-Scale Caching

Despite Q-learning guarantees the optimality for the small-scale content set, its applicability in a large-scale content set faces challenges. A large-scale content set provides the agent with the challenge of "curse of dimensionality". The Q-table in the standard Q-learning model has space size $|\mathcal{P}||\mathcal{D}||\mathcal{A}|$, where $|\mathcal{A}| = \binom{F}{N_C}$. The system action consists of F bits binary vectors. In the standard Q-learning, the action space increases exponentially with the number of contents F. The huge number of state-action pairs in the Q-table makes the learning convergence rate unacceptably slow. To reduce the learning complexity, we propose to reduce the updating of state-action values from exponential space size to linear space size. In the following, we reconfigures the Q-learning algorithm with linear approximation.

The linear approximation for updating the Q-table is inspired by relaxing Problem I. In the optimization problem, constraint (17b) couples the caching actions. Caching content f_k means another cached content f_l has to be replaced at the same time slot. To decouple the caching actions, we optimistically assume that constraint (17b) is always satisfied. Without constraint (17b), the actions for different contents at the same time slot are irrelevant. Then, we obtain the maximal potential reward $\hat{V}^{\pi} = \sum_{k=1}^{F} V_k^{\pi}(\chi_k)$ which is an upper bound of V^{π}, and $\hat{V}_k^{\pi}$ is separable w.r.t. k.

Accordingly, the expected rewards for caching multiple contents can be obtained separately. Hence, the Q-value can be rewritten as the system reward for caching each content, which is denoted by $Q^k(\vartheta, \chi_k), \forall f_k \in \mathcal{F}$. Recalling that the storage size should be satisfied so that the caching policy is implementable, the combinational action χ of a BS with storage size C can be split into N_C independent selections. For each selection, the agent selects the entity with the highest $Q^k(\vartheta, \chi_k), \forall f_k \in \mathcal{F}$ as the candidate content to be cached at the next time slot. The indices of the contents to be cached are obtained as

$$\begin{aligned} c_1 &= \arg\max_{f_k \in \mathcal{F}} Q^k(\vartheta, \chi_k), \\ c_2 &= \arg\max_{f_k \in \mathcal{F} \setminus f_1} Q^k(\vartheta, \chi_k), \\ &\cdots \\ c_{N_C} &= \arg\max_{f_k \in \mathcal{F} \setminus \{f_1, f_2, \ldots\}} Q^k(\vartheta, \chi_k) \end{aligned} \tag{28}$$

Then, the n-th updating of the Q-value is reconfigured from recurring a combinational action to recurring multiple actions for different contents concurrently, which is shown as follows:

$$\begin{aligned} Q_n^k(\vartheta, \chi_k') &= (1-\gamma) Q_{n-1}^k(\vartheta, \chi_k') \\ &+ \gamma \left[\overline{\mathcal{R}}^k(\vartheta, \pi(\vartheta')) + \beta \max_{\chi_k \in \{0,1\}} Q_{n-1}^k(\vartheta', \chi_k) \right], \forall f_k \end{aligned} \tag{29}$$

Algorithm 2: Caching Based on Approximate Q-learning.

Initialize: $\vartheta(0) := \{\mathbf{P} \in \mathcal{P}, \eta \in \mathcal{D}\}$, $Q_0(\vartheta, \chi_k) = 0$, $\forall \vartheta \in \mathcal{S}, \forall \chi_k \in \{0,1\}, \forall f_k \in \mathcal{F}$, $\mathcal{V} = \emptyset$.
for $t = 1$ to T **do**
  Choose a random probability ξ.
  if $\xi < \epsilon$ **then**
    Randomly select an action $\chi(t) \in \mathcal{A}$.
  else if $(1-\xi) < \epsilon$ **then**
    while $C > 0$ **do**
      $f_* = \arg\max_{f_k \in \mathcal{F} \setminus \mathcal{V}} Q^k(\vartheta(t-1), \chi_k(t))$
      if $\sum_{f_k \in \mathcal{V}} \chi_k(t) s_k + \min_{f_k \in \mathcal{F} \setminus \mathcal{V}} \{s_k\} > C$ **then**
        break,
      else if $\sum_{f_k \in \mathcal{V}} \chi_k(t) s_k + s_* \leq C$ **then**
        $\chi_*(t) = 1$, $\mathcal{V} = \mathcal{V} \cup f_*$
      end if
    end while
    $\mathcal{V} = \emptyset$
  end if
  Execute action $\chi(t)$, and observe state $\mathrm{P}(t)$ and $\eta(t)$.
  Calculate the reward $\overline{\mathcal{R}}^k(\vartheta(t-1), \chi_k(t) | \mathrm{P}(t), \eta(t))$.
  Update the Q-value according to (29) for each content.
end for

As we mentioned above, the standard Q-learning algorithm has a high-dimension space with state size $|\mathcal{P}||\mathcal{D}|$ and action size $|\mathcal{A}|$. For the considered system with F contents, given a certain state ϑ, the computational complexity for finding the optimal caching action is at most $O(2^F)$, which is tremendously high and unacceptable for practical implementation. The approximate Q-learning algorithm for the network with a large-scale content set is described in Algorithm 2. Compared with the standard Q-learning, the proposed approximate Q-learning reduces the number of parameters from $|\mathcal{P}||\mathcal{D}|\binom{F}{N_C}$ to $|\mathcal{P}||\mathcal{D}|F$. Based on the independent selections in (28), finding the optimal caching action is transformed into finding a suboptimal solution, which reduces the computational complexity from $O(2^F)$ to $O(F)$. Clearly, this linear approximation bears a complexity-accuracy tradeoff, which helps to achieve desirable accuracy in a large-scale caching network within a polynomial time.

VI. Simulation Settings

We consider 5 BSs controlled by 1 SGW. The content size is uniformly distributed in $[8, 10]$ MB. The intracell transmission rates of ST and JT are set as $r_i^{\mathrm{ST}} = 20$ Mb/s and $r_i^{\mathrm{JT}} = 30$ Mb/s, respectively. The transmission rate between BSs is set as $r_i^{\mathrm{NE}} = 120$ Mb/s, and the transmission rate from the content provider to BS b_i is set as $r_i^{\mathrm{SP}} = 60$ Mb/s. For the small-scale caching, the number of contents $F = 100$ and storage size of BS $N_C = 4$; while for the large-scale caching, $F = 1000$ and $N_C = 50$. The content popularity profile follows a six-state markov process with different parameters $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\} = [0.5, 0.6, 0.7, 0.8, 0.9, 1]$, and its transitions follow classical

Markov state transition laws $\mathrm{T_P}$ as below [33]:

$$\mathrm{T}_\alpha = \begin{bmatrix} 0.52 & 0.256 & 0.128 & 0.064 & 0.032 & 0.016 \\ 0.016 & 0.52 & 0.256 & 0.128 & 0.064 & 0.32 \\ 0.032 & 0.016 & 0.52 & 0.256 & 0.128 & 0.064 \\ 0.064 & 0.032 & 0.016 & 0.52 & 0.256 & 0.128 \\ 0.128 & 0.064 & 0.032 & 0.016 & 0.52 & 0.256 \\ 0.256 & 0.128 & 0.064 & 0.032 & 0.016 & 0.52 \end{bmatrix} \tag{30}$$

Through adjusting user movement parameters p_r, p_t, p_d and λ, the request distribution coefficient is set as a six-state markov process with $\{\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6\} = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Also, the state transition probability matrix follows

$$\mathrm{T}_\eta = \begin{bmatrix} 0.752 & 0.128 & 0.064 & 0.032 & 0.016 & 0.008 \\ 0.008 & 0.752 & 0.128 & 0.064 & 0.032 & 0.016 \\ 0.016 & 0.008 & 0.752 & 0.128 & 0.064 & 0.032 \\ 0.032 & 0.016 & 0.008 & 0.752 & 0.128 & 0.064 \\ 0.064 & 0.032 & 0.016 & 0.008 & 0.752 & 0.128 \\ 0.128 & 0.064 & 0.032 & 0.016 & 0.008 & 0.752 \end{bmatrix} \tag{31}$$

The discount factor is set as $\beta = 0.99$, the ϵ-greedy parameter is set as $\epsilon = 0.05$. In the simulations, average values are taken to reduce the randomness in the simulations. To evaluate the performance of the proposed caching strategy, we compare our strategy with the existing caching strategies configured as follows:

1) Proposed strategy with known state transition probabilities (Proposed with known STP): In the strategy, the BSs are assumed to have known the state transition probabilities. At the end of each time slot, each BS pre-caches contents with the prior knowledge of the future request pattern.
2) FemtoCaching with MPC strategy (FemtoMPC): This strategy is derived by modifying the FemtoCaching scheme [10]. In this strategy, the BSs make caching decisions according to historical popularity distribution, and each MT can be served by any of its associated BSs with ST.
3) Joint content placement with MPC and LCD (JPML): This strategy is derived from the popularity-based combinational caching strategy [16], where the cache space of BSs is partitioned to store the most popular contents and the less-popular contents. Meanwhile, the transmission-level cooperation is considered to perform JT.
4) Random caching: The BSs cache contents randomly.

VII. Performance Evaluation

In this section, we construct simulations to evaluate the performance of the proposed caching strategy.

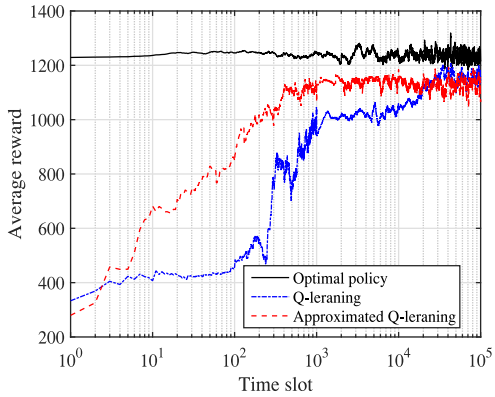


Fig. 4. Convergence performance of Q-learning-based strategies for a small-scale content set.

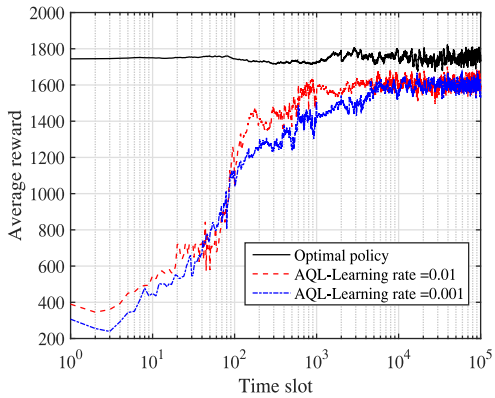


Fig. 5. Convergence performance for a large-scale content set with different learning rate γ.

A. Convergence Performance

The convergence performances of the standard Q-learning (SQL)-based caching strategy and the approximate Q-learning (AQL)-based caching strategy in a small-scale caching network are shown in Fig. 4. The performance of the optimal policy is obtained under known state transition probability of the popularity profile and the request distribution coefficient. From Fig. 4, we can observe that the average rewards of the two learning algorithms are small at the beginning, and then increase as the iterative process continues. After about 500 time slots, the reward of the AQL algorithm firstly converges to a near optimum. The SQL algorithm converges slowly to a near-optimal solution after about 25000 iterations, even for a small-scale content set. It shows that both the two learning algorithms can converge to the stable value. Although the SQL algorithm slightly outperforms the AQL algorithm when they converge to a stable value, it is at the expense of experiencing a large number of iterations. The AQL algorithm has a faster convergence rate and can contribute almost as good performance as the SQL algorithm.

Fig. 5 shows the convergence performance of the AQL algorithm for a large-scale content set with different learning rates. We observe that the average reward of the AQL algorithm increases until it reaches a stable value of 1600 after about

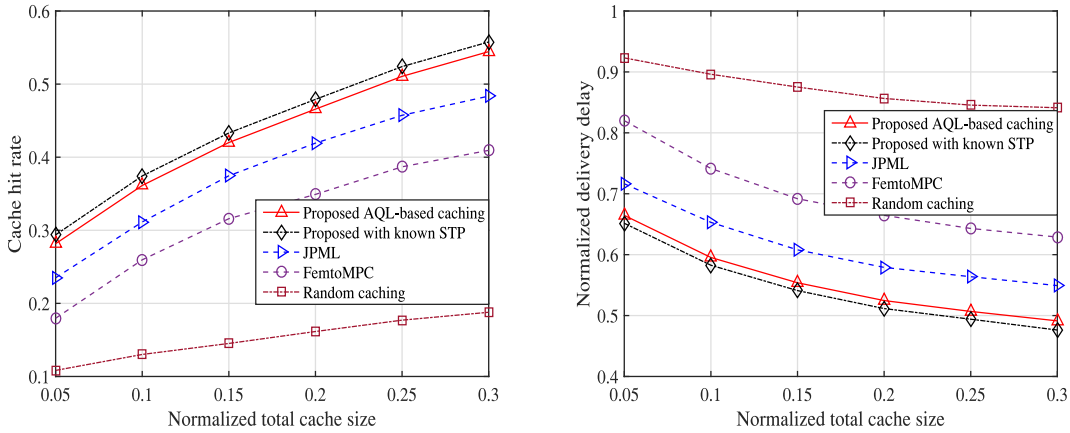


Fig. 6. Cache hit rate (a) and content delivery delay (b) versus total cache size for different caching strategies with fixed total content number $F = 1000$.

4000 time slots with a learning rate $\gamma = 0.001$. By setting a larger learning rate $\gamma = 0.01$, the AQL algorithm reaches the same stable value after about 1000 time slots, which shows that a larger learning rate leads to a much faster convergence rate of the proposed algorithm. Compared with optimal policy, the AQL algorithm provides a near-optimal result at a high convergence rate, showing that the proposed strategy can be applied in practical environments.

B. Performance Comparison

Fig. 6 shows the performance of the proposed caching strategy in terms of cache hit rate and content delivery delay under different normalized cache size. Here, the normalized cache size is denoted as the ratio of total storage space at BSs to the total size of contents in the network, and is set from 5% to 30%. From Fig. 6(a), we observe that the cache hit rates of all the strategies increase with the increased cache size. The proposed caching strategy always holds advantage for improving cache hit rates, and it outperforms the other strategies with up to 5% improvement to the JPML strategy, 15% improvement to the FemtoMPC strategy, and 30% improvement to the random caching strategy. Compared with the JPML strategy, the hit-rate gap achieved by the proposed strategy is contributed by the joint optimization of content placement between BSs. Although the JPML strategy can provide joint transmissions for the popular contents, it does not exploit the storage-level cooperation to optimize the content placement. Therefore, unnecessary content redundancies between BSs lead to low storage utilization, degrading the cache hit performance. The hit-rate gap between the JPML and FemtoMPC strategies indicates that transmission-level cooperation can contribute higher performance gain than cooperative caching. Furthermore, the proposed AQL-based strategy can provide almost the same hit-rate performance as the strategy with known STP, indicating that the AQL algorithm accurately tracks the variation of the user request pattern.

More apparent results can be seen in Fig. 6(b), which evaluates the delay performance of the proposed strategy compared with the other strategies. Here, the normalized delivery delay (NDD) is used. The NDD is denoted as the percentage of content delivery delay under a caching strategy to that without caching. Consistent with the cache hit rate, the NCDs of all the strategies decrease with the increased total cache size, because high cache hit rate can offload more requests from the core to the local network. From Fig. 6(b), the JPML strategy outperforms the random caching strategy because it has opportunities to perform JT, which improves the delay performance from the perspective of transmission. The FemtoMPC strategy outperforms the random caching strategy greatly, because it has more opportunities to caching diverse contents, which improves the cache hit rate from the perspective of storage. However, the JPML strategy always outperforms the FemtoMPC strategy under different caching capacities, demonstrating that transmission-level cooperation contributes more performance gain than storage-level cooperation. Compared with the JPML and FemtoMPC strategies, the proposed AQL-based strategy holds up to 5% and 15% improvement, respectively. One reason for the improvement is that our algorithm can track the change of user request pattern. Moreover, our strategy jointly optimizes the content placement for JT and ST, simultaneously. Therefore, unnecessary content redundancy in the RAN is reduced and the probabilities of performing JT are improved, which further reduces the content delivery delay.

The performance of the proposed strategy compared with the other strategies under the increased number of contents is evaluated in Fig. 7, where Fig. 7(a) and (b) show the performance of cache hit rate and NDD, respectively. The total number of contents in the simulation is set from 300 to 1500. From Fig. 7(a), when the number of contents in the system is small (ranging from 300 to 600), the proposed AQL-based caching strategy contributes at least 60% cache hit rate compared with no caching. It outperforms the JPML strategy with about 8% improvement and the FemtoMPC strategy with about 10% improvement, respectively, for the entire system. The cache hit rates of all the caching strategies decrease when the number of contents increases. This is because when there are more contents

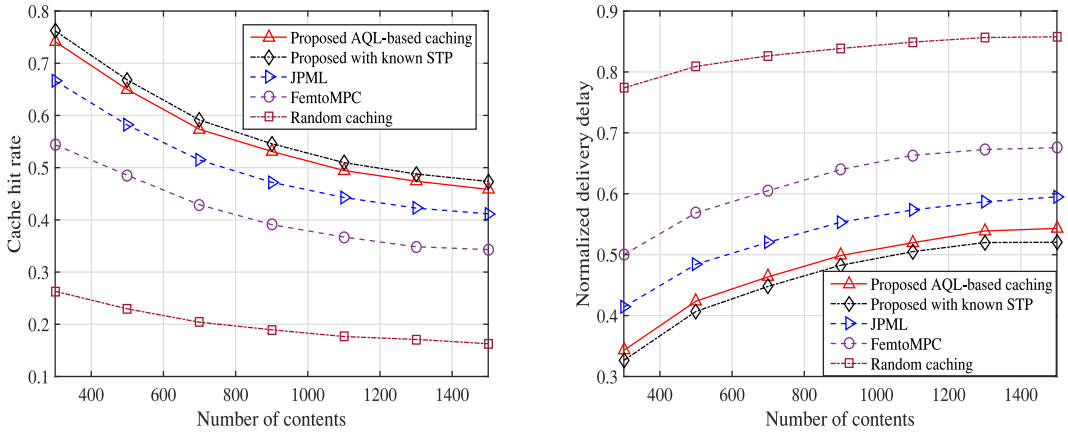


Fig. 7. Cache hit rate (a) and content delivery delay (b) versus total number of contents with fixed cache size of each BS $\mathcal{N}_C = 50$.

throughout the network, the user requests scatter more widely. As a result, the popularity of hot contents is diluted. However, the cache capacities of BSs are too limited to contain more popular contents, which leads to more cache misses in the RAN.

The delay performance of the caching strategies under different numbers of contents is shown in Fig. 7(b). We observe that the NDDs of all the caching strategies go up with the increased number of contents. It is consistent with the analysis of the cache-hit rate that more cache misses in the RAN definitely increase the content delivery delay. From Fig. 7(b), when the number of contents in the system is not large (ranging from 300 to 800), the proposed AQL-based caching strategy can reduce the NDD of the system by at least 50% compared with no caching. Even when the number of contents is sufficiently large (e.g., 1500), the proposed strategy can still reduce the delay by about 45%. Meanwhile, the proposed strategy outperforms the JPML strategy with at least 6% improvement and the FemtoMPC strategy with at least 15% improvement, respectively, in the system with both small-scale and large-scale content sets. It demonstrates that our strategy can always maintain good performance under different scales of caching networks, indicating the stability of the proposed strategy.

VIII. Conclusion

In this paper, we studied the problem of cooperative caching and transmission in the CoMP-integrated cellular network. We proposed the RL-based cooperative caching strategy to minimize the content delivery delay. Compared with the existing caching strategies, our strategy is able to perceive the variation of user request patterns, and exploits two types of cooperation to improve the utilization of storage space. Based on the cooperation strategy, we formulated the content caching problem to be an MDP, in which the system reward was defined as the delay reduction after a caching action. Then, to maximize the system reward, the Q-learning-based algorithm was proposed to learn the optimal policy in an online fashion. In the simulations, we first evaluated the convergence performance of the proposed strategy, and then evaluated its performance in terms of cache hit rate and content delivery delay. The results show that our strategy provides almost the same performance as the optimal policy with faster convergence.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast updata, 2016-2021," Mar. 2017. [Online]. Available: https://www.cisco.com/

[2] F. Zhou, L. Fan, N. Wang, G. Luo, J. Tang, and W. Chen, "A cache-aided communication scheme for downlink coordinated multipoint transmission," *IEEE Access*, vol. 6, pp. 1416–1427, 2018.

[3] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[4] C. Fang, F. R. Yu, T. Huang, J. Liu, and Y. Liu, "A survey of green information-centric networking: Research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 17, no. 3, pp. 1455–1472, Third Quarter 2015.

[5] A. Liu and V. K. N. Lau, "How much cache is needed to achieve linear capacity scaling in backhaul-limited dense wireless networks?" *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 179–188, Feb. 2017.

[6] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1207–1221, May 2016.

[7] S. H. Chae, T. Q. S. Quek, and W. Choi, "Content placement for wireless cooperative caching helpers: A tradeoff between cooperative gain and content diversity gain," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6795–6807, Oct. 2017.

[8] L. Jiang, G. Feng, and S. Qin, "Cooperative content distribution for 5G systems based on distributed cloud service network," in *Proc. IEEE Int. Conf. Commun. Workshop*, Jun. 2015, pp. 1125–1130.

[9] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

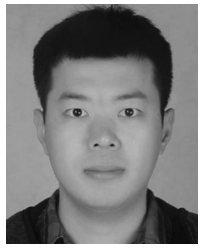
[10] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[11] X. Li, X. Wang, K. Li, Z. Han, and V. C. M. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926–6939, Oct. 2017.

[12] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 131–145, Jan. 2016.

[13] P. Lin, Q. Song, Y. Yu, and A. Jamalipour, "Extensive cooperative caching in D2D integrated cellular networks," *IEEE Commun. Lett.*, vol. 21, no. 9, pp. 2101–2104, Sep. 2017.

[14] Z. Ning *et al.*, "Joint computing and caching in 5G-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2020.2970276.
[15] A. Liu and V. K. N. Lau, "Cache-enabled opportunistic cooperative MIMO for video streaming in wireless systems," *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 390–402, Jan. 2014.
[16] Z. Chen, J. Lee, T. Q. S. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3401–3415, May 2017.
[17] Y. Yu, T. Hsieh, and A. Pang, "Millimeter-wave backhaul traffic minimization for CoMP over 5G cellular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4003–4015, Apr. 2019.
[18] P. Lin, Q. Song, and A. Jamalipour, "Multidimensional cooperative caching in CoMP-integrated ultra-dense cellular networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1977–1989, Mar. 2020.
[19] H. Li, C. Yang, X. Huang, N. Ansari, and Z. Wang, "Cooperative RAN caching based on local altruistic game for single and joint transmissions," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 853–856, Apr. 2017.
[20] P. Lin, K. S. Khan, Q. Song, and A. Jamalipour, "Caching in heterogeneous ultradense 5G networks: A comprehensive cooperation approach," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 22–32, Jun. 2019.
[21] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2432–2455, Fourth Quarter 2017.
[22] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surv. Tut.*, vol. 22, no. 1, pp. 38–67, First Quarter 2020.
[23] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From IoT to 5G I-IoT: The next generation IoT-based intelligent algorithms and 5G technologies," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 114–120, Oct. 2018.
[24] D. Wang, B. Song, D. Chen, and X. Du, "Intelligent cognitive radio in 5G: AI-based hierarchical cognitive cellular networks," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 54–61, Jun. 2019.
[25] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2019.2902661.
[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[27] Z. Ning, P. Dong, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, Dec. 2019.
[28] Z. Ning *et al.*, "When deep reinforcement learning meets 5G-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1352–1361, Feb. 2020.
[29] Y. Guo, F. R. Yu, J. An, K. Yang, Y. He, and V. C. M. Leung, "Buffer-aware streaming in small-scale wireless networks: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6891–6902, Jul. 2019.
[30] H. S. Jo, Y. J. Sang, P. Xia, and J. G. Andrews, "Heterogeneous cellular networks with flexible cell association: A comprehensive downlink sinr analysis," *IEEE Trans. Wireless Commun.*, vol. 11, no. 10, pp. 3484–3495, Oct. 2012.
[31] N. Rajatheva, *5G Mobile and Wireless Communications Technology*. New York, NY, USA: Cambridge Univ. Press, 2016.
[32] K. S. Khan and A. Jamalipour, "Coverage analysis for multi-request association model (MRAM) in a caching ultra-dense network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3882–3889, Apr. 2019.
[33] Y. He *et al.*, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10433–10445, Nov. 2017.



Peng Lin (Member, IEEE) received the M.S. degree in communication and information systems, in 2017, from Northeastern University, Shenyang, China, where he is currently working toward the Ph.D. degree in communication and information systems. He is also a Visiting Student with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada, and the Carleton University, Ottawa, ON, Canada. His current research interests include mobile edge computing, edge caching, and machine learning in wireless networks.

Qingyang Song (Senior Member, IEEE) received the Ph.D. degree in telecommunications engineering from the University of Sydney, Camperdown, NSW, Australia, in 2007. From 2007 to 2018, she was with Northeastern University, China. She joined the Chongqing University of Post and Telecommunications in 2018, where she is currently a Professor. She has authored more than 80 papers in major journals and international conferences. Her current research interests include cooperative resource management, edge computing, vehicular ad hoc network, mobile caching, and simultaneous wireless information and power transfer. She is the editorial boards of two journals, including Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and Technical Editor for Digital Communications and Networks.

Jing Song (Student Member, IEEE) received the B.E. degree in communication engineering from Dalian Maritime University, Dalian, China, in 2012, and the M.S. degree in electronics and communication engineering from Northeastern University, Shenyang, China, in 2018. She is currently working toward the Ph.D. degree with Northeastern University and Chongqing University of Post and Telecommunications, China. Her research interests include radio resource management, mobile edge computing, and caching.

Abbas Jamalipour (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Nagoya University, Nagoya, Japan. He is currently a Professor of Ubiquitous Mobile Networking with the University of Sydney, Australia. He has authored nine technical books, 11 book chapters, more than 450 technical papers, and five patents, all in the area of wireless communications. He is a fellow with the Institute of Electrical, Information, and Communication Engineers (IEICE) and the Institution of Engineers Australia, an ACM Professional Member, and an IEEE Distinguished Lecturer. He is an elected member of the Board of Governors and President of IEEE Vehicular Technology Society. He was the Editor-in-Chief IEEE Wireless Communications, Vice President-Conferences and a member of Board of Governors of the IEEE Communications Society, and has been an editor for several journals and is on the Editorial Board of the IEEE Access. He has been a General Chair or Technical Program Chair for a number of conferences, including IEEE International Conference on Communications, GLOBECOM, WCNC, and PIMRC. He is the recipient of a number of prestigious awards such as the 2016 IEEE ComSoc Distinguished Technical Achievement Award in Communications Switching and Routing, 2010 IEEE ComSoc Harold Sobol Award, the 2006 IEEE ComSoc Best Tutorial Paper Award, as well as 15 best paper awards.

F. Richard Yu (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of British Columbia, Vancouver, BC, Canada, in 2003. From 2002 to 2006, he was with Ericsson (in Lund, Sweden) and a start-up in California, USA. He joined Carleton University in 2007, where he is currently a Professor. He received the IEEE Technical Committee on Green Communications and Computing Best Journal Paper Award, in 2019, Distinguished Service Awards, in 2019 and 2016, Outstanding Leadership Award in 2013, Carleton Research Achievement Award, in 2012, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award), in 2011, the Excellent Contribution Award at IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from Canada Foundation of Innovation, in 2009, and the Best Paper Awards at IEEE International Conference on Computing, Networking and Communications 2018, Vehicular Technology Conference 2017 Spring, International Conference on Communications 2014, Globecom 2012, IEEE/IFIP TrustCom 2009, and Int'l Conference on Networking 2005. His research interests include connected/autonomous vehicles, security, artificial intelligence, distributed ledger technology, and wireless cyber-physical systems. Dr. Yu is with the editorial boards of several journals, including Co-Editor-in-Chief for Ad Hoc and Sensor Wireless Networks, Area Editor for IEEE Communications Surveys and Tutorials, Lead Series Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He was the Technical Program Committee Co-Chair of numerous conferences. He is a registered Professional Engineer in the province of Ontario, Canada, IET Fellow, and Engineering Institute of Canada Fellow. He is an IEEE Distinguished Lecturer of both Vehicular Technology Society (VTS) and Comm. Society. He is an elected member of the Board of Governors of the IEEE VTS.