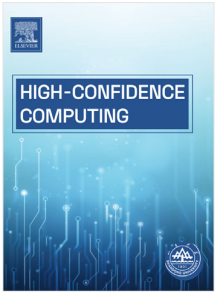
Journal Pre-proof



Learning-based cooperative content caching and sharing for multi-layer vehicular networks

Jun Shi, Yuanzhi Ni, Lin Cai, Zhuocheng Du

PII: S2667-2952(24)00080-1
DOI: https://doi.org/10.1016/j.hcc.2024.100277
Reference: HCC 100277

To appear in: *High-Confidence Computing*

Received date : 12 April 2024
Revised date : 6 June 2024
Accepted date : 25 September 2024

Please cite this article as: J. Shi, Y. Ni, L. Cai et al., Learning-based cooperative content caching and sharing for multi-layer vehicular networks, *High-Confidence Computing* (2024), doi: https://doi.org/10.1016/j.hcc.2024.100277.

Learning-Based Cooperative Content Caching and Sharing for Multi-layer Vehicular Networks

Jun Shi, Yuanzhi Ni, Lin Cai, and Zhuocheng Du

***Abstract*—Caching and sharing the content files is critical and fundamental for various future vehicular applications. However, how to satisfy the content demands in a timely manner with limited storage is an open issue owing to the high mobility of vehicles and the unpredictable distribution of dynamic requests. To better serve the requests from the vehicles, a cache-enabled multi-layer architecture, consisting of a Micro Base Station (MBS) and several Small Base Stations (SBSs), is proposed in this paper. Considering that vehicles usually travel through the coverage of multiple SBSs in a short time period, the cooperative caching and sharing strategy is introduced, which can provide comprehensive and stable cache services to vehicles. In addition, since the content popularity profile is unknown, we model the content caching problems in a Multi-Armed Bandit (MAB) perspective to minimize the total delay while gradually estimating the popularity of content files. The reinforcement learning-based algorithms with a novel Q-value updating module are employed to update the caching files in different timescales for MBS and SBSs, respectively. Simulation results show the proposed algorithm outperforms benchmark algorithms with static or varying content popularity. In the high-speed environment, the cooperation between SBSs effectively improves the cache hit rate and further improves service performance.**

***Index Terms*—Cooperative content caching, MAB, reinforcement learning, multi-layer vehicular networks, high-speed environment.**

I. Introduction

In recent decades, the tremendous growth of mobile devices and Internet applications brings great convenience to daily travel. In the future, the demand for delay-stringent services with high data rates, e.g., multimedia streaming, edge computing and mobile crowdsensing, are expected to increase and impose pressure on vehicular networks with limited capacity. Caching content files in a distributed storage-enabled vehicular network, makes it easier for the users to acquire the content files for further application, computing and sharing. Storage entities in vehicular networks refer to the Base Station (BS), equipped with a reliable backhaul link to the core network, which could help to reduce traffic congestion and alleviate transmission pressure. It is believed that caching popular content files in the BS will lead to significant reductions in energy consumption, bandwidth usage, and costs, while also enhancing user satisfaction [1]–[3].

However, some bottleneck problems significantly affect the content caching efficiency and the implementation of the service network. To begin with, the BS relies on lots of observations to learn when and which content file to cache in order to find the optimal caching strategy. In the second place, how to deploy and manage the BS to facilitate caching and updating in a highly dynamic environment is also a critical issue. At last, since the vehicle may travel through the coverage of multiple SBSs, the total downloading delay under the proper scheduling is the key metric which is determined by the source and routes together. Therefore, an adaptive caching and sharing strategy based on the cooperation between the heterogeneous BSs is significant to improve the Quality of Service (QoS) of the vehicles.

Many recent works are devoted to addressing the above issues. Artificial intelligence are employed to equip the network entities with smart caching units, which could learn, track, and adjust to unknown dynamic environments [4]–[10]. A hierarchical architecture of cache-enabled networks can provide comprehensive and low latency cache services, which can store more abundant content files and improve the QoS of vehicles [11]–[14]. On the other hand, cooperation between SBSs can also provide more content files for vehicles, reduce the requests served directly by the MBS or the cloud [4], [15]–[17]. However, in a high-speed environment, the current technologies may fail to satisfy some requests due to long downloading delay, even with the optimal caching strategy. This paper utilizes the cooperation between SBSs, i.e., the SBS relays the requests with long downloading delay to another SBS on the route of the vehicle. When the vehicle enters the latter SBS's coverage, it will acquire the requested content file quickly. In this paper, we study the content caching and sharing optimization problem in the finite area deployed with several SBSs and an MBS by minimizing the delivery delay of the requested content files in storage-enabled network entities. With the help of learning-based intelligence, the caching strategies of BSs could be optimized collaboratively in each time period. Due to the variability of the environment, BSs tend to learn and optimize the caching strategy online in the MAB framework gradually. Besides, the MBS helps to store more content files and regulate the SBS caching strategies in real-time, coordinated with several SBSs to form a cache-enabled multi-layer network. In the high-speed environment, if the MBS knows the caching strategies of SBSs and the route of the vehicles, cooperative service between SBSs would be possible to improve the QoS and also alleviate the traffic pressure. The main contributions are summarized as follows.

- A cache-enabled multi-layer architecture, consisting of the cloud, an MBS and several SBSs, is proposed to handle the dynamic and unpredictable requests of the vehicular networks. In addition, based on the proposed architecture and different caching timescales, the content caching problems aiming at minimizing the total down-

loading delay are formulated for both the SBSs and the MBS from an MAB perspective.
- Considering the different capabilities and caching timescales of SBS and MBS, the learning-based algorithm MCUCB is proposed to adapt to the unknown and varying popularity of the content files, and cache the most suitable files in the SBS and MBS, which can provide comprehensive and stable cache services.
- To further improve the caching efficiency in the high-speed environment, for the requests that could not be served locally, cooperative MCUCB algorithm is introduced to utilize the traffic information and the cooperation between the deployed infrastructures, which further improves the QoS.

The remainder of this paper is organized as follows. Section II reviews the existing work. Section III introduces the proposed system model and the caching and sharing optimization problem is formulated in Section IV. Section V presents the details of the proposed reinforcement learning-based caching policy. Section VI introduces the simulation results followed by the conclusions and the future work in Section VII.

II. Related Work

Caching content files in the BS to increase the QoS of vehicles has attracted great attention from both the industry and academia. In this section, recent research will be reviewed and classified into those with known content popularity profile and those without.

A. Caching strategies for known content popularity profile

Extensive works have studied the content caching problem with the known content popularity profile. In [18], BS cooperation in the radio access is considered and a low-complexity algorithm for the content caching problem is proposed to minimize the average downloading delay over user requests. In [19], comparing the uncoded case and the coded case of the video files, the uncoded case is proved to be NP-hard, and authors develop a greedy strategy for the coded optimum cache assignment problem. In [20], SBS is considered always active or activated on demand of mobile users, and the authors investigate the probabilistic small-cell caching strategy of the above cases. In [21], in the context of wireless networks with varying link rates, a new network coding scheme called nested coded modulation is utilized for the delivery phase, and for the placement phase, a novel file partition scheme is proposed, which is based on the allocation of unequal cache sizes.

B. Caching strategies for unknown content popularity profile

Practically, the content popularity profile is usually unknown in real life, thus more related works are devoted to addressing the content caching problem without the knowledge of content popularity.

Previous works addressed the content caching problem with the least frequently used (LFU), least recently used (LRU), first in first out (FIFO), random replacement (RR), and myopic algorithms, which are usually ineffective to cope with the dynamics of content popularity. Recent works tend to develop learning and optimization-based algorithms that can learn to cache content files. In [22], the content caching problem is modeled as an MAB problem, and then an approximation solution is proposed to solve the knapsack problem. In [23], from an MAB perspective, the authors study the content caching problem by jointly optimizing content caching in cooperative BSs, and propose a centralized algorithm and a distributed algorithm for the content caching problem.

However, in real life, the content popularity exhibits dynamics, and reinforcement learning has advantages in dealing with the dynamics of the content caching problem. In [4], both global and local popularity are considered, and the authors propose a Q-learning algorithm to learn the caching strategies and adapt the dynamics of content popularity. In [5], the simple but flexible generic time-varying fetching and caching costs are introduced to minimize aggregate cost across files and time, and the Q-learning algorithm is employed to find the optimal fetch-cache decisions. DRL algorithms have been applied for content caching in, e.g., [6]–[10]. In [6], to solve the content caching problem in a timely and efficient manner, a pre-trained Deep Neural Network (DNN) is used to train the optimization algorithm, which can reduce the complexity in the delay-sensitive operation phase. In [8], to adapt to the dynamics of the content cache, based on the Actor-Critic framework, the authors propose a novel size adaptive content caching algorithm.

Furthermore, a hierarchical architecture to provide content files could also help to improve the QoS of vehicles, which has become a common practice in recent works [11]–[14]. In [11], a parent node is connected to multiple leaf nodes to serve user requests for content files, and the authors propose a reinforcement learning framework to model the bidirectional impact between caching decisions made at parent and leaf nodes. In [13], the cloud servers are deployed at the network edge and the edge cloud is designed as a tree hierarchy of geo-distributed servers, to better satisfy the peak loads from the mobile users.

Nevertheless, related works above tend to cache the most popular content files, which usually ignores those relatively unpopular content files and reduces the hit rate. In addition, all requests to be served locally is not always the optimal choice considering the different delay requirements and priorities. In this paper, the most popular content files are cached in the local SBS, and the cooperative service between SBSs is introduced to serve the requests for the relatively unpopular content files. A cache-enabled multi-layer architecture is expected to serve the most requests from vehicles, and the cooperative service between SBSs can ensure that vehicles can acquire the requested content files eventually.

III. System Model

A. Network Model

The system model of the content caching network is illustrated in Fig. 1. We consider a two-layer interconnected caching network, where the MBS is connected to U SBSs.

An MBS associates with several cache-enabled SBSs in its coverage, denoted as $\mathcal{U} = \{1, 2, \ldots, U\}$. It is assumed that the SBS u has a limited communication range l_u, which means that the vehicle can communicate with the SBSs when the distance $l_{u,v}$ between the SBS u and vehicle v is less than l_u. In this paper, in the finite area, the MBS can communicate with all the SBSs in this area, but not directly with vehicles.

TABLE I
SYSTEM PARAMETERS

Notation	Definition
$\mathcal{U}$	SBSs set
l_u	Communication range of SBS u
$l_{u,v}$	Communication distance between the SBS u and vehicle v
$l_{m,u}$	Communication distance between the MBS and SBS u
$\mathcal{F}$	Files set
s_f	Size of content file f
θ_f	Popularity of content file f
γ	Parameter of the Zipf distribution
A_u	Cache matrix of SBS u
A_m	Cache matrix of the MBS
S_u	Cache size of SBS u
S_m	Cache size of the MBS
P_{SBS}	Transmission power of SBS
P_{MBS}	Transmission power of MBS
$B_{u,v}$	Bandwidth of SBS-vehicle links
$B_{m,u}$	Bandwidth of MBS-SBS links
$\tau_{u,v}$	Transmission delay from the SBS u to the vehicle v
$\tau_{m,u}$	Transmission delay from the MBS to the SBS u
$\tau_{c,m}$	Transmission delay from the cloud to the MBS

B. Service Model

An MBS and several SBSs in this network store content files to serve requests from vehicles. When the vehicle enters the SBS's coverage and requires a content file from this SBS, the SBS serves the request if it has cached the content file. Otherwise, the SBS will appeal to the MBS for the requested content file and then return it to the vehicle. In case the MBS has not cached the requested content file, the MBS has to communicate with the cloud to acquire the content file, which will be relayed to the MBS and SBS before arriving at the corresponding vehicle.

The service time is split into several periods, and each period consists of a user request phase followed by a cache replacement phase. During the user request phase, we use $d_{u,f}^t$ to represent the number of times vehicles request content file f from SBS u within period t, which is assumed to be an independent identically distributed (i.i.d.) random variable with a mean value of $\theta_{u,f} = E(d_{u,f}^t)$. It is assumed that in each time period, the SBS's requests received from the vehicles occur independently following the Poisson process with an average rate of N_u^t. In addition, it is assumed that the popularity of the files follows a Zipf distribution with parameter γ, $\theta_f = \frac{1}{f^\gamma \sum_{i=1}^{F} \frac{1}{i^\gamma}}$, and the popularity set of all files is denoted as $\Theta = (\theta_1, \ldots, \theta_F)$, which usually reflects the frequency distribution of word occurrences in textual data. Hence, the popularity of content file f, which is the expected number of requests for content file f, can be described as $\theta_{u,f} = \frac{N_u^t}{f^\gamma \sum_{i=1}^{F} \frac{1}{i^\gamma}}$, $u \in \mathcal{U}$, $f \in \mathcal{F}$, where $\mathcal{F} = \{1, 2, \ldots, F\}$ is the content file set. Notice that the parameter γ describes the skewness of the popularity distribution. When $\gamma = 0$, the popularity is evenly distributed across files, and as γ increases, the skewness of the popularity intensifies.

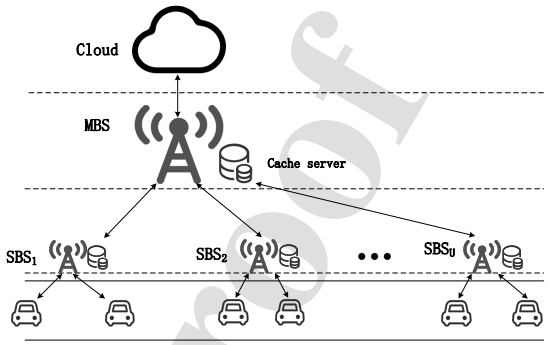


Fig. 1. system model

C. Cache Model

Each content file is of size s_f, $f \in \mathcal{F}$, which follows a power-law distribution. Each SBS is equipped with a local cache size of S_u, and the caching strategy of the SBS is denoted as $A_u = \{(a_{u,1}, a_{u,2}, \ldots, a_{u,F}) | a_{u,f} \in \{0, 1\}, f \in \mathcal{F}\}$, which should fulfill the cache size constraint $\sum_{f=1}^{F} a_{u,f} \cdot s_f \leq S_u$, $\forall u \in \mathcal{U}$.

Different from the SBS, the MBS m has a much bigger local cache size S_m so that the MBS can cache more files. As with SBS, the caching strategy of the MBS is denoted as $A_m = \{(a_{m,1}, a_{m,2}, \ldots, a_{m,F}) | a_{m,f} \in \{0, 1\}, \sum_{f=1}^{F} a_{m,f} \cdot s_f \leq S_m, f \in \mathcal{F}\}$.

D. Transmission Model

In this paper, the total transmission delay of acquiring the content file is employed as the performance metric. When the vehicle requires a content file from its nearby SBS, it will get the content file after a transmission delay $\tau_{u,v}$ if the SBS caches the requested content file. In the noise-limited network, we model the transmission delay neglecting the interference and considering only the large-scale fading. Applying the signal-to-noise ratio (SNR) to estimate the transmission rate, the SNR of SBS-vehicle communication links can be given as $SNR_{u,v} = P_{SBS} \cdot l_{u,v}^{-\alpha} / \sigma_N^2$, where σ_N^2 is the white Gaussian noise power, α is the path loss exponent, P_{SBS} denotes the transmission power of SBS, and $l_{u,v}$ represents the communication distance from the SBS u to the vehicle v. Therefore, the transmission delay for transmitting a content file with unit size from SBS to the vehicle can be computed as $\tau_{u,v} = 1/[B_{u,v} \log_2 (1 + SNR_{u,v})]$. Similarly, the SNR of MBS-SBS communication links can be expressed as $SNR_{m,u} = P_{MBS} \cdot l_{m,u}^{-\alpha} / \sigma_N^2$, where P_{MBS} denote the transmission power of MBS, and $l_{m,u}$ represents the communication distance from the MBS m to the SBS u, and the transmission delay for transmitting a unit-size content file from the MBS to the SBS is given by $\tau_{m,u} = 1/[B_{m,u} \log_2 (1 + SNR_{m,u})]$.

Furthermore, the transmission delay between the cloud and the MBS would be considered as $\tau_{c,m} \gg \tau_{m,u}$.

E. Caching Refreshing Model

To better serve the requests from vehicles, the SBS refreshes its caching queue in terms of the "considered" locally popular content files, and the MBS caches the content files that are not locally cached in the SBS. Since the SBSs are closer to the vehicles, they receive requests frequently which exhibit rapid temporal evolution at a short timescale. On the other hand, the MBS observes requests from several SBSs, which exhibit smaller fluctuations and experience the evolution at a longer timescale.

We adopt a two-timescale approach to manage the network model consisting of multiple SBSs and an MBS. The long timescale is denoted as $T = 1, 2, \ldots$, each of which could be divided into n short time periods $t = 1, 2, \ldots, n$. The short time period is assumed to be 1 to 2 minutes based on the dynamics of requests, while the long time interval is 8 to 10 minutes depending on the dynamics of the requests from the SBSs. It is assumed that the network model will not change during the short time period t, but change between t and $t+1$.

IV. Problem Formulation

In this section, we model the SBS caching problem and the MBS caching problem as the MAB problem. Specifically, learning from the requests received and the historical downloading delay for the vehicles, the caching strategy for the SBSs and the MBS is adjusted to minimize the total downloading delay for the vehicles over a long time period.

A. Non-Cooperative Service for SBS Caching

For a vehicle v entering the SBS's coverage, it may require content file f from the SBS, and the requests will be satisfied by the nearby SBS, the MBS, or the cloud. If the nearby SBS has cached the requested file f, the vehicle will be served by the nearby SBS with the downloading delay $\tau_{u,v,f}$. If the nearby SBS has not cached the requested file f, the SBS will request the MBS for the file f, and then the total downloading delay will be $\tau_{u,v,f} + \tau_{m,u,f}$ if the MBS has cached the requested file f. At last, if neither the nearby SBS nor the MBS has cached the requested file f, the MBS will request the file f from the cloud, which will experience a much longer delay $\tau_{c,m,f}$, and then the total downloading delay will be given as $\tau_{u,v,f} + \tau_{m,u,f} + \tau_{c,m,f}$. It is assumed that the downloading delay from the cloud will be lower than that from the other SBSs, thus the case that the vehicle requests the local SBS for the content file relayed from the other SBSs will not be considered in this paper.

Thus, the downloading delay of SBS u for content file f is

$$D_{u,f} = \tau_{u,v,f} + (1 - a_{u,f}) \left[\tau_{u,m,f} + (1 - a_{m,f}) \tau_{c,m,f}\right]. \quad (1)$$

In time period t, $d^t_{u,f}$ represents the number of requests SBS u received for content file f, Thus the total downloading delay of SBS u in period t can be computed as the sum of all individual delays $D_{u,f}$ for each content file f.

$$D^t_u = \sum_{f=1}^{F} d^t_{u,f} \left\{\tau_{u,v,f} + (1 - a_{u,f}) \left[\tau_{u,m,f} + (1 - a_{m,f}) \tau_{c,m,f}\right]\right\}. \quad (2)$$

In this section, the objective of the content caching problem is to minimize the total downloading delay of the SBS over a long time period N by learning and finding the optimal caching strategy when the preference for the content file is unknown in advance. The SBS caching optimization problem can be expressed as follows:

$$\mathbf{P1}: \min_{\{A_u\}} \sum_{t=1}^{N} D^t_u \quad (3a)$$

$$\text{s.t. } a_{u,f} \in \{0, 1\}, \ \forall f \in \mathcal{F}, \quad (3b)$$

$$\sum_{f=1}^{F} a_{u,f} \cdot s_f \leq S_u, \ u \in \mathcal{U}. \quad (3c)$$

The optimization problem **P1** is a classical caching optimization problem. In the real world, we usually cannot obtain the popularity information in advance and cache the content files accordingly. Several previous work took the approach to estimate the popularity profile, $\Theta = (\theta_1, \ldots, \theta_F)$, and then optimize the caching strategy. In this paper, instead, we directly learn the caching strategy with a low-complexity algorithm to make sequential caching decisions. In this section, the reinforcement learning-based method is employed to solve the sequential decision-making problem from an MAB perspective.

In the formulated SBS caching problem, the cache server is considered as a gaming machine, which is a collection of M "arms", and the agent chooses which arm to pull, that is, the cache server chooses which content file to cache. The strategy aims to find the optimal choice which maximizes the cumulative reward among multiple choices in a limited number of times. Since over a period of time, the SBS will cache several content files instead of only one, we choose to apply the variant of the classical MAB model, i.e., the combinatorial MAB (CMAB), where we can pull $m < M$ arms over a period of time [24].

At first, the reward for caching content file f in the SBS is defined as the reduction in downloading delay compared to the case without caching. If the SBS u does not cache the content file f, the downloading delay is described as

$$D_{u,f}{}' = \tau_{u,v,f} + \tau_{u,m,f} + (1 - a_{m,f}) \tau_{c,m,f}. \quad (4)$$

Then we can get the reward for caching content file f in the SBS as

$$Y_{u,f} = D_{u,f}{}' - D_{u,f}. \quad (5)$$

After defining the reward, **P1** is reformulated as a CMAB

problem **P2**.

$$\mathbf{P2}: \max_{\{A_u\}} \sum_{t=1}^{N}\sum_{f=1}^{F} d_{u,f}^{t} Y_{u,f} \tag{6a}$$

$$\text{s.t. } a_{u,f} \in \{0,1\},\ \forall f \in \mathcal{F}, \tag{6b}$$

$$\sum_{f=1}^{F} a_{u,f} \cdot s_f \le S_u. \tag{6c}$$

In the formulated model, we mainly focus on how to place the most popular content file in the BS. The cost of replacing content files with new required files is not directly included in the objective function for the following two reasons. First, the cost for replacing the content files is a necessary overhead to maintain the content sharing efficiency regardless which strategy is employed. Even the cost is relatively large in certain situations, the facilities including the MBS and SBS will always update the cache to avoid more cost in case the file cannot be obtained directly. Therefore, the cache replacing cost is not the main concern in the formulated problem. Second, the cost for replacing cached files is mainly correlated with the size of files. As described in problem **P2**, considering the storage limitation of BSs, the reward per unit size has already been considered as the performance indicator for the caching strategy to incline those files with smaller sizes. Thus, the proposed caching policy in Section V also favors the solutions with lower replacing cost during the optimization process.

B. Cooperative Service for SBS Caching

Considering the high speed of vehicle movement, vehicles usually move out of the transmission range of the local SBS that received its request before acquiring the desired content file. Many situations may result in content files failing to be returned to the vehicles, e.g., the size of the content file is too large to be returned in the limited time period or the content file is in the MBS or the cloud leading to a long downloading delay. Therefore, we prefer to cache these content files in the SBS in advance to serve the corresponding requests. However, the fact is that we cannot cache all the content files in the SBS before it receives the requests. In this section, based on the non-cooperative service for SBS caching, cooperative service for SBS caching is proposed to address the above concerns and improve the QoS of serving requests from vehicles.

As illustrated in Fig. 2, to serve these requests fail to be satisfied in the local SBS i, the collaboration between SBSs could be employed to solve this problem. When the request is relayed to the MBS, the MBS could compute the downloading delay in order to determine whether the request should be relayed to other SBSs. It is assumed that the MBS knows the locations of all the SBSs and the route plan in this area. If the vehicles fully trust the MBS, which means the vehicles will not hide their driving routes from the MBS, the MBS could assign these requests to the corresponding SBS on the route of the vehicles. More precisely, according to the instruction of the MBS, SBS j could cache the corresponding content file in the previous time period. Several time periods later, the vehicle v will receive the desired content file from SBS j.

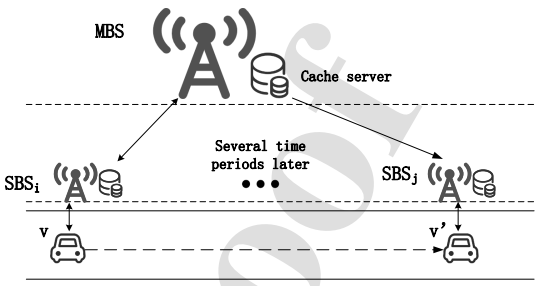


Fig. 2. Cooperative service for SBS caching

In the user request phase, the SBS could receive requests from the vehicles in this area and the vehicles moving from other areas. According to the local cache in the SBS, SBS will return the content file to the vehicles, and the remaining requests will be relayed to the MBS. After computing the downloading delay of the requests, MBS will serve the requests that could be satisfied in this SBS, and relay the other requests to the corresponding SBSs in the following time period. In the cache replacement phase, the SBS would refresh its content file based on the requests in this time period and the requests from other areas.

The reward for caching content file f is defined in (5), and the local SBS u should pay a part of reward $Y_{u,f}$ to the SBS w who fulfills the request for content file f. Therefore, the reward of SBS u will be $(1-r)\cdot Y_{u,f}$, and the reward of SBS w will be $r \cdot Y_{u,f}$, where $0 \le r \le 1$ is the proportional factor.

In time period t, for SBS u, we use $d_{u,u,f}^{t}$ to denote the number of requests satisfied in the SBS u itself, $d_{u,w,f}^{t}$ to denote the number of requests which could not be satisfied in the SBS u and be relayed to the SBS $w \in \mathcal{U}, w \ne u$, and $d_{w,u,f}^{t}$ to denote the number of requests relayed to the SBS u from SBS $w \in \mathcal{U}, w \ne u$. Thus, the reward SBS u receives in time period t can be given by

$$R_{u,f}^{t} = d_{u,u,f}^{t} Y_{u,f} + \sum_{\substack{w\in\mathcal{U}\\ w\ne u}} d_{u,w,f}^{t} r Y_{u,f} + \sum_{\substack{w\in\mathcal{U}\\ w\ne u}} d_{w,u,f}^{t} (1-r) Y_{u,f} \tag{7}$$

After introducing the cooperative service for SBS caching, the SBS caching problem can be given as

$$\mathbf{P3}: \max_{\{A_u\}} \sum_{t=1}^{N}\sum_{f=1}^{F} R_{u,f}^{t} \tag{8a}$$

$$\text{s.t. } a_{u,f} \in \{0,1\},\ \forall f \in \mathcal{F}, \tag{8b}$$

$$\sum_{f=1}^{F} a_{u,f} \cdot s_f \le S_u. \tag{8c}$$

The optimization problem **P3** is a CMAB problem, which is challenging to solve. First, the popularity profile is unknown in advance, thus this problem cannot be directly solved by optimization or game theory. In addition, the selection of action is difficult due to the uncertainty about the expected reward associated with various actions. Second, with the

introduction of cooperative service, the SBS selection and the action selection are coupled with each other and the feasible solution domain increases exponentially. Thus, the traditional searching algorithms cannot be directly applied to solve the formulated problem while maintaining a high efficiency.

C. Problem Formulation for MBS Caching

In this section, the MBS caching problem is also modeled as an MAB problem with a longer cache refreshing timescale. It is worth noted that since the SBS keeps learning and optimizing the caching strategies, the requests for the MBS also evolves accordingly and constantly. Therefore, the MBS caching encounters a more complicated and dynamic problem. At the start of each time interval T, the MBS refreshes its cache queue, and then the MBS serves the requests from SBSs to help the content file delivery. Different from SBSs, MBS receives requests from the SBSs and decides its caching strategy based on the historical requests from the SBSs, to reduce the requests served by the cloud.

First, for the requests from the SBS u, the downloading delay of MBS m of content file f is

$$D_{m,u,f} = \tau_{u,v,f} + \tau_{u,m,f} + (1 - a_{m,f})\,\tau_{c,m,f}, \tag{9}$$

which is the same as equation (4) and the downloading delay if MBS does not cache the content file f is described as

$$D_{m,u,f}{}' = \tau_{u,v,f} + \tau_{u,m,f} + \tau_{c,m,f}, \tag{10}$$

which is actually the downloading delay from the cloud to the corresponding vehicle.

The number of the requests for content file f from SBS u in time interval T is denoted as $d^T_{m,u,f}$, then the reward of MBS for serving the content file f requests from the SBS u in time interval T can be described as,

$$R^T_{m,u,f} = d^T_{m,u,f}(D_{m,u,f}{}' - D_{m,u,f}). \tag{11}$$

In the end, with the reward for the requests from each SBS, the MBS caching problem can be expressed as:

$$\mathbf{P4}: \max_{\{A_m\}} \sum_{T=1}^{\infty} \sum_{f=1}^{F} \sum_{u=1}^{U} R^T_{m,u,f} \tag{12a}$$

$$\text{s.t. } a_{m,f} \in \{0,1\},\ \forall f \in \mathcal{F}, \tag{12b}$$

$$\sum_{f=1}^{F} a_{m,f} \cdot s_f \le S_m. \tag{12c}$$

V. Reinforcement Learning-Based Caching Policy

A. Cooperative SBS Caching Strategy

Reinforcement learning is applied by agents to learn and optimize their actions in the SBS caching optimization problem. The SBS caching policy is designed based on Q-learning, an effective reinforcement learning algorithm. The main idea of Q-learning is to use Q-value to represent the expected cumulative reward that agents receive for taking a specific action in a given state. However, the environment cannot always be represented by states and only the action space is taken into consideration. In such cases, the standard Q-learning method is simplified to its stateless version [25]. In this paper, it is obvious that we cannot describe the environment in terms of finite states, and then the stateless Q-learning is considered. Therefore, the reinforcement model comprises three components, i.e., agent, action, and reward. Here, the agent is the SBS, actions are its caching decisions and the rewards are reductions of the downloading delay. For SBS u, the Q-value $Q_u(a_u)$ is defined to estimate the effectiveness of executing action a_u in the next time period, and can be updated after receiving the reward of the execution.

For SBS u, the caching decision employed at each time period t can be considered as the super action. First, we can treat the super action as an action, and then simply use the traditional Q-learning to solve the optimization problem **P2**. In the stateless setting, $Q_u(a_u),\ a_u \in A_u$ is denoted as the estimated reward of executing super action a_u in the next time period t. At each time period t, SBS u updates $Q_u(a_u),\ a_u \in A_u$ with the reward of executing super action a_u. The update equation for Q-value in time period t is defined as [26]:

$$Q_u(a_u) \leftarrow Q_u(a_u) + \alpha^t \left(R^t_u - Q_u(a_u)\right), \tag{13}$$

where R^t_u is the reward associated with the super action executed in time period t, which can be computed by equation (5). $\alpha^t \in \{0,1\}$ is the learning rate, which is used to control the effect of reward on current Q-value.

However, the traditional Q-learning algorithm raises two issues in solving the formulated problem. Firstly, if the super action a_u is treated as an action, we need to decide whether to cache each content file f or not, which will lead to a large action space of 2^F. Owing to the combinatorial explosion, 2^F is exponential to the number of content files F, thus it takes a long time to traverse the entire action space. Secondly, the reward R^t_u can be shared by several super actions. This is due to that when the content file f is requested in time period t, part of the reward R^t_u will be shared by the super actions which contain $a_{u,f} = 1$. In other words, the outcome of super action a_u is the sum of underlying actions $a_{u,f},\ f \in \mathcal{F}$, and we cannot distinguish the content file with high rewards. Therefore, it is ineffective to employ the traditional Q-learning algorithm to solve **P2**.

For the MAB problem, the upper confidence bound (UCB) algorithm balances exploration and exploitation, and can achieve relatively good performance. Specifically, this algorithm selects actions by balancing the potential to exploit known actions with high payoffs and the potential to explore unknown actions. Firstly, the counter and cumulative reward are initialized for each action to 0. Secondly, in time period t, for each action i, its upper bound value is calculated as $\mu_i = \hat{\mu}_i + \sqrt{\frac{\ln t}{n_i}}$. $\hat{\mu}_i$ is the average reward, and $\sqrt{\frac{\ln t}{n_i}}$ is the upper confidence interval where t is the time period number and n_i is the number of times the action i is selected. After calculating the upper bound, we select the action with the maximum upper bound value and execute the action for the current time period t. At last, according to the reward in the time period t, counters and cumulative rewards for selected actions are updated. In summary, the UCB algorithm ranks actions by calculating the upper bound value where actions

with higher average rewards and fewer selections have a larger upper bound value and are more likely to be selected, and it converges to the optimal action over time.

In this paper, we consider the CMAB framework where a super action a_u is a vector comprised of underlying actions $a_{u,f}$. In time period t, a super action a_u is selected and the rewards of underlying actions $a_{u,f}$ in the super action a_u could be observed. Therefore, the combinatorial upper confidence bound (CUCB) algorithm is proposed to utilize the expected rewards of underlying actions instead of the rewards of the super actions. As a result, by employing the CUCB algorithm, we could reduce the action space from the number of super actions 2^F to the number of underlying actions $2F$. However, the reduction of action space is at the cost that the optimal super action cannot be obtained directly and we have to compute the optimal super action or sub-optimal super action. The CUCB algorithm relies on a computation oracle, which computes the optimal or sub-optimal super action based on the expected rewards of the underlying actions and the problem instance.

Different from the UCB algorithm, the CUCB algorithm maintains an empirical mean reward $\hat{\mu}_f$ for each action $a_{u,f}$. The CUCB algorithm is described as $\mu_f = \hat{\mu}_f + \sqrt{\frac{\ln t}{n_f}}$. The difference between the CUCB and UCB is that the CUCB algorithm focuses on the rewards of the underlying actions $a_{u,f}$ instead of the super action a_u. Given the rewards of underlying actions, we could use the computation oracle to get the super action, and update n_f and $\hat{\mu}_f$ accordingly.

Hence, we further propose a learning-based CUCB algorithm as follows. Instead of $Q_u(a_u)$ in the traditional Q-learning algorithm, this algorithm employs $Q_{u,f}(a_{u,f})$, which is the average reward of action $a_{u,f}$ selected by SBS u. Let $N_{u,f}(a_{u,f})$ represent the number of times that SBSu has selected the action $a_{u,f} \in \{0,1\}$. In time period t, if action $a_{u,f}$ is selected by SBS u, the Q-value $Q_{u,f}(a_{u,f})$ is updated as:

$$Q_{u,f}(a_{u,f}) \leftarrow Q_{u,f}(a_{u,f}) + \frac{1}{N_{u,f}(a_{u,f})+1}\left(R^t_{u,f} - Q_{u,f}(a_{u,f})\right), \quad (14)$$

where $R^t_{u,f}$ can be calculated by equation (7).

The CUCB algorithm updates the Q-values based on the reward $R^t_{u,f}$ in time period t and the number of times action $a_{u,f}$ has been selected. Improving the CUCB algorithm, we add a weight parameter to the upper confidence interval to speed up the convergence and promote exploitation-exploration. The algorithm is described as $\mu_f = \hat{\mu}_f + c\sqrt{\frac{\ln t}{n_f}}$, which is called the modified combinatorial upper confidence bound (MCUCB) algorithm. Due to the Zipf-like distribution of the content popularity and the specific structure of the problem, we can define the weight parameter and update the Q-value as

$$\bar{Q}_{u,f} = Q_{u,f} + l \cdot \frac{14 \cdot N_u \cdot s_f}{F^\gamma}\sqrt{\frac{3\log t}{2N_{u,f}(1)}}, \quad (15)$$

where $Q_{u,f} = Q_{u,f}(1) - Q_{u,f}(0)$, $l = \max_{i \in \mathcal{F}} Q_{u,i}/s_i$. F is the number of files, and parameter γ is the shape factor of Zipf distribution. The weight parameter $l \cdot \frac{N_u \cdot s_f}{F^\gamma}$ balances exploitation and exploration as follows. When the Zipf distribution is skewed, characterized by a large γ value and few popular content files, the factor $\frac{1}{F^\gamma}$ promotes exploitation. Though we know none about the content popularity distribution in advance, we can empirically approximate the parameter γ as in [27]. The exploration is promoted when the parameter N_u is large, since N_u independent realizations of the reward distribution can be observed.

After computing the adjusted Q-value $\bar{Q}_{u,f}$ for each content file f, the single period problem (SPO) problem can be written as follows.

$$\mathbf{P5}: \max \sum_{f=1}^{F} a_{u,f} \cdot \bar{Q}_{u,f}, \quad (16a)$$

$$\text{s.t.} \sum_{f=1}^{F} a_{u,f} \cdot s_f \le S_u, \quad (16b)$$

$$a_{u,f} \in \{0,1\}. \quad (16c)$$

Let $a^O_u = (a^O_{u,1}, a^O_{u,2}, \ldots, a^O_{u,F})$ denote the optimal solution of the SPO problem. The SPO problem **P3** for finding the optimal super action a^O_u is the 0-1 knapsack problem, where the value is $\bar{Q}_{u,f}$ and the weight is s_f for each content file f. Knapsack problems known as NP-hard [28], can be solved by algorithms such as the Dynamic Programming Algorithm, Greedy Algorithm, Branch and Bound Algorithm, etc. In this paper, we study content caching in vehicular networks, so we need to obtain the super action promptly to adapt to the high dynamics. However, through these algorithms we cannot calculate the super action easily and these algorithms may involve randomness with a low probability of failure. Therefore, an algorithm converges to the sub-optimal solution a^O_u with a short computation time is preferred.

We employ an approximation oracle to solve **P3**, which is a (α, β)-solver. The (α, β)-solver is defined as: the algorithm can output a super action whose expected reward is at least α proportion of the optimal expected reward, with a probability of β when $\alpha, \beta \le 1$. To solve the 0-1 knapsack problem, a greedy algorithm is employed as the approximation oracle. In this algorithm, the reward per unit of weight is used as the performance indicator, and content files with a higher reward per unit of weight would be cached. At first, the performance indicator can be described as $Y_{u,f} = \bar{Q}_{u,f}/s_f, f \in \mathcal{F}$. After that, $Y_{u,f}$ is arranged in a descending order. The super action is initialized as $a^*_u = (0, 0, \ldots, 0)$, and a^*_{u,f_i} with the current maximum i in each iteration is set to 1 until the last a^*_{u,f_i}, which means we cannot cache any more content files because of the SBS cache size constraint S_u. At last, assuming $a^*_{u,f_1} = 1$, for $i = 1, 2, \ldots, F$, the elements a^*_{u,f_k} of the super action a^*_u is obtained as follows.

$$a^*_{u,f_i} = \begin{cases} 1, & \sum_{j=1}^{k-1} a^*_{u,f_j} s_{f_j} + s_{f_i} \le S_u \\ 0, & \sum_{j=1}^{k-1} a^*_{u,f_j} s_{f_j} + s_{f_i} > S_u. \end{cases} \quad (17)$$

However, after introducing the cooperative service for SBS caching, the SBS has to cache the content files for the requests

Algorithm 1 The Learning-Based Algorithm of Cooperative Service for SBS Caching

Step 1. Initialize:
$N_{u,f}(a_{u,f}) = 0$, $Q_{u,f}(a_{u,f}) = 0$, $\forall f \in \mathcal{F}$.
Cache all the content files once, observe the rewards $R^t_{u,f}$, and then update $N_{u,f}(a_{u,f})$ and $Q_{u,f}(a_{u,f})$, $\forall f \in \mathcal{F}$.
Set $t \leftarrow F+1$.
Step 2. observe (user request phase in period t):
for $f \in \mathcal{F}$ **do**
  Observe the reward $R^t_{u,f}$.
  Update $Q_{u,f}(a_{u,f}) \leftarrow Q_{u,f}(a_{u,f})+1/(N_{u,f}(a_{u,f})+1)\cdot(R^t_{u,f} - Q_{u,f}(a_{u,f}))$ and $N_{u,f}(a_{u,f}) \leftarrow N_{u,f}(a_{u,f}) + 1$, $\forall a_{u,f}$ selected in time period t.
  Compute $Q_{u,f} = Q_{u,f}(1) - Q_{u,f}(0)$.
end for
Compute $l = \max_{i\in\mathcal{F}} Q_{u,i}/s_i$ and $\bar{Q}_{u,f} = Q_{u,f} + l \cdot \frac{14\cdot N_u \cdot s_f}{F^\gamma}\sqrt{\frac{\log t}{N_{u,f}(1)}}$, $\forall f \in \mathcal{F}$.
Step 3. optimize (cache replacement in time periot t):
Observe the relayed requests $F_r^{t+1} = \{f_1, f_2, \ldots, f_r\}$ in the next time period $t+1$.
Compute $Y_{u,f} = \bar{Q}_{u,f}/s_f$ for the remaining content files, and arrange $Y_{u,f}$ in a descending order: Y_{u,f_i}, $f_i \in \mathcal{F}$, $i = r+1, r+2, \ldots, F$.
Assign a_{u,f_i} to 1 for content files in F_r^{t+1} and compute a_{u,f_i} for the remaining content files according to equation (17), then get super action a_u.
Cache the content files according to a_u.
Set $t \leftarrow t+1$.
Go to Step 2.

relayed from other SBSs the next time period $t+1$ to guarantee the relayed requests could be served in this SBS. Therefore, the solution of finding the optimal super action has to be modified, i.e., the SBS should cache the content files to serve the relayed requests in the next time period $t+1$ first. Specifically, assuming the relayed requests in the next time period $t+1$ is $F_r^{t+1} = \{f_1, f_2, \ldots, f_r\}$, arrange the remaining content files as $Y_{u,f_{r+1}} \geq Y_{u,f_{r+2}} \geq \cdots \geq Y_{u,f_F}, f_i \in \mathcal{F}$. Initializing the super action as $a_u = (0, 0, \ldots, 0)$, the optimal super action of cooperative service for SBS caching can be given as $a^*_{u,f_i} = 1$, $i = 1, 2, \ldots, r$, and the remaining elements are updated according to equation (17).

We define δ as the ratio between the optimal solution and the proposed approximation oracle, which can be estimated as $\delta = \sum_{f=1}^{F} a^O_{u,f}\cdot\bar{Q}_{u,f} / \sum_{i=1}^{F} a^*_{u,f_i}\cdot\bar{Q}_{u,f_i} \leq 2$ [28]. Therefore, the expected reward of the proposed approximation oracle is at least 1/2 proportion of the optimal expected reward with probability 1. Then we can derive the proposed algorithm as an (α, β)-solver with $\alpha = 1/2$ and $\beta = 1$. The learning-based algorithm for SBS caching is detailed in Algorithm 1.

As described in Algorithm 1, the computational complexity for computing the Q-values for content files is $O(F)$, and that for finding the optimal super action by the approximation oracle is also $O(F)$. This means that the computational complexity for Algorithm 1 is $O(F)$.

B. MBS Caching Strategy

The MBS caching problem is modeled as a CMAB problem as described in equation (12). The objective of the MBS caching problem is to cache the content files with more expected rewards in the MBS in order to reduce the requests relayed to the cloud, which is similar to the SBS caching problem. Thus, the proposed learning-based CUCB algorithm can be employed to solve the MBS caching problem from a greater timescale. In addition, rather than the greedy-based $(1/2, 1)$-solver, the dynamic programming algorithm is applied to solve the knapsack problem in the MBS caching.

In the MBS caching problem, the super action is the caching strategy of the MBS, $A_m = \{(a_{m,1}, a_{m,2}, \ldots, a_{m,F}) | a_{m,f} \in \{0,1\}, \sum_{f=1}^{F} a_{m,f} \leq S_m, f \in \mathcal{F}\}$. In the CMAB framework, we consider the action $a_{m,f}$, $f \in \mathcal{F}$ to reduce the action space. In the stateless setting, $Q_{m,f}(a_{m,f})$, $f \in \mathcal{F}$ denotes the estimated reward of executing action $a_{m,f}$ in the next time interval. The learning rate is defined as the reciprocal of the number of times action $a_{m,f}$ selected by the MBS in the past, $\frac{1}{N_{m,f}(a_{m,f})+1}$, thus in time interval T the Q-value $Q_{m,f}(a_{m,f})$ is updated by

$$Q_{m,f}(a_{m,f}) \leftarrow Q_{m,f}(a_{m,f}) + \frac{1}{N_{m,f}(a_{m,f})+1}\left(\sum_{u=1}^{U} R^T_{m,u,f} - Q_{m,f}(a_{m,f})\right), \quad (18)$$

where $R^T_{m,u,f}$ is given by equation (11).

To promote the exploitation-exploration, the MCUCB algorithm is employed to adjust the Q-values based on the content popularity and the particular structure of the problem, and the adjusted Q-value will be given by

$$\bar{Q}_{m,f} = Q_{m,f} + l' \cdot \frac{2\cdot U \cdot N_u \cdot s_f}{F^\gamma}\sqrt{\frac{3\log T}{2N_{m,f}(1)}}, \quad (19)$$

where $Q_{m,f} = Q_{m,f}(1) - Q_{m,f}(0)$, $l' = \max_{i\in\mathcal{F}} Q_{m,i}/s_i$. For the MBS caching problem, we modify the weight parameter by multiplying it by the number of SBSs U to further promote the exploitation-exploration.

Therefore, the SPO problem for the MBS caching problem could be described as follows,

$$\mathbf{P6}: \max \sum_{f=1}^{F} a_{m,f}\cdot\bar{Q}_{m,f}, \quad (20a)$$

$$\text{s.t.} \sum_{f=1}^{F} a_{m,f}\cdot s_f \leq S_m, \quad (20b)$$

$$a_{m,f} \in \{0,1\}. \quad (20c)$$

The optimization problem **P6** is also a 0-1 knapsack problem. Different from the SBS caching problem, in time interval T, the MBS has a relatively long time to get the solution of **P6**, and the dynamic programming algorithm is chosen to find a better solution. The detailed MBS caching strategy is shown in Algorithm 2, where $V(f, S)$ denotes the maximum value obtained by caching the first i content files in the MBS with the capacity of S. Initializing the boundary conditions by

Algorithm 2 The Learning-Based Algorithm for MBS Caching

Step 1. Initialize:
$N_{m,f}(a_{m,f}) = 0$, $Q_{m,f}(a_{m,f}) = 0$, $\forall f \in \mathcal{F}$.
Cache all the content files once, observe the rewards $R^T_{m,u,f}$, and then update $N_{m,f}(a_{m,f})$ and $Q_{m,f}(a_{m,f})$, $\forall f \in \mathcal{F}$.
Initial the MBS cache queue according to the Q-value $Q_{m,f}(a_{m,f})$.
Set $T \leftarrow F + 1$.
Step 2. observe (SBS request phase in interval T):
for $f \in \mathcal{F}$ **do**
  Observe the reward $R^T_{m,u,f}$, $\forall u \in \mathcal{U}$.
  Update $Q_{m,f}(a_{m,f}) \leftarrow Q_{m,f}(a_{m,f}) + 1/(N_{m,f}(a_{m,f}) + 1) \cdot (\sum_{u=1}^{U} R^T_{m,u,f} - Q_{m,f}(a_{m,f}))$ and $N_{m,f}(a_{m,f}) \leftarrow N_{m,f}(a_{m,f}) + 1$, $\forall a_{m,f}$ selected in time interval T.
  Compute $Q_{m,f} = Q_{m,f}(1) - Q_{m,f}(0)$.
end for
Compute $l' = \max_{i \in \mathcal{F}} Q_{m,i}/s_i$ and $\bar{Q}_{m,f} = Q_{m,f} + l' \cdot \frac{2 \cdot U \cdot N_u \cdot s_f}{F^\gamma} \sqrt{\frac{\log T}{N_{m,f}(1)}}$, $\forall f \in \mathcal{F}$.
Step 3. optimize (cache replacement in time interval T):
Create a table $V(f, S)$, $f = 0, 1, 2, \ldots, F$, $S = 0, 1, 2, \ldots, S_m$, and initial the boundary conditions, $V(0, S) = V(f, 0) = 0$.
Set the state transition equation as, $V(f, S) = max(V(f-1, S), V(f-1, S-s_f) + \bar{Q}_{m,f})$, thus fill the table $V(f, S)$.
According to the result $V(F, S_m)$, compute the optimal solution a^*_m and Cache the content files according to a^*_m.
Set $T \leftarrow T + 1$.
Go to Step 2.

$V(0, S) = V(f, 0) = 0$, the state transition equation is found by $V(f, S) = \max(V(f-1, S), V(f-1, S-s_f) + \bar{Q}_{m,f})$. In other words, the first case is the MBS cannot cache the content file f, $V(f, S) = V(f-1, S)$, and if the MBS is able to cache the content file f, $V(f, S) = V(f-1, S-s_f) + \bar{Q}_{m,f})$. After filling the table, the result $V(F, S_m)$ will be the solution to find the optimal super action a^*_m. For the dynamic programming algorithm, we create a table of $V(f, S)$, from which the complexity is derived as $O(S_m \cdot F)$.

VI. Simulation Results

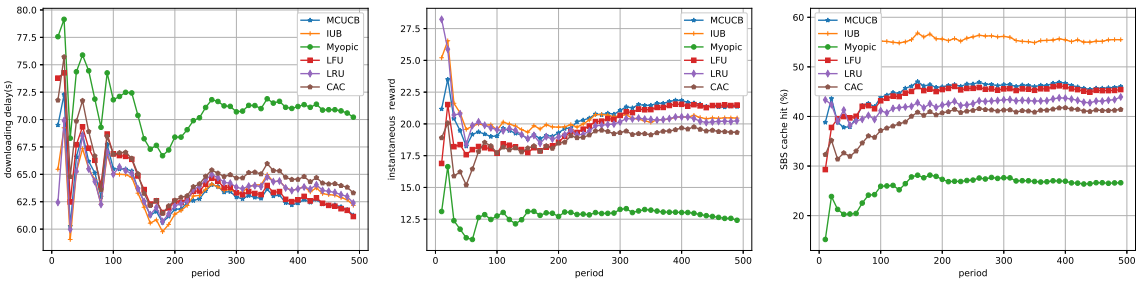
In this section, we demonstrate the performance of the proposed MCUCB algorithm for SBS caching and MBS caching. The performance metrics include: 1) the downloading delay, 2) the instantaneous reward, 3) the percentage of the local cache hit rate, 4) the percentage of the local SBS cache hit rate, 5) the MBS cache hit rate, and 6) the total cache hit rate (the sum of the SBS cache hit rate and the MBS cache hit rate). It is noted that the downloading delay is calculated by the models in Section III-D, including the total downloading delay of all requests during a time period. The instantaneous reward is the reduction of downloading delay compared to the case without caching for all the content files in the SBS. The percentage of local cache hit rate is the part of the requests that could be served by the local SBS, which is the sum of the percentage of the local SBS cache hit rate, the MBS cache hit rate, and the cloud cache hit rate. Moreover, due to the dynamics of the environment, the average of the performance metrics from the start to the current period is chosen to better display the performance of the algorithm.

TABLE II
Simulation Parameters

Parameters	Values
Transmission range of SBS	$100m$
Transmission range of MBS	$200m$
Transmission power of SBS (P_{SBS})	$1W$
Transmission power of MBS (P_{MBS})	$40W$
Gaussian white noise power (σ^2_N)	$1W$
Path loss exponent (α)	4
Bandwidth of SBS-vehicle links ($B_{u,v}$)	$10MHz$
Bandwidth of MBS-SBS links ($B_{m,u}$)	$10MHz$
Number of SBSs (U)	10
Number of content files (F)	50
Poisson distribution parameter (λ)	5
Size of content files (s_f)	$[1, 9]$
Cache size of SBSs (S_u)	25
Cache size of MBS (S_m)	125
Parameter of Zipf distribution (γ)	1
Maximum local downloading delay (D_{max})	$25s$
Proportion factor of payment (r)	0.5

To evaluate the performance of the algorithm proposed in this paper, the following benchmark algorithms are selected for comparison.

- **Informed Upper Bound (IUB) Algorithm**: The popularity profile Θ is known in advance, which provides an upper bound on the performance of any MAB algorithm. Since the size of the content file is not evenly distributed, instead of the popularity, popularity per unit size is sorted to improve the caching strategy. Moreover, if the remaining cache size is inefficient to cache the most popular content file, the less popular content file with smaller size will be cached. Therefore, the IUB algorithm actually provides the relatively better caching strategy for comparison.
- **Context-Aware Proactive Caching (CAC) Algorithm [29]**: The caching strategy is divided into two phases: exploration and exploitation. In the exploration phase, the under-explored content files will be cached first. In the exploitation phase, the content caching problem is modeled as an MAB problem, and the popular content files are cached.
- **Least Frequently Used (LFU) Algorithm [30]**: The least requested content file is replaced with the requested content file which is unavailable in the local cache in the current time period. The LFU algorithm measures the popularity of the content file to cache the content file receiving more requests.
- **Least Recently Used (LRU) Algorithm [31]**: The least recently requested content file is replaced with the requested content file which is unavailable in the local cache in the current time period. For simplicity, in each



(a) Downloading delay (b) Instantaneous reward (c) SBS cache hit rate

Fig. 3. The downloading delay, instantaneous reward and SBS cache hit rate for SBS caching with the stationary content popularity

time period, the most popular content files during the recent time interval are kept.

- **Myopic Algorithm [22]**: The requested content files in the last time period are kept and the rest of the content files are replaced randomly.

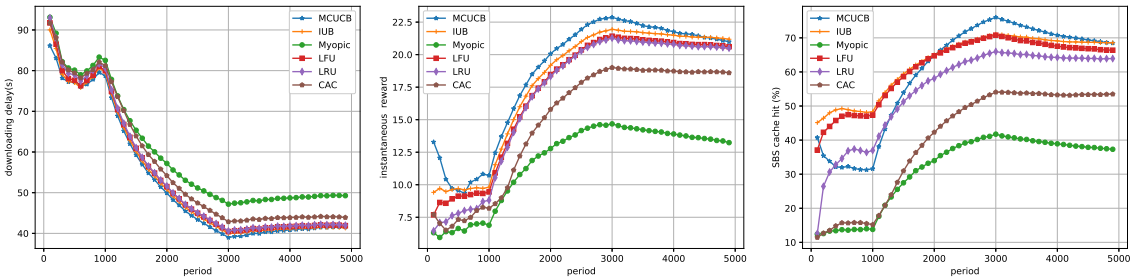
In this paper, the requests for content follow the independent Zipf distribution, $\theta_f = \frac{1}{f^\gamma \sum_{i=1}^{F} \frac{1}{i^\gamma}}$, where the Zipf parameter γ is set to 1. Unless otherwise stated, the number of content files F is 50, and the size of content files follows the power-law distribution, $P(s_f = x) = c \cdot x^{-\alpha}$, $1 \leq s_f \leq 10$, $s_f \in Z$, where c is the normalization constant and the power-law index α is set to 1. The number of requests SBS u received in time period t, i.e., N_u^t follows the Poisson distribution, and the distribution parameter $\lambda = 5$. Both the SBSs and the MBS are uniformly distributed in a square area of 500×500 m^2, and the transmission range of the SBS and the MBS will be 100 m and 200 m, respectively. The local cache size of the SBSs and the MBS are set as $S_u = 25$ and $S_m = 125$. The transmission power of the SBS and MBS are $P_{SBS} = 1$ W, $P_{MBS} = 40$ W, respectively. The path loss exponent $\alpha = 4$, Gaussian white noise $\sigma_N^2 = 1$ W, and the bandwidth of the SBS and MBS are $B_{u,v} = B_{m,u} = 10$ MHz, respectively. Furthermore, the transmission delay between the MBS and the cloud is set as twice the maximum value of that between the SBS and the MBS, i.e., $\tau_{c,m} = \max_{u \in \mathcal{U}} 2/[B_{m,u} \log_2 (1 + SNR_{m,u})]$. Specifically, considering the cooperative service for SBS caching, the maximum local downloading delay is reduced to a constant, $D_{max} = 25$ s. The default system parameters are shown in TABLE II. In addition, for the cooperative service for SBS caching, the relayed SBS selection is modeled as the random selection of the SBSs that the vehicle may encounter on its route.

A. SBS caching with the stationary content popularity

First, as shown in Fig. 3, with stationary content popularity $\gamma = 1$, We compare the performance of SBS caching algorithms by analyzing downloading delay, instantaneous reward and SBS cache hit rate. To simplify, the IUB algorithm will be employed to solve the MBS caching problem, and we will only consider the single SBS in this simulation. In Fig. 3a, Fig. 3b and Fig. 3c, we could observe that the downloading delay, the instantaneous reward, and the SBS cache hit rate will be relatively random in the beginning and finally converge, which is due to the dynamics of the environment. Additionally, due to the large number of content files, the performance of algorithms varies with the random requests even under the stationary popularity distribution. Due to the fixed content popularity, LFU and LRU algorithms achieve similar performance. The Myopic algorithm achieves the worst performance due to that the Myopic algorithm learns only from the recent time period and random cache content files. Moreover, the SBS cache hit rate for the IUB algorithm is significantly more than other algorithms, while for the other two performance metrics, the IUB algorithm performs similarly to and even worse than other algorithms. Since the IUB algorithm will cache the most popular content file, content files in the SBS and MBS will have many duplicates, which increases the downloading delay and decreases the instantaneous reward. The CAC algorithm balances the exploration and the exploitation, then obtains a relatively poor performance. In Fig. 3a, the downloading delay for the MCUCB algorithm is about $61s$, similar to other benchmark algorithms. In Fig. 3b, the instantaneous reward for the MCUCB algorithm is approximately 22, which is more than other benchmark algorithms. In Fig. 3c, since the content popularity is known in advance, the SBS cache hit rate for the IUB algorithm is about 56%, and those for the MCUCB, CAC, LFU and LRU algorithms are close to 45%. For the SBS caching problem, with the stationary content popularity, the MCUCB algorithm achieves the relatively better performance.

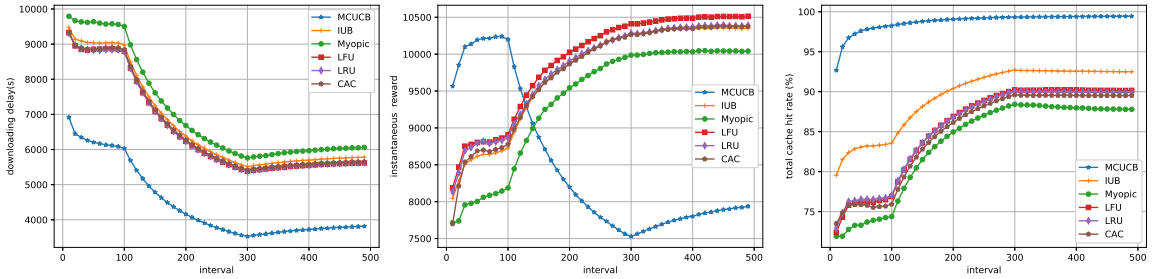
B. SBS caching with the non-stationary content popularity

In Fig. 4, with the non-stationary content popularity, the downloading delay, instantaneous reward, and the SBS cache hit rate are compared with the benchmark algorithms. Similarly, the IUB algorithm will be employed to solve the MBS caching problem, and only consider the single SBS. For the non-stationary content popularity, the parameter γ is initialized as 0.5, and changed to 1.5 and 1.0 in time period 1000 and 3000, respectively. Ignore the fluctuation at the beginning, we will focus on the following performance. In Fig. 4a, we observe that the MCUCB algorithm acquires the least downloading delay of about $40s$, similar to that of other benchmark algorithms. In Fig. 4b, the figure shows that the instantaneous



(a) Downloading delay (b) Instantaneous reward (c) SBS cache hit rate

Fig. 4. The downloading delay, instantaneous reward and SBS cache hit rate for SBS caching with non-stationary content popularity



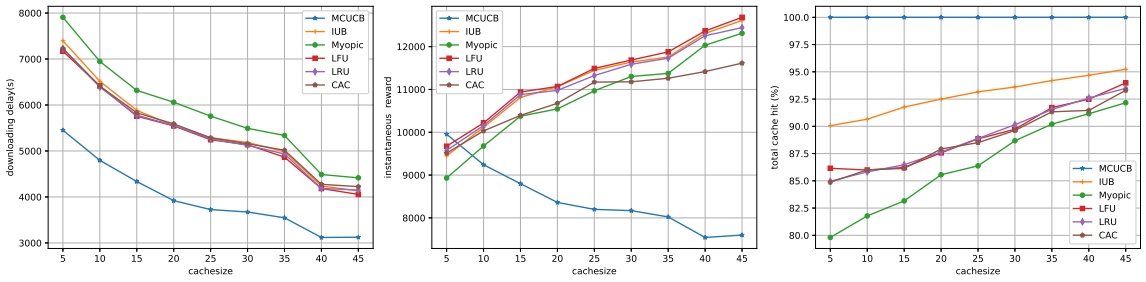
(a) Downloading delay (b) Instantaneous reward (c) Total cache hit rate

Fig. 5. The downloading delay, instantaneous reward and total cache hit rate for MBS caching with non-stationary content popularity

reward for the MCUCB algorithm is about 21, similar to that of the IUB algorithm. Fig. 4c indicates that the SBS cache hit for the MCUCB is close to 69%, slightly less than the IUB algorithm. We could observe that when the parameter γ increases, the performance of the algorithms improves. when the parameter γ is large, most content requests concentrate on a small number of content files, and caching the most popular content files could serve more requests, that is, decrease the downloading delay, and increase the instantaneous reward and SBS cache hit. Comparing Fig. 3 and Fig. 4, it is indicated that the MCUCB acquires a similar performance, in other words, it adapts to the variation of parameter γ and still offers a promising performance.

C. MBS caching with non-stationary content popularity

In Fig. 5, for the non-stationary content popularity, we study the performance of the proposed algorithm in the MBS caching problem, compared with the benchmark algorithms. In this simulation, the IUB algorithm will be employed to solve the SBS caching problem for the benchmark algorithms, and the MCUCB algorithm will be employed to solve both the MBS caching problem and the SBS caching problem for the proposed algorithm. Instead of the time period t, the long timescale interval T composed of 10 time periods will used in this simulation. For the non-stationary popularity, the parameter γ is initiated as 0.5, and changed to 1.5 and 1.0 in time interval 100 and 300, respectively. Therefore, the downloading delay is the sum of the downloading delay for the requests in this interval, the instantaneous reward is the reduction of the downloading delay in this interval, and the total cache hit rate is the sum of the SBS cache hit rate and the MBS cache hit rate. In Fig. 5a, the Myopic algorithm get the worst performance, and the proposed algorithm gets the best performance, where the downloading delay is about $4000s$ in the interval, with the improvement on the percentage is approximately 30% compared with the IUB algorithm. In Fig. 5b, the instantaneous rewards of IUB, LFU, LRU, CAC and Myopic algorithms increase with time while that of the MCUCB algorithm decreases. In Fig. 5c, the proposed algorithm gets the maximum total cache hit rate, more than that of the IUB algorithm, and the Myopic algorithm gets the minimum total cache hit rate. For the non-stationary content popularity, the LRU and LFU algorithm cache the content files which is requested most, but due to the size of the content files being unfixed, the higher cache hit rate does not mean a lower downloading delay and higher instantaneous reward. For the CAC algorithm, after the algorithm converges, some unpopular content files will also be cached in the exploration phase, worsening the performance. Fig. 5 demonstrates that when the content popularity changes, the proposed algorithm outperforms other algorithms for the MBS caching problem.



(a) Downloading delay (b) Instantaneous reward (c) Total cache hit rate

Fig. 6. The downloading delay, instantaneous reward and total cache hit rate for MBS caching with different SBS cache sizes

D. MBS caching with different SBS cache sizes

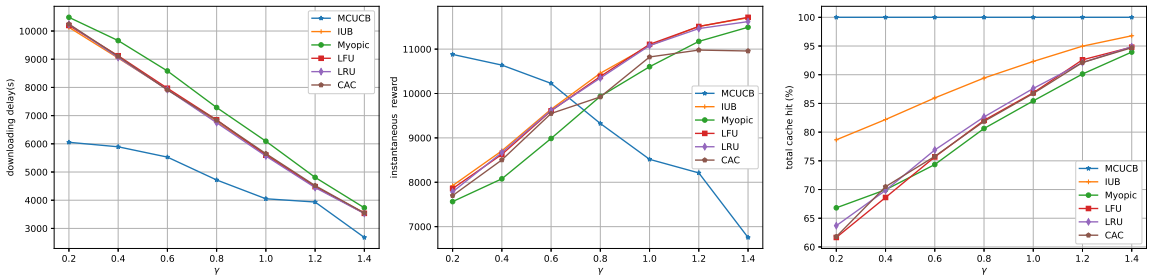
Fig. 6 shows the performance of the algorithms when the cache size of SBS varies from 5 to 45. The setting in this simulation is similar to the simulation in Fig. 5. The performance metrics in this simulation will be the average during 200 intervals. In Fig. 6a, for these algorithms, the downloading delay decreases with the increase in cache size of the SBS. When the cache size of SBS increases, more content files could be cached in the SBS, thus these content files could be acquired with a lower downloading delay. In Fig. 6b, the instantaneous reward for the benchmark algorithms increases with the increase of the cache size of the SBS, and that of the proposed algorithm decreases. In Fig. 6c, the total cache hit rate for the IUB, CAC, Myopic, LFU, and LRU algorithms increases with the increase of cache size of SBS, and the total cache hit rate for the MCUCB algorithm remains almost 100%. For the benchmark algorithms, when the cache size of SBS increases, the SBS cache hit rate increases, so more requests can be served in the SBS. Therefore, the total cache hit rate and instantaneous reward increases with the increase of cache size. However, when the cache size of SBS increases, the total cache hit rate remains the same, where the SBS cache hit rate increases and the MBS cache hit rate decreases. As illustrated in (5), the instantaneous reward for serving the requests in the MBS is more than that in the SBS. When the cache size of SBS increases, more requests will be served in the SBS, thus the instantaneous reward decreases. As shown in Fig. 6, regardless of the cache size of SBS, the proposed MCUCB algorithm maintain a good performance, better than other benchmark algorithms.

E. MBS caching with different content popularity

Fig. 7 shows the performance of the algorithms when the Zipf parameter γ varies from 0.2 to 1.4. The parameter setting is similar to the simulation in Fig. 6. In Fig. 7a, when γ increases, the downloading delay for these algorithms decreases. In Fig. 7b, for the benchmark algorithms, the instantaneous reward increases and that of the MCUCB algorithm decreases. In Fig. 7c, total cache hit rate for the MCUCB algorithm slightly increases and that of other benchmark algorithms increases. As γ increases, the popular content files are more likely to be requested by users and the popularity difference between popular files and unpopular files will be larger. Therefore, more requests will focus on a small portion of content files with highest popularity. Total cache hit rate increases with the increase of γ. As more requests are served in SBSs, the downloading delay decreases with the increase of γ. For the benchmark algorithms, the instantaneous reward increases due to that more requests are served. However, for the MCUCB algorithm, more requests are served in SBSs and less requests are served in the MBS, thus the instantaneous reward for the MCUCB algorithm decreases with the increase of γ. Fig. 7 verifies that the MCUCB algorithm adapts to the changes in the content popularity compared with benchmark algorithms.

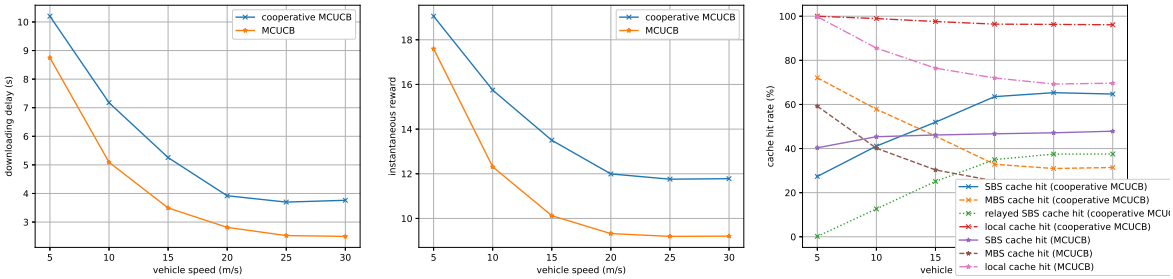
F. Cooperative MBS caching for different vehicle speeds

Finally, to verify the performance of the proposed cooperative MCUCB algorithm in a high-speed environment, we compare the performance between the MCUCB algorithm and the cooperative MCUCB algorithm for different vehicle speeds in Fig. 8. In this simulation, the maximum local downloading delay D_{max} is computed by $D_{max} = 2 \cdot l_u / v$. For the MCUCB algorithm, if the downloading delay for the content file is higher than D_{max}, the content file will fail to be returned to the vehicle, while for the cooperative MCUCB algorithm, the request will be relayed to another SBS on the route of the vehicle. The downloading delay and instantaneous reward in this simulation are those of a single request, to better show the performance. Fig. 8a and Fig. 8b shows that the downloading delay and instantaneous reward for the cooperative MCUCB algorithm are higher than that of MCUCB algorithm, which is because for the cooperative MCUCB algorithm, the last SBS in this area cannot relay the request to another SBS, thus increasing the downloading delay and instantaneous reward. In Fig. 8c, for the MCUCB algorithm, when the vehicle speed increases and the maximum local downloading delay D_{max} decreases, the SBS cache hit rate almost keeps the same. Due to the limit of the maximum local downloading delay, fewer requests can be served by the MBS, so the MBS cache hit rate and local cache hit rate decrease. For the cooperative MCUCB algorithm, with the increase of vehicle speed, more requests are relayed to other SBSs, thus the MBS cache hit rate decreases, SBS cache hit rate and the relayed SBS cache



(a) Downloading delay (b) Instantaneous reward (c) Total cache hit rate

Fig. 7. The downloading delay, instantaneous reward and total cache hit rate for MBS caching with different content popularity



(a) Downloading delay (b) Instantaneous reward (c) Cache hit rate

Fig. 8. The downloading delay, instantaneous reward and cache hit rate of the MCUCB and cooperative MCUCB algorithms for MBS caching

hit rate increases. Fig. 8c indicates that the higher the vehicle speed is, the higher the importance of cooperative service is. In conclusion, the cooperative MCUCB algorithm can adapt to the high-speed environment well.

VII. Conclusion and Future Work

Caching the popular content files in the distributed storage-enabled network entities can alleviate the pressure on the backhaul links with limited capacity in the vehicular networks. This paper considers a cache-enabled multi-layer architecture, comprising several SBSs and an MBS. A two-timescale approach is proposed for cache refreshing, where the SBSs refresh the content files in a short timescale because they are closer to the vehicles, while the MBS refreshes the content in a long timescale as it receives requests from the SBSs with a lower frequency. The content caching problem is modeled from an MAB perspective, and the learning-based algorithm MCUCB is proposed for the SBSs and the MBS. Additionally, to guarantee that the vehicles can get the requested content files in the high-speed environment, cooperative service was introduced to further improve the QoS of vehicles. Simulation results show the impressive performance of the proposed algorithms and prove that the proposed approach successfully adapts to the dynamics of content file popularity.

There are some challenges in the cooperative caching problems that are worth further exploration. For example, the relaying SBSs should be carefully selected to improve content delivery efficiency and reduce the redundant transmission. In addition, how to handle the requests from the vehicles with a stringent time requirement is also a challenging problem.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (NSFC) (62002138) and the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT2022B26).

References

[1] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1111–1125, 2018.

[2] C. Fan, X. Zhou, T. Zhang, W. Yi, and Y. Liu, "Cache-enabled uav emergency communication networks: Performance analysis with stochastic geometry," *IEEE Transactions on Vehicular Technology*, 2023.

[3] X. Zhu, C. Jiang, L. Kuang, and Z. Zhao, "Cooperative multilayer edge caching in integrated satellite-terrestrial networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 2924–2937, 2021.

[4] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2017.

[5] A. Sadeghi, F. Sheikholeslami, A. G. Marques, and G. B. Giannakis, "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2267–2281, 2019.

[6] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled hetnet," in *2017 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2017, pp. 449–453.

[7] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, 2017.

[8] X. Zhou, Z. Liu, M. Guo, J. Zhao, and J. Wang, "SACC: A size adaptive content caching algorithm in fog/edge computing using deep reinforcement learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 1810–1820, 2021.

[9] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2018, pp. 1–6.

[10] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2017.

[11] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1024–1033, 2019.

[12] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1635–1648, 2016.

[13] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[14] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Proactive retention-aware caching with multi-path routing for wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1286–1299, 2018.

[15] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile d2d networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1610–1622, 2019.

[16] X. Xu, M. Tao, and C. Shen, "Collaborative multi-agent multi-armed bandit learning for small-cell caching," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2570–2585, 2020.

[17] X. Xu and M. Tao, "Decentralized multi-agent multi-armed bandit learning with calibration for multi-cell caching," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2457–2472, 2020.

[18] X. Peng, J.-C. Shen, J. Zhang, and K. B. Letaief, "Backhaul-aware caching placement for wireless networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

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

[20] Y. Chen, M. Ding, J. Li, Z. Lin, G. Mao, and L. Hanzo, "Probabilistic small-cell caching: Performance analysis and optimization," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4341–4354, 2016.

[21] A. Tang, S. Roy, and X. Wang, "Coded caching for wireless backhaul networks with unequal link rates," *IEEE Transactions on Communications*, vol. 66, no. 1, pp. 1–13, 2017.

[22] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 1897–1903.

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

[24] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *International Conference on Machine Learning*. PMLR, 2013, pp. 151–159.

[25] S. Kapetanakis, D. Kudenko, and M. J. Strens, "Reinforcement learning approaches to coordination in cooperative multi-agent systems," in *Symposium on Adaptive Agents and Multi-Agent Systems*. Springer, 2001, pp. 18–32.

[26] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM'99*, vol. 1. IEEE, 1999, pp. 126–134.

[28] A. Korbut and I. K. Sigal, "Exact and greedy solutions of the knapsack problem: The ratio of values of objective functions," *Journal of Computer and Systems Sciences International*, vol. 49, pp. 757–764, 2010.

[29] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2016.

[30] G. Zhang, B. Tang, X. Wang, and Y. Wu, "An optimal cache placement strategy based on content popularity in content centric network," *Journal of Information &Computational Science*, vol. 11, no. 8, pp. 2759–2769, 2014.

[31] M. Bilal and S.-G. Kang, "Time aware least recent used (TLRU) cache management policy in ICN," in *16th International Conference on Advanced Communication Technology*. IEEE, 2014, pp. 528–532.

Declaration of interests

⏏ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The author is an Editorial Board Member/Editor-in-Chief/Associate Editor/Guest Editor for *[Journal name]* and was not involved in the editorial review or the decision to publish this article.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: