Proactive Edge Caching in Content-Centric Networks With Massive Dynamic Content Requests

XIAOGENG XU[1], (Student Member, IEEE), CHUNYAN FENG[1], (Senior Member, IEEE), SIYANG SHAN[1], (Student Member, IEEE), TIANKUI ZHANG[1], (Senior Member, IEEE), AND JONATHAN LOO[2]

[1]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]School of Computing and Engineering, University of West London, London W5 5RF, U.K.

Corresponding author: Tiankui Zhang (zhangtiankui@bupt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61971060 and Grant 61502046.

ABSTRACT Edge computing is a promising infrastructure evolution to reduce traffic loads and support low-latency communications. Furthermore, content-centric networks provide a natural solution to cache contents at edge nodes. However, it is a challenge for edge nodes to handle massive and highly dynamic content requests by users, and if without an efficient content caching strategy, the edge nodes will encounter high traffic load and latency due to increasing retrieval from content providers. This paper formulates a proactive edge caching problem to minimize the content retrieval cost at edge nodes. We exploit the inherent content caching and request aggregation mechanism in the content-centric networks to jointly minimize traffic load and content retrieval delay cost generated by the massive and dynamic content requests. We develop a Q-learning algorithm, which is an online optimal caching strategy, as it is adaptable to dynamic content popularity and content request intensity, and derive the long-term minimization of the content retrieval cost. Simulation results illustrate that the proposed algorithm can achieve a lower content retrieval cost compared with several baseline caching schemes.

INDEX TERMS Content-centric networks, dynamic content requests, edge computing, proactive caching, Q-learning.

I. INTRODUCTION

The demand for broadband multimedia applications has been rising exponentially, leading to a dramatic increase in network capital and operating expenditures. Edge computing is an emerging paradigm which stimulates the computing, caching and communication abilities of the network edge nodes (ENs), to satisfy the high quality of service requirements including low latency and high throughput [1]. However, the high expenditure of deployment and the location-centric network architecture restrict the efficiency and scalability of edge computing [2]. To handle these problem, content-centric networks (CCN) provides an intelligent, flexible and convenient edge caching solution [3], [4].

CCN is one of the well known information-centric networking (ICN) architectures which meets the requirements of the content dissemination services in future networks [5]. It utilizes the named data rather than the location as the network narrow waist, and realizes the in-network caching and named data forwarding are realized by means of content name resolution [6]. Taking the advantage of in-network caching in CCN, contents can be cached to any forwarding node adjacent to users, which reveals the opportunity to satisfy the content requests from users efficiently. Moreover, the name-based forwarding enables the CCN forwarding engine to deploy the inherent request aggregation mechanism to reduce redundancy transmissions [7].

Unfortunately, compared with the increasing volume of contents, the cache spaces at the ENs are limited, because of the line-speed forwarding requirements to the forwarding nodes [8]. The conflict between the massive content requests and the limited cache space makes it impossible to store all demanded contents at the ENs. Therefore, the ENs should adopt an intelligent caching strategy to be able to satisfy the user requirements with the limited resources.

In order to guarantee the best caching performance, the optimum contents have to be carefully chosen and proactively cached to ENs according to a precise optimization objective [9]. The proactive caching strategies in CCN have been widely studied. The early studies aim to enhance the caching performance in the static environments [10]–[12], while do not take into account the time-varying nature of the content requests, new content generations as well as user mobility. Migrating the aforementioned caching strategies to dynamic environments would introduce iterative calculations. Even worse, the caching performance would be suffered from the postponed caching decisions. Therefore, the recent research has paid more attention to content caching in dynamic scenarios in CCN [13]–[19].

Given that content popularity is an effective measure for making caching decisions, extensive works have been devoted to dynamic content popularity-based content caching in CCN. In addition to content popularity prediction [17], [18], reinforcement learning [19] has been emerging to capture the dynamic popularity characteristics. Their formulated optimization problems are able to approximate to the actual system performances with low content request intensity. However, in the scenarios of massive content requests, the performances become distinctive due to the inherent request aggregation mechanism in CCN [20], [21].

Under the circumstances, higher content request intensity results in less forwarded content requests probability, which alleviates the traffic load burden to the upstream networks and shortens the content retrieval delay. As a result, the proactive caching in CCN with massive and dynamic content requests, jointly considering the time-varying content popularity and content request intensity, is to be explored.

To handle the massive and highly dynamic content requests at ENs, we formulate a proactive edge caching problem to minimize the content retrieval cost based on CCN architecture. We exploit both the inherent content caching and request aggregation mechanism in CCN to jointly minimize traffic load and content retrieval delay cost. We further develop a Q-learning algorithm, which is an online optimal caching strategy, as it is adaptable to dynamic content popularity and content request intensity, and derive the long-term minimization of the content retrieval cost. The innovations and contributions of this work are summarized as follows.

- We formulated a proactive edge caching optimization problem to handle the massive and highly dynamic content requests to ENs, with an objective to minimize the content retrieval cost. The objective takes into account the traffic load and content retrieval delay, and makes a trade-off between the two metrics. The proposed problem is an integer non-linear programming problem, which is NP-complete.
- We exploited the request aggregation mechanism to reduce the redundancy transmissions incurred by massive content requests. We explored the traffic load and content retrieval delay functions under the joint effects of content caching and request aggregation mechanism. Simulation results verified that the employment of request aggregation mechanism can obtain extra remarkable performance gain additional to content caching.
- We developed a Q-learning algorithm, which is an online optimal caching strategy, as it is adaptable to both dynamic content popularity and content request intensity. The Q-learning algorithm provided the optimal proactive caching strategy and derived the long-term minimization of the content retrieval cost.
- We also implemented Greedy algorithms, most popular content (MPC) algorithms and random caching (RND) algorithm for the purposed of experimentation and comparisons. Simulation results showed that the proposed Q-learning based proactive caching algorithm outperformed the benchmark algorithms with dynamic content requests. Furthermore, simulation results also demonstrated that deploying CCN architecture at ENs can significantly relieve the burden encountered from massive content requests.

The rest of the paper is organized as follows. Section II introduces the system model. Section III states the optimization problem. Section IV and V solve the proactive caching optimization problem given that the dynamic content requests is perfectly known and unknown respectively. Section VI analyzes the caching performance numerical results. And Section VII concludes the paper.

II. RELATED WORK

How to design effective edge caching strategies for dynamic networks is still an open issue. Prediction [22], [23] and reinforcement learning [24], [25] are two main approaches to for ENs to track the dynamic network characteristics. To adapt the time-spatial content popularity fluctuations, content popularity prediction plays an important role in efficient proactive caching [22]. A linear model for content popularity prediction was constructed considering both the content and location features, and a location-customized caching scheme was derived to maximize the total content hit rate [23]. In recent years, reinforcement learning has attracted much academic and industry interest, and been adopted to solve the optimization in dynamic scenarios. It is utilized either as a prediction algorithms [24] or to directly output the best policy according to the proposed optimization objectives [25]. Considering the new content generation, the popularity prediction models for the published and unpublished videos were established respectively [24]. A closed form solution of proactive caching algorithm load was presented to minimize the backhaul.

According to the caching proactivity, the caching strategies in CCN are able to be categorized to the reactive caching (or passive caching) and the proactive caching (or prefetching), respectively [8]. With reactive caching strategies, the caching nodes make the caching decisions to its passing-by contents. Reactive caching is adept at distributed deployments, but provides insufficient caching performance unfortunately [26]. On the contrary, proactive caching is able to provide a caching performance guarantee by means of fetching the optimum contents to the cache nodes before users actually request them [9]. Proactive caching has been widely researched in static environment in CCN [11], [12]. An proactive caching placement problem was studied for arbitrary topology combined with multi-hop forwarding to minimize the user delay considering the load balancing requirements of the forwarding nodes simultaneously [11]. An distributed energy-efficient in-network caching scheme for CCN was proposed as an integer linear problem, with limited caching capacity constrains [12].

Comparatively, only a few work have been proposed to the proactive caching strategies in dynamic environment in CCN. Among them, content popularity fluctuations as well as network topology variances are the most typical scenarios. Content popularity fluctuation is one of the most attractive dynamic factors [17]–[19]. By discovering the relevance among video chunks in ICN, the most popular content chunks in the future were predicted and cached and those with least future popularity were evicted [17]. Another popularity-based caching strategy was proposed considering the temporal evolutions and spatial dependencies of network traffic. A deep-learning-based global content popularity prediction scheme was utilized to improve the prediction accuracy [18]. A QoE-based caching placement issue was derived in mobile edge for dynamic video-streaming, in terms of reducing the storage cost of the base stations while maintaining a high value of QoE [19]. Some work has been done studying the network topology variance in mobile CCN, including cellular network, internet of things (IoT), and Vehicular Ad-hoc network (VANET), etc [14]–[16]. Entropy was proposed to both measure mobility prediction uncertainty and locate the best prefetching node, aiming for eliminating redundancy [14]. The content retrievability improvement in vehicle-to-infrastructure (V2I) scenario was studied [15]. Content prefetching was decided by an offline management platform to maximize the probability that a user retrieves their desired contents while accounting for the number and mobility models of moving users, with the available cache capacity and link capacity. The optimal caching for producer mobility support was investigated to minimize the total network overhead. By exploiting location predictors and content requests' patterns, the demand contents can be cached proactively before handover occurs [16].

The request aggregation mechanism has been verified to provide considerable performance gain given a large enough content request intensity [27]. The main research on request aggregation mechanism came from the modeling and analyzing of caching performance after considering request aggregation mechanism. The early caching performance analysis was realized in two steps. Firstly, the caching hit and caching miss probabilities were derived with a two-state Markov model. Consequently, the approximate request forwarded and aggregated request probabilities were calculated by multiplying the missed request probability to the coefficients afterwards [28], [29]. In recent years, the realistic scenarios with request aggregation have been considered. It has been theoretically proved that the random process of the request arrival distribution and the content arrival distribution became different due to the request aggregation mechanism [20], [21]. The cache hit rate, average response time of requests and the cache space size distribution occupied by the pending interest table (PIT) were analyzed by using the cache strategy with fixed content sojourn time [30]. Our previous research on the caching performance analysis modeled the states of the arrival content requests as a three-state Markov model to explore a more accurate analysis model [31]. More analysis has been extended to other general traffic models and alternative cache placement policies [32].

Although some existing work has analyzed the benefits of request aggregation mechanism, they have not demonstrated the performance gain to improve the caching strategy optimization yet. Especially, in a massive content requests scenario, the performance gain of adopting the request aggregation mechanism can be evident and crucial to making caching decisions. To fill this aforementioned gap, we will formulate the accurate theoretical models for the traffic load and content retrieval delay to jointly represent the contribution from content caching and request aggregation mechanism. Moreover, since the request aggregation performance gain relies on the content request intensity, we will work on revealing the dynamic content request intensity in addition to the dynamic content popularity to the proposed content caching strategy.

III. SYSTEM MODEL

In this section, the network model, forwarding process of the EN and the dynamic content requests model are described. For the sake of clarity, the frequently used notations in this paper are listed in Table 1.

A. NETWORK MODEL

As shown in Fig. 1, we exhibit the network model for the proposed edge caching. From top to bottom, the network model is composed of four elements, which are content providers, Internet, edge nodes (ENs) and users (UEs). Content providers locate remotely and publish contents. Internet contains several forwarding nodes, which transmit the content requests and the targeting contents. The ENs locate at the edge of the network, which have the abilities of communication, content caching and computing. In addition, the ENs adopt CCN architecture, which enable not only content caching but also request aggregation mechanism to reduce redundancy transmissions. Users access to the Internet through ENs. They send their requests to ENs and receive targeting contents from ENs.

TABLE 1. List of notation.

Notation	Description	Notation	Description
M	Amount of content providers	$\mathcal{P}$	Set of content providers: $\{P_1, \ldots, P_M\}$
K	Amount of contents	$\mathcal{C}$	Set of contents: $\{c_1, \ldots, c_K\}$
$\mathcal{A}$	Set of caching contents, $\mathcal{A} \subset \mathcal{C}$	$\mathcal{C}_m$	Set of contents provided by P_m, $\mathcal{C}_m \in \mathcal{C}$
$\lambda(t)$	Request arrival intensity in interval t	$q_k(t)$	Popularity of content c_k during interval t
$\lambda_k(t)$	Request arrival intensity of content c_k in interval t, $\lambda_k(t) = \lambda(t)\, q_k(t)$	$a_k(t)$	Boolean variable indicating the caching content c_k or not during interval t
f_0	Content size	b_k	Transmission speed of content c_k
d_k	Hop distance for EN to obtain c_k from its content provider	Δ_k	Transmission delay of content c_k from provider to EN, $\Delta_k = f_0/b_k$
ω_T, ω_D	Weighting factors of traffic load and content retrieval delay	$\tau_k(t)$	Content retrieval delay of content c_k considering request aggregation in interval t
L	Cache space size		

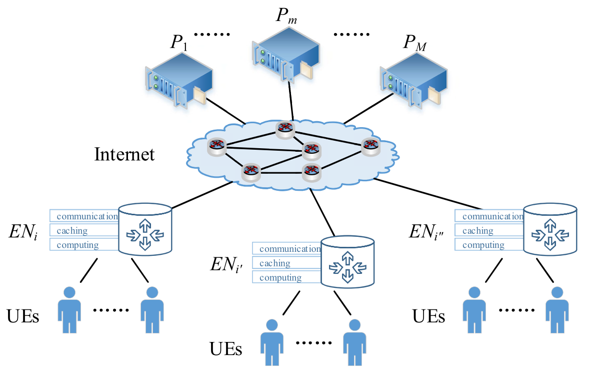


FIGURE 1. Network model for edge caching in CCN.

There are K contents published by M content providers. The content set is denoted as $\mathcal{C} = \{c_1, \ldots, c_K\}$, and the content provider set is expressed as $\mathcal{P} = \{P_1, \ldots, P_M\}$. $\mathcal{C}_m \subset \mathcal{C}$ represents the contents that published by content provider P_m. Assuming that every content is published by only one content provider, it is fulfilled that $\mathcal{C}_m \bigcap \mathcal{C}_{m'} = \emptyset, \forall m \neq m'$ and $\bigcup\limits^{M} \mathcal{C}_m = \mathcal{C}$. Without loss of generality, assuming that all contents are with the unified size f_0. The larger content is able to be sliced into multi contents (chunks) with size of f_0.

Based on this assumption, the size of the cache space embedded to the EN is L contents. Define the set of cached contents as $\mathcal{A} \subset \mathcal{C}$. Let a Boolean vector $\mathbf{a} = [a_k]_{K\times 1}$ represents the caching contents, whose element a_k is defined as

$$a_k = \begin{cases} 1, & c_k \in \mathcal{A}, \\ 0, & \text{otherwise}. \end{cases} \tag{1}$$

In order to satisfy the limitation of the caching space size, the restriction of $\sum\limits_{k=1}^{K} a_k \leq L$ has to be fulfilled.

Assume that N_U users retrieve their desired contents via accessing to the EN. The external content requests generation process arrived at EN is modeled as a Poisson process with intensity of λ. Content popularity of $c_k \in \mathcal{C}$ is defined as the ratio of requests for c_k to the total content requests. The popularity of the content c_k is denoted as q_k. Referring to statistics of Web services, denoting that the content popularity obeys Zipf distribution with skewness factor α as

$$q_k = \frac{1/k^\alpha}{\sum\limits_{i=1}^{K} 1/i^\alpha}, \quad \alpha > 0. \tag{2}$$

Considering the content request intensity λ, the content request intensity to content c_k is $\lambda_k = \lambda q_k, \forall k$.

Assume the network topology is static. The content provider $P_m, m \in [1, M]$ locates D_m hops away from the EN, and the transmission speed of each hop is B_m. For a content $c_k \in \mathcal{C}_m$, the hop distance is $d_k = D_m$ and the transmission speed is $b_k = B_m$. Therefore, transmitting content c_k from its provider to the EN consumes traffic load of $f_0\, d_k$. Content transmission delay of c_k from its provider to EN is defined as Δ_k. It is an index to indicate the network performance, and also affect to user experience. As previously defined, the content size is f_0 and the transmission speed for c_k is b_k. Then the transmission delay of c_k is expressed as $\Delta_k = f_0/b_k$.

B. FORWARDING PROCESS

The forwarding process at a CCN node considering the request aggregation mechanism is shown in Fig. 2 [33]. The CCN architecture utilizes *Interest* and *Data* packets to transmit content requests and the required contents, respectively. There are three core functionality of a cache-enabled router in the CCN architecture: the content store (CS) for caching content, the pending interest table (PIT) which keeps track of the forwarded *Interest*s that are not yet satisfied with a returned *Data* packet, and the forwarding information base (FIB) to guide the *Interest*s forwarding [34]. After receiving a content request, the router first matches the request to its CS. If the cache hit occurs, the router forwards the content to the user. Otherwise, the cache missed request is handled to PIT. The basic function of PIT is to record the name of every requested content and the input interface of the request. These PIT entries provide their recorded interfaces when transmitting the requested contents to the users. After forwarding the requested content, the router will erase the relative PIT entry. For the content request aggregation, an extra judgement

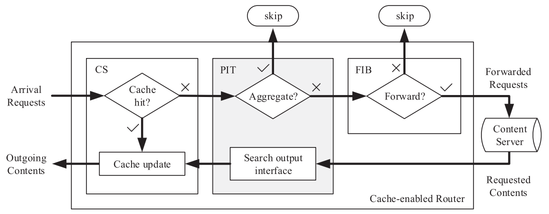


FIGURE 2. Forwarding process at a CCN node.

is implemented on the PIT module, that the cache missed request is matched to the existent PIT entries. When a PIT match is found, the request is decided to be aggregated. The router adds the input interface of the request to the matched PIT entry and then skips the request. During the *Data* forwarding process, the router will multicast the content to all the recorded interfaces at once. The rest missed requests, said the non-aggregated ones, are propagated to the content provider from the interfaces recorded in FIB.

After proceeded by the EN, the arrival content request might belong to one of the three states, which are cache hit, request aggregated and request forwarded respectively. Let p_k^{H} and p_k^{M} represent the cache hit probability and the cache miss probability for acquiring c_k, where $p_k^{\mathrm{H}} + p_k^{\mathrm{M}} = 1$. Denote p_k^{A} and p_k^{NA} as the aggregated and non-aggregated request probabilities, where $p_k^{\mathrm{NA}} + p_k^{\mathrm{A}} = p_k^{\mathrm{M}}$. Furthermore, assume that FIB contains all forwarding information, which makes all the non-aggregated requests being forwarded to the upstream network. Therefore the forwarded request probability is denoted as $p_k^{\mathrm{F}} = p_k^{\mathrm{NA}}$.

Our previous research has explored the analytical expressions of the steady probabilities distribution of cache hit, request forwarded and request aggregated probabilities regarding to the average content sojourn time in cache space δ_k considering the content request aggregation mechanism in CCN [31]. According to this work, these three probabilities are presented as follows,

$$\begin{cases} \pi_{k,\delta_k}^{\mathrm{H}} &= \phi_k \varphi_k \left(1 - e^{-\lambda_k \delta_k}\right) \\ \pi_{k,\delta_k}^{\mathrm{F}} &= \phi_k \varphi_k e^{-\lambda_k \delta_k} \\ \pi_{k,\delta_k}^{\mathrm{A}} &= \phi_k e^{-\lambda_k \delta_k} \left(1 - e^{-\lambda_k \Delta_k}\right), \end{cases} \tag{3}$$

where φ_k and ϕ_k are expressed as

$$\varphi_k = \frac{1}{1 - e^{-\lambda_k \Delta_k}} \sum_{m=1}^{\infty} \frac{\left[\frac{(\lambda_k \Delta_k)^m}{m!} e^{-\lambda_k \Delta_k}\right]^2}{1 - \sum\limits_{i=0}^{m-1} \frac{(\lambda_k \Delta_k)^i}{i!} e^{-\lambda_k \Delta_k}}, \tag{4}$$

and

$$\phi_k = \frac{1}{e^{-\lambda_k \delta_k} \left(1 - e^{-\lambda_k \Delta_k}\right) + \varphi_k}. \tag{5}$$

In addition, δ_k represents the average duration that content c_k is cached in the EN.

For a deterministic content caching strategy, caching content c_k in the EN means that the content caching duration of c_k is $\delta_k \to \infty$. In this case,

$$\begin{cases} \pi_{k,\infty}^{\mathrm{H}}(t) = 1 \\ \pi_{k,\infty}^{\mathrm{F}}(t) = 0 \\ \pi_{k,\infty}^{\mathrm{A}}(t) = 0. \end{cases} \tag{6}$$

On the contrary, when content c_k is not cached in the EN, the content caching duration is $\delta_k = 0$. In this case,

$$\begin{cases} \pi_{k,0}^{\mathrm{H}}(t) = 0 \\ \pi_{k,0}^{\mathrm{F}}(t) = \dfrac{\varphi_k}{\left(1 - e^{-\lambda_k \Delta_k}\right) + \varphi_k} \\ \pi_{k,0}^{\mathrm{A}}(t) = \dfrac{1 - e^{-\lambda_k \Delta_k}}{\left(1 - e^{-\lambda_k \Delta_k}\right) + \varphi_k}, \end{cases} \tag{7}$$

where φ_k refers to (4).

Jointly considering that $a_k(t)$ is the Boolean variable representing whether the content c_k is cached or not during interval t, the cache hit, request forwarded and request aggregated probabilities of a deterministic content caching strategy in CCN are expressed as

$$\begin{cases} p_k^{\mathrm{H}}(t) = a_k(t) \\ p_k^{\mathrm{F}}(t) = \pi_{k,0}^{\mathrm{F}}(t)\,[1 - a_k(t)] \\ p_k^{\mathrm{A}}(t) = \pi_{k,0}^{\mathrm{A}}(t)\,[1 - a_k(t)], \end{cases} \tag{8}$$

where the $\pi_{k,0}^{\mathrm{F}}(t)$ and $\pi_{k,0}^{\mathrm{A}}(t)$ refer to (7).

C. DYNAMIC MODEL

Considering a discrete time system model. Assume the duration of interval t is long enough to represent the content request statistics. During an interval, the content request intensity and content popularity maintain unchanged. In terms of the functionalities of the EN, each interval is partitioned into three steps, whose procedure is depicted in Fig. 3. The first phase of every interval is the "content delivery" phase, in which the caching content set at the EN is denoted as $\mathcal{A}(t) \subset \mathcal{C}$. During this phase, the users generate content requests and retrieve contents from the EN. Meanwhile, the EN observes and derives the content requests feature during this period, including the content request intensity and the content popularity distribution, as well as the cache performance indices, including the cache hit probability, request forwarded probability, and the content retrieval delay. The second phase pertains to "performance judgement", where the EN calculates the content retrieval cost based on the statistics during the prior phase. In addition, the overall content retrieval state of the current period is distinguished to a predefined category. Finally, "proactive

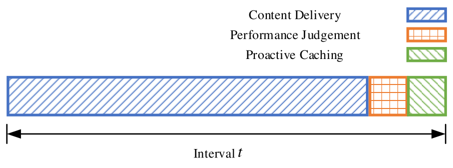


FIGURE 3. Three-step procedure in interval *t*.

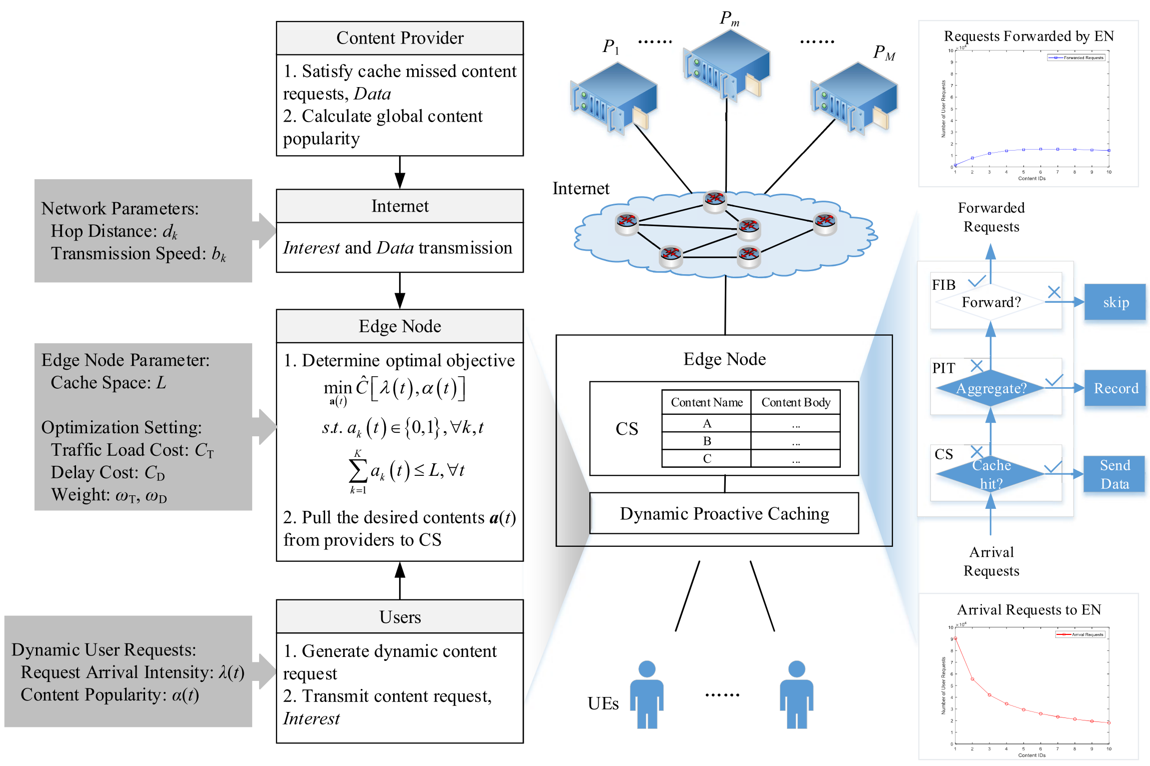

Content Provider
1. Satisfy cache missed content requests, *Data*
2. Calculate global content popularity
Internet
Interest and *Data* transmission
Edge Node
1. Determine optimal objective
$\min_{\mathbf{a}(t)} \hat{C}[\lambda(t), \alpha(t)]$
$s.t.\ a_k(t) \in \{0,1\}, \forall k, t$
$\sum_{k=1}^{K} a_k(t) \le L, \forall t$
2. Pull the desired contents $\boldsymbol{a}(t)$ from providers to CS
Users
1. Generate dynamic content request
2. Transmit content request, *Interest*
Network Parameters:
Hop Distance: d_k
Transmission Speed: b_k
Edge Node Parameter:
Cache Space: L
Optimization Setting:
Traffic Load Cost: C_T
Delay Cost: C_D
Weight: ω_T, ω_D
Dynamic User Requests:
Request Arrival Intensity: $\lambda(t)$
Content Popularity: $\alpha(t)$
P_1
P_m
P_M
Internet
Edge Node
CS
Content Name
Content Body
A
B
C
Dynamic Proactive Caching
UEs
Requests Forwarded by EN
Forwarded Requests
FIB
Forward?
skip
PIT
Aggregate?
Record
CS
Cache hit?
Send Data
Arrival Requests
Arrival Requests to EN

FIGURE 4. **Proposed proactive caching framework.**

caching'' is carried out and the selected contents are stored for the next interval, named $\mathcal{A}(t+1)$. If the idle cache space is not enough for the coming contents, the EN also specifies the existing contents to be replaced. Both the reserved and the coming contents are stored during the next interval, and serve the content requests targeting to the content retrieval cost minimization.

IV. PROBLEM STATEMENT

In this section, we will describe the proactive caching problem in order to minimize the content retrieval cost at the ENs. Prior to elaborating the optimization problem, we would like to give a description to the framework of the proposed proactive caching problem, as illustrated in Fig. 4.

The framework is composed of three columns. The middle part represents the simplified network topology and the key modules to implement the proposed proactive caching algorithm. EN is the most adjacent forwarding node to users, which owns three functionalities: packet forwarding (communication), content caching (caching) and, the most important, caching content decision (computation). Among these functionalities, package forwarding and content caching are operated under the CCN architecture. By means of the computation capability of the EN, the caching content decision module is to decide the optimal caching contents in order to minimize the content retrieval cost. Content providers locate remotely in the upstream network. The distance from the content providers to the EN and the transmission speed of every path are determined by the Internet deployment. The users deliver their content requests to the EN, and receive the required contents from the EN.

The right column in the framework demonstrates the packet forwarding procedure based on CCN architecture. It processes the arrival content requests through CS, PIT, and FIB successively before transmits them to upstream networks. The details of the *Interest*/*Data* forwarding process has been depicted in Section III-B. In order to demonstrate an intuitive sight of the request aggregation mechanism, we plot two line graphs representing the content popularity distribution arrival to and forwarded from the EN respectively. The lower graph represents the user generated content requests arrival to EN, which is so called the local content popularity. And the upper graph represents the popularity distribution of the forwarded content requests from EN, which is so called global content popularity. Since the request aggregation mechanism reduces part of the duplicated content requests, the amount and content popularity of the forwarded requests are changed, which further influences both the traffic load and content retrieval delay consumptions. Therefore, the accurate formulation of the content retrieval cost during content dissemination needs to be studied comprehensively considering the content caching and request aggregation mechanism. Furthermore, the proactive caching algorithm should adapt to the dynamic content requests.

Finally, the left column explains how the dynamic proactive caching function operates. The EN gathers information from both the users locating in its downstream networks and the content providers and Internet locating in its upstream networks. From the downstream, EN experiences the features of content requests. The dynamic content requests are described by two parameters, the content request intensity $\lambda(t)$ and content popularity $\alpha(t)$, which denotes the skewness factor of Zipf distribution, at interval t. From the upstream, EN is aware of the network topology including the hop distance to content providers d_k as well as the transmission speed for each content b_k. All these collected information are utilized to formulate the content retrieval cost function $\hat{C}$. The cost function will represents both traffic load C_{T} and content retrieval delay C_{D}. Finally, the optimal content caching strategy is developed to minimize the proposed content retrieval cost adapting to the dynamic content requests.

In this paper, we study the proactive caching algorithm on one EN to serve the users accessing to it. This algorithm can be deployed to other parallel ENs, based on their parameters of dynamic content requests and network topology. On the basis of the proactive caching framework, we will elaborate the proactive edge caching problem in the rest of this section.

A. COST FUNCTIONS FORMULATION

Two cost function metrics, the traffic load cost and content retrieval delay cost, are formulated in this section. The mathematical model of the request aggregation mechanism is precisely presented for both cost functions.

1) TRAFFIC LOAD

Traffic load in interval t, $C_{\mathrm{T}}(t)$, is composed of two aspects. One is the content dissemination traffic load caused by transmitting the cache missed content requests, denoted as $C_{\mathrm{T1}}(t)$. The other is the proactive caching traffic load caused by prefetching the desired contents, denoted as $C_{\mathrm{T2}}(t)$. Therefore, the traffic load is expressed as

$$C_{\mathrm{T}}(t) = C_{\mathrm{T1}}(t) + C_{\mathrm{T2}}(t). \tag{9}$$

The content dissemination traffic load is incurred during the "Content Delivery" phase. According to the request process in CCN, only the content requests, which are neither cache hit nor aggregated, are forwarded to the upstream network, and introduce traffic load consumptions. So, the content dissemination traffic load generated in interval t is expressed as

$$C_{\mathrm{T1}}(t) = f_0 \lambda\,(t)\, T\,(t) \sum_{k=1}^{K} [1 - a_k(t)]\, q_k\,(t)\, d_k \pi_{k,0}^{\mathrm{F}}(t), \tag{10}$$

where f_0 and d_k are the unified content size and hop distance to acquire c_k. $\lambda\,(t)$ is the content requests arrival intensity during interval t. $T\,(t)$ presents the duration of interval t. $q_k\,(t)$ denotes the content popularity of c_k. $\pi_{k,0}^{\mathrm{F}}\,(t)$ indicates the request forward probability when the content is not cached, referring to (7).

Another traffic load cost is generated by the procedure of proactive caching, when the EN downloads the contents for the next interval. In order to further reduce the proactive caching traffic load, the contents that are decided to be cached in the contiguous intervals will be maintained by CS, instead of being downloaded duplicately. Therefore, the proactive caching traffic load cost is defined as

$$C_{\mathrm{T2}}\,(t) = f_0 \sum_{k=1}^{K} a_k\,(t)\,[1 - a_k(t-1)]d_k, \tag{11}$$

where $[1 - a_k(t-1)]\, a_k\,(t) = 1$ represents that c_k is prefetched from the content provider.

2) CONTENT RETRIEVAL DELAY

Another component of the content retrieval cost is induced by content retrieval delay, denoted as $C_{\mathrm{D}}\,(t)$. The content retrieval delay is categorized into three possible types according the three states of the arrival content request: i) The cache hit requests are immediately satisfied, which introduces zero extra delay. ii) The aggregated requests consumers partially shortened content retrieval delay, since they are able to reuse the previously required contents. iii) The forwarded content requests cause the complete transmission delay from the content providers. Therefore, the content retrieval delay $C_{\mathrm{D}}\,(t)$ is expressed as

$$\begin{aligned} C_{\mathrm{D}}\,(t) = \lambda\,(t)\,T\,(t) &\sum_{k=1}^{K} [1 - a_k(t)] q_k\,(t) \\ &\times \left[\pi_{k,0}^{\mathrm{F}}\,(t)\,\Delta_k + \pi_{k,0}^{\mathrm{A}}\,(t)\,\bar{\tau}_k\,(t)\right], \end{aligned} \tag{12}$$

where Δ_k indicates the content transmission delay from the content providers to the EN, $\bar{\tau}_k\,(t)$ represents the average delay that an aggregated request has to wait until the previously request content arrives at the EN. $\pi_{k,0}^{\mathrm{F}}\,(t)$ and $\pi_{k,0}^{\mathrm{A}}\,(t)$ represent the request forwarded and aggregated probabilities when the required content is not cached, whose expressions are referring to (7).

To clarify the average content retrieval delay for the aggregated content requests, a duration denoted as 'a request aggregation round' is defined. A request aggregation around starts from $t_k^{(0)}$ when the request for c_k is forwarded upstream and terminates at $t_k^{(0)} + \Delta_k$ when the previously requested c_k arrives back to the EN. The request aggregation round and the events that occur during this duration are depicted in Fig. 5.

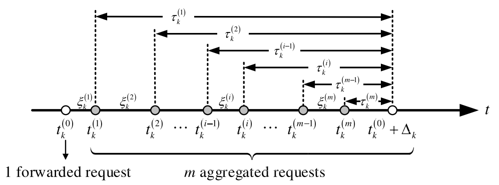


FIGURE 5. **The sequence diagram of a request aggregation round.**

Assume that there are $m \in \mathbb{Z}^{+}$ requests for c_k arrives at EN in an round, and the time interval between two adjacent arrival requests for c_k is random process $\left\{\xi_k^{(i)}, i = [1, m]\right\}$.

The arrival time of the i^{th} aggregated request in this content request aggregation round is expressed as $t_k^{(i)} = t_k^{(0)} + \sum_{j=1}^{i} \xi_k^{(j)}$. The content retrieval delay of the i^{th} aggregated request is $\tau_k^{(i)} = \Delta_k - \sum_{j=1}^{i} \xi_k^{(j)}$. Therefore, the mean content retrieval delay of the m aggregated requests in a content request aggregation round, denoted as $\hat{\tau}_{k,m}$, is expressed as

$$\hat{\tau}_{k,m}(t) = \Delta_k - \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{i}\xi_k^{(j)}, \quad (m \in \mathbb{Z}^+). \tag{13}$$

Furthermore, the average content retrieval delay of the m aggregated requests is expressed as $\bar{\tau}_{k,m}(t) = \mathbb{E}\left[\hat{\tau}_{k,m}(t)\right]$. Given the content request arrival process for c_k is modeled as a Poisson process with intensity of $\lambda_k(t)$, the probability of m request arriving during a request aggregation round with interval of Δ_k is

$$\mathrm{P}\{N(\Delta_k) = m\} = \frac{(\lambda_k \Delta_k)^m}{m!} e^{-\lambda_k \Delta_k}, \quad m \in \left\{0, \mathbb{Z}^+\right\}. \tag{14}$$

Based on the expressions of (13) and (14), the overall content retrieval delay for all aggregated requests during interval t is expressed as

$$\bar{\tau}_k(t) = \sum_{m=1}^{\infty} \hat{\tau}_{k,m}(t) \mathrm{P}\{N(\Delta_k) = m\}. \tag{15}$$

B. PROPOSED OPTIMIZATION PROBLEM

The network traffic offloading and the content retrieval delay reduction are two kernel purpose of edge caching. Our target is to formulate an optimization problem to achieve both aforementioned performance gain. However, there exists a magnitude gap between the traffic load and delay. We employ normalization to eliminate the magnitude gap. $C_{\mathrm{T},\emptyset}(t)$ and $C_{\mathrm{D},\emptyset}(t)$ represent the original traffic load and content retrieval delay benefiting from neither content caching nor request aggregation mechanism, which are expressed as

$$\begin{cases} C_{\mathrm{T},\emptyset}(t) = f_0 \lambda(t) T(t) \sum_{k=1}^{K} q_k(t) d_k \\ C_{\mathrm{D},\emptyset}(t) = \lambda(t) T(t) \sum_{k=1}^{K} q_k(t) \Delta_k. \end{cases} \tag{16}$$

Therefore, we denote $C'_{\mathrm{T}}(t)$ and $C'_{\mathrm{D}}(t)$ to represent the normalized traffic load and content retrieval delay as

$$\begin{cases} C'_{\mathrm{T}}(t) = \dfrac{C_{\mathrm{T}}(t)}{C_{\mathrm{T},\emptyset}(t)} \\ C'_{\mathrm{D}}(t) = \dfrac{C_{\mathrm{D}}(t)}{C_{\mathrm{D},\emptyset}(t)}, \end{cases} \tag{17}$$

where $C_{\mathrm{T}}(t)$ and $C_{\mathrm{D}}(t)$ have been defined previously in (9) and (12).

After filling the magnitude gap between the traffic load and content retrieval delay, we formulate the proposed optimization edge caching problem as a weighted sum for multi-objective optimization problem. Denote ω_{T} and ω_{D} as the weighting factors of the traffic load and content retrieval delay, representing their importance in the total content retrieval cost function. The comprehensively weighted content retrieval cost function is constructed as

$$C(t) = \omega_{\mathrm{T}} C'_{\mathrm{T}}(t) + \omega_{\mathrm{D}} C'_{\mathrm{D}}(t). \tag{18}$$

Without loss of generality, we assume that the two weighting factors satisfies

$$\omega_{\mathrm{T}} + \omega_{\mathrm{D}} = 1. \tag{19}$$

Based on this assumption, the proposed optimization problem has two special circumstances. When $\omega_{\mathrm{T}} = 1$ and $\omega_{\mathrm{D}} = 1 - \omega_{\mathrm{T}} = 0$, the proposed cost function only concerns the traffic load. On the contrary, when $\omega_{\mathrm{T}} = 0$ and $\omega_{\mathrm{D}} = 1$, the proposed cost becomes content retrieval delay. By varying the weighting factors, it is able to adjust the importance of the traffic load and content retrieval delay in the content retrieval cost function according to the system design requirements.

The proactive caching problem in the dynamic content requests scenario is formulated as an optimization problem, whose objective is to minimize the long-term average content retrieval cost for each content request, denoted as $\hat{C}$,

$$\hat{C} = \frac{\sum_{t=1}^{\infty} C(t)\lambda(t) T(t)}{\sum_{t=1}^{\infty} \lambda(t) T(t)}. \tag{20}$$

The optimization problem is indicated as follows,

$$\min_{\mathcal{A}} \hat{C} \tag{21a}$$

$$\text{s.t. } a_k(t) \in \{0, 1\}, \quad \forall k, t \tag{21b}$$

$$\sum_{k=1}^{K} a_k(t) \le L, \quad \forall t, \tag{21c}$$

where $\mathcal{A}$ represents set of caching contents. The definition of the Boolean variable $a_k(t)$ refers to (1). Equation (21c) restricts that the maximum amount of caching contents should not exceed the cache space size L.

The proposed optimization problem (21) is an integer programming problem. We discuss its time complexity as follows. The local content retrieval cost during interval t has been modeled as (18). Because of considering the caching replacement traffic load cost, the performance is affected by the caching contents of both the current and the previous intervals, namely $\{a_k(t)\}$ and $\{a_k(t-1)\}$ respectively, as (11). Assuming that the set of caching contents in the previous interval is known, the local content retrieval cost minimization problem transforms to a typical Knapsack problem trying to determine whether to cache each content, so that the local caching contents is no more than the cache space and the content retrieval cost during this interval is as small as possible [35], whose time complexity is NP-complete. Therefore, the proposed proactive caching problem (21) is also a NP-complete problem. To solve the proposed proactive caching problem, we will develop a Q-learning algorithm adapting to the dynamic content requests and derive the long-term minimization of the content retrieval cost.

V. PROPOSED Q-LEARNING ALGORITHM

Since the proposed problem is NP-complete,we employ Q-learning algorithm to derive the optimal caching strategy. We first model the proposed proactive edge caching problem with the dynamic content requests process to a Markov decision process (MDP). Particularly, the reward function is carefully design to represent the performance in upcoming interval with the assistant of the temporal proactive caching strategy. Q-learning is implemented to jointly infer the optimal policy and obtain the long-term cost minimization.

A. FORMULATION OF A MARKOV DECISION PROCESS

The dynamic content request intensity and content popularity distribution are modeled using two independent Markov chains. The content requests process arrival to the EN is modeled as a Poisson process with intensity of $\lambda(t)$ in interval t. Assume that request intensity is generated by an underlying Markov process with $|\mathcal{L}|$ states collected in the set $\mathcal{L} = \left\{\lambda^{(1)}, \ldots, \lambda^{(|\mathcal{L}|)}\right\}$. Similarly, the content popularity distribution $\mathbf{q}(t)$ is generated by an underlying Markov process with $|\mathcal{Q}|$ states collected in the set $\mathcal{Q} = \left\{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(|\mathcal{Q}|)}\right\}$, where $\mathbf{q}$ is a $K \times 1$ vector representing the popularity of all K contents. Assume that the set of $\mathcal{L}$ and $\mathcal{Q}$ are known to the EN. Meanwhile, neither transition probability of the two Markov chain are considered unknown by the EN. Then we describe the dynamic process as a MDP characterized by a quadruple $(\mathcal{S}, \mathcal{Z}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ and $\mathcal{Z}$ are finite state and action spaces, respectively. $\mathcal{P}$ is the transition probability set, and $\mathcal{R}$ represents the immediate reward set.

1) STATE

Given the request intensity $\mathcal{L}$, the content popularity $\mathcal{Q}$ as well as the caching contents $\mathbf{a}$, the overall set of system states in the network is

$$\mathcal{S} = \left\{\mathbf{s} \middle| \mathbf{s} = \left\{\lambda, \mathbf{q}^{\top}, \mathcal{A}\right\}, \lambda \in \mathcal{L}, \mathbf{q} \in \mathcal{Q}\right\}. \tag{22}$$

The size of $\mathcal{S}$ is $|\mathcal{L}| \times |\mathcal{Q}| \times |\{\mathcal{A}\}|$. The state in interval t is denoted as $\mathbf{s}(t) \in \mathcal{S}$.

2) ACTION

The EN's action is to refresh its cache space according to its caching strategy. The action consists of two step: to decide the eviction contents and to pull the requested contents from their content providers. Let $\mathcal{Z}^{-}(t) = \{c_k(t)\}, c_k \in \mathcal{A}(t-1)$ represent the eviction contents the EN evicts from its cache space in interval t. Denote $\mathcal{Z}^{+}(t) = \{c_k(t)\}, c_k \in (\mathcal{C} \setminus \mathcal{A}(t-1))$ as the prefetched contents that the EN pulls and caches in interval t. Therefore, the strategy is to determine $\mathcal{Z}(t) = (\mathcal{Z}^{-}(t), \mathcal{Z}^{+}(t))$. After these two steps, the caching contents in the EN in interval t becomes $\mathcal{A}(t)$ as

$$\mathcal{A}(t) = \mathcal{A}(t-1) \setminus \mathcal{Z}^{-}(t) \cup \mathcal{Z}^{+}(t), \quad \forall t > 0, \tag{23}$$

where symbol $\setminus$ represents content eviction, and symbol $\cup$ represents prefetching fresh contents.

3) TRANSITION PROBABILITY

$P_{\mathcal{Z}}(\mathbf{s}, \mathbf{s}') \in \mathcal{P}, \forall \mathbf{s}, \mathbf{s}' \in \mathcal{S}$ is the probability that action $\mathcal{Z}$ in state $\mathbf{s}$ at interval $t-1$ will lead to state $\mathbf{s}'$ at interval t, that is

$$P_{\mathcal{Z}}(\mathbf{s}, \mathbf{s}') = \Pr\left\{\mathbf{s}(t) = \mathbf{s}' \,|\, \mathbf{s}(t-1) = \mathbf{s}, \mathcal{Z}(t) = \mathcal{Z}\right\}. \tag{24}$$

4) IMMEDIATE REWARD

Our purpose is to minimize the content retrieval cost, which means that the actions that leads to the lower cost is supposed to obtain the higher immediate reward. Therefore, we define the reward function as

$$R_{\mathcal{Z}}(\mathbf{s}, \mathbf{s}') = C^{-2}\left[\lambda(t), \mathbf{q}(t), \mathcal{A}(t), \mathcal{A}(t-1)\right], \tag{25}$$

where the cost function C refers to (18). The size of the immediate reward $\mathcal{R}$ is $|\mathcal{S}|^2 \times |\mathcal{Z}|$.

Let us now define the policy function $\pi : \mathcal{S} \to \mathcal{Z}$, which maps state to the action. Under policy $\pi(\cdot)$, the action $\mathbf{z}(t+1) = \pi[\mathbf{s}(t)]$ represents the proactive caching contents during interval $t+1$ under the state $\mathbf{s}(t)$. Respecting to the policy π, the caching performance is measured through the state-value function

$$V_{\pi}\left[\mathbf{s}(t+1)\right] = \lim_{T\to\infty} E\left\{\sum_{\tau=t}^{T} \gamma^{\tau-t} C^{-2}_{\pi[\mathbf{s}(\tau)]}\left[\mathbf{s}(\tau), \mathbf{s}(\tau+1)\right]\right\}, \tag{26}$$

which represents the total average content retrieval cost incurred over an infinite time with a discounted factor $\gamma \in [0, 1)$. A large γ implies that the current state value is highly affected by the future.

Our purpose is to derive the optimal policy π^*, which could achieve the maximal value of any state $\mathbf{s}$

$$\pi^* = \arg\max_{\pi\in\Pi} V_{\pi}(\mathbf{s}), \quad \forall \mathbf{s} \in \mathcal{S}, \tag{27}$$

where Π is the set of all feasible policies.

In MDP, Bellman equations express the state-value function (26) by a recursion for expected reward

$$V_{\pi}(\mathbf{s}) = C^{-2}_{\pi(\mathbf{s})}(\mathbf{s}, \mathbf{s}') + \gamma \sum_{\mathbf{s}'\in\mathcal{S}} P_{\pi(\mathbf{s})}(\mathbf{s}, \mathbf{s}') V_{\pi}(\mathbf{s}'), \quad \forall \mathbf{s}, \mathbf{s}'. \tag{28}$$

The equation describes the expected reward for taking the action prescribed by the policy π.

The optimal policy π^* is referred to as the Bellman optimality equation,

$$V_{\pi^*}(\mathbf{s}) = \max_{\mathbf{z}}\left\{C^{-2}_{\mathbf{z}}(\mathbf{s}, \mathbf{s}') + \gamma \sum_{\mathbf{s}'} P_{\mathbf{z}}(\mathbf{s}, \mathbf{s}') V_{\pi^*}(\mathbf{s}')\right\}, \tag{29}$$

where π^* indicates the optimal policy and $V_{\pi^*}(\mathbf{s})$ refers to the value function of the optimal policy.

For a policy π, define Q value, named the state-action value function as

$$Q_{\pi}(\mathbf{s}, \mathbf{z}) = C^{-2}_{\pi(\mathbf{s})}(\mathbf{s}, \mathbf{s}') + \gamma \sum_{\mathbf{s}'} P_{\mathbf{z}}(\mathbf{s}, \mathbf{s}') V_{\pi}(\mathbf{s}'), \tag{30}$$

where the Q value is the expected discounted reward for executing action $\mathbf{z}$ at state $\mathbf{s}$ and following policy π thereafter.

Given the transition probability $\mathcal{P}$ and the initialized policy π_0, the best policy is derived through the policy iteration algorithm as follow,

1) Policy evaluation: Determine $V_{\pi_t}(\mathbf{s})$ for all states $\mathbf{s} \in \mathcal{S}$ under the current policy π_t, given the value-state function in (28).
2) Q function update: For each state-action pair, calculate its state-action value function $Q_{\pi_t}(\mathbf{s}, \mathbf{z})$ as (30).
3) Policy update: Update the policy π_{t+1} using

$$\pi_{t+1}(\mathbf{s}) = \arg\max_{\boldsymbol{\zeta}} Q_{\pi_t}(\mathbf{s}, \boldsymbol{\zeta}), \quad \forall \mathbf{s} \in \mathcal{S}. \tag{31}$$

The iteration terminates until the difference $\pi_{t+1}(\mathcal{S}) = \pi_t(\mathcal{S})$, when the caching policy is proved to converge to the best policy.

The per iteration complexity of the policy iteration algorithm is $\mathcal{O}\left(|\mathcal{S}|^3 + |\mathcal{S}|^2|\mathcal{Z}|\right)$ [36]. Furthermore, this algorithm depends on the known transition probability $\mathcal{P}$, which is not practical in real systems. In contrast, the Q-learning algorithm does not require to estimate the model of the environment, i.t. the transition probability $\mathcal{P}$ with size of $|\mathcal{S}|^2 \times |\mathcal{Z}|$, to obtain the optimal policy π^* as well as the optimal state value $V_\pi(\mathbf{s})$.

B. Q-LEARNING ALGORITHM

Q-learning is a form of model-free reinforcement learning. It provides the EN with the capability of learning to act optimally in Markov domains by experiencing the consequences of actions [37]. Q-learning jointly infers the optimal policy π^*, and estimate the optimal state-action value function of

$$Q^*(\mathbf{s}, \mathbf{z}) = Q_{\pi^*}(\mathbf{s}, \mathbf{z}), \quad \forall \mathbf{s}, \mathbf{z}. \tag{32}$$

It is straightforward to show that

$$V^*(\mathbf{s}) = V_{\pi^*}(\mathbf{s}) = \max_{\boldsymbol{\zeta}} Q^*(\mathbf{s}, \boldsymbol{\zeta}), \tag{33}$$

and that if π^* is the optimal policy which the maximum is attained, it is easily decide what is the optimal todo as

$$\pi^*(\mathbf{s}) = \arg\max_{\boldsymbol{\zeta}} Q^*(\mathbf{s}, \boldsymbol{\zeta}), \quad \forall \mathbf{s} \in \mathcal{S}. \tag{34}$$

During every interval, the estimated Q value is updated according to

$$\begin{aligned}
&\hat{Q}_{\pi_t}[\mathbf{s}(t-1), \mathbf{z}(t)] \\
&\quad = (1-\beta_t)\hat{Q}_{\pi_{t-1}}[\mathbf{s}(t-1), \mathbf{z}(t)] + \beta_t \\
&\qquad \cdot \Big\{ C^{-2}[\mathbf{s}(t-1), \mathbf{z}(t) | \lambda(t), \mathbf{q}(t)] \\
&\qquad + \gamma \max_{\boldsymbol{\zeta}} \hat{Q}_{\pi_{t-1}}[\mathbf{s}(t), \boldsymbol{\zeta}] \Big\},
\end{aligned} \tag{35}$$

where β_t is the learning rate, which affects the learning result to the Q values. γ is discount factor representing that the future action influence to the current decision making. While the rest Q values keep unchanged.

The Q-learning scheme for the proactive caching policy is listed under **Algorithm 1**.

Algorithm 1 Q-Learning Based Proactive Caching Algorithm

Input: f_0, L, $\{d_k\}$, $\{b_k\}$, ω_{T}, ω_{D}, e-greedy value ϵ, discount factor γ, a relatively small value ε.
Output: The optimal caching strategy π^*.

```
1:  Initialization: random $\mathbf{s}(0)$, $Q_0(\mathbf{s}, \mathbf{z}) = 0$.
2:  repeat
3:      Generate random value $\epsilon(t)$.
4:      $\mathbf{z}(t) = \begin{cases} \arg\max_{\boldsymbol{\zeta}} \hat{Q}_{\pi_t}[\mathbf{s}(t-1), \boldsymbol{\zeta}], & \epsilon(t) > \epsilon, \\ \text{random } \mathbf{z}, & \epsilon(t) \le \epsilon. \end{cases}$
5:      Derive caching contents in interval $t$, $\mathcal{A}(t)$, as (23).
6:      Observe state in interval $t$, $\mathbf{s}(t)$.
7:      Calculate the immediate reward as (25).
8:      Update $\hat{Q}_{\pi_t}[\mathbf{s}(t-1), \mathbf{z}(t)]$ value as (35).
9:      Update the policy $\pi_{t+1}$ as (31).
10:     Calculate the state-value function $V_\pi[\mathbf{s}(t)]$ as (26).
11: until $|V_{\pi_{t+1}}[\mathcal{S}] - V_{\pi_t}[\mathcal{S}]| \le \varepsilon$
12: Return $\pi^* = \pi_t$.
```

C. CONVERGENCE ANALYSIS

The convergence of the Q-learning has been proved. According to the **Theorem 1** in [38], given the predefined finite MDP $(\mathcal{S}, \mathcal{Z}, \mathcal{P}, \mathcal{R})$, the Q-learning algorithm, whose update rule refers to (35), converges to w.p.1 to the optimal Q-function as long as the learning rate satisfies

$$\sum_t \beta_t(\mathbf{s}, \mathbf{z}) = \infty \quad \sum_t \beta_t^2(\mathbf{s}, \mathbf{z}) < \infty \tag{36}$$

for all $(\mathbf{s}, \mathbf{z}) \in \mathcal{S} \times \mathcal{Z}$. Therefore, we define $t(\mathbf{s}, \mathbf{z})$ as the amount that state-action pair $(\mathbf{s}, \mathbf{z})$ is visited. Furthermore, define $\beta_t(\mathbf{s}, \mathbf{z}) = t(\mathbf{s}, \mathbf{z})^{-1}$, which satisfies the convergence requirement clarified in (36).

VI. SIMULATION RESULTS

In this section, in order to verify the performance of the proposed proactive caching algorithm, we simulate the proposed proactive caching algorithm and compare it with following benchmarks caching algorithms:

- Greedy algorithms: the linear programming caching algorithms which proposes the caching content decision based on the previous interval. Since the content retrieval cost is composed of weighted sum of traffic load and content retrieval delay, we proposes two Greedy algorithms, i.t. Greedy TL2DL and Greedy DL2TL. Greedy TL2DL searches $2L$ candidates caching contents which achieve maximum traffic load firstly and then picks L contents which introduce largest content retrieval delay among the $2L$ candidates. Greedy DL2TL is implemented in the opposite order.
- MPC algorithms: the widely-used caching algorithm, in which the most popular contents in current interval is cached in the next interval. The local MPC algorithm determines the most popular contents according to the received content request at EN. On the other hand,

TABLE 2. **Simulation parameters.**

Parameters	Range	Default Value
Content request intensity λ (requests per second)	$[1, 10]$	$\lambda^{(1)} = 1$, $\lambda^{(2)} = 10$
Skewness factor in Zipf distribution α	$[0.3, 1.5]$	$\alpha^{(1)} = 0.3$, $\alpha^{(2)} = 0.7$
Weight of traffic load cost ω_{T} and content retrieval delay cost ω_{D}	$[0, 1]$	$\omega_{\mathrm{T}} = 0.2$, $\omega_{\mathrm{D}} = 0.8$
Number of contents K (contents)	-	10
Cache space L (contents)	-	2
Transmission Delay Δ (ms)	$[10, 100]$	100

the global MPC algorithm determines the most popular contents arrival to content providers. Influenced by the request aggregation mechanism, local MPC is no longer coincide with global MPC.

- RND algorithm: randomly caching contents regardless the content request characteristics and network environments. The performance of RND is seen as a baseline.

A. SIMULATION ASSUMPTIONS

Assume that 10 contents are published by 10 content providers. Every content provider publishes only one content. The network topology of Internet is an arbitrary network. Let's imagine that there is an end to end path from every content provider to the EN. The number of intermediate nodes between the content providers to the EN are randomly set between 0 and 9. Therefore, the hop accounts of the paths are set between 1 and 10 hops. Meanwhile, the transmission rates of the paths are selected between 10Mbps and 100Mbps. In order to reflect the network heterogeneity, the hop accounts and transmission rates of the paths are independently set. Consequently, a path with large hop account may serve with high transmission rate. The content size is $f_0 = 1$Mb identically. Considering the confliction between the vast amount of contents and the limited cache space, the cache space is set to be $L = 2$, which is 20% of the total contents at most.

Assuming that the request arriving at the EN obeys the Poisson process with strength λ. Considering the dynamic content request, the content request intensity varies among [1, 10] requests per second. Content popularity is modeled by Zipf distribution, with the skewness factor of α. The value range of α is [0.3, 1.5]. The larger α implies the high consistency of the user's preference for a small amount of contents, while the requests for the rest contents is relatively few. The weight of the traffic load cost ω_{T} and the content retrieval delay cost ω_{D} vary from 0 to 1 and satisfy $\omega_{\mathrm{T}} + \omega_{\mathrm{D}} = 1$. The default value of weighting factors are set as $\omega_{\mathrm{T}} = 0.2$ and $\omega_{\mathrm{D}} = 0.8$. Given this parameter setting, the content retrieval delay cost dominates the proposed content retrieval cost, while the proportion of the traffic load cost is relatively low. The aforementioned parameters and the default value adopted in the simulation are concluded in Table 2.

The dynamic content request intensity and content popularity distribution are modulated by two independent Markov process. Each Markov process has two states, respectively. Define the content requests intensity λ to present the busy hour and the idle hour. The content request intensity transition probability is

$$\mathbf{P}^{(\lambda)} = \begin{bmatrix} p_{11}^{(\lambda)} & p_{12}^{(\lambda)} \\ p_{21}^{(\lambda)} & p_{22}^{(\lambda)} \end{bmatrix} = \begin{bmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{bmatrix}. \tag{37}$$

While the transition probability of dynamic content popularity is represented as

$$\mathbf{P}^{(\alpha)} = \begin{bmatrix} p_{11}^{(\alpha)} & p_{12}^{(\alpha)} \\ p_{21}^{(\alpha)} & p_{22}^{(\alpha)} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.75 & 0.25 \end{bmatrix}. \tag{38}$$

The dynamic content popularity contains two aspects. One aspect is the skewness factor of the Zipf distribution α, which reflects the centralization degree of the content requests. The other aspect is the alternation of the most popular contents. In state 1, the descend content popularity follows the order of $\{c_1, \ldots, c_{10}\}$. In state 2, the descend content popularity follows the order of $\{c_3, \ldots, c_{10}, c_1, c_2\}$.

B. SIMULATION RESULTS

In this section, we express the simulation results of the proposed proactive edge caching algorithm in three aspects. First, we verify the convergence of the proposed Q-learning algorithm, which is the foundation of adopting Q-learning algorithm to solve the proactive caching problem. Then, with the purpose of verifying the effectiveness of the proposed caching algorithm with highly dynamic content requests, the performance of the proposed proactive caching algorithm is demonstrated with diverse content request intensity, content popularity, transmission speed and weighting factor. Finally, to explore the algorithm's performance gain under massive content requests, we compare the proposed caching algorithm with and without adopting the request aggregation mechanism.

The convergence of the proposed Q-learning algorithm is shown in Fig. 6. According to the simulation assumption, the Q table is size of $|\mathcal{S}|^2 \times |\mathcal{Z}| = (|\mathcal{L}| \times |\mathcal{Q}|)^2 \times C_K^L = 8100$. The simulation results show that when Q-learning algorithm iterates to about 1.4×10^4 times, until it converges to the optimal performance.

From Fig. 7 to Fig. 10, we demonstrate the proposed content retrieval cost, the traffic load and the content retrieval delay of the proposed proactive caching algorithm as well as the benchmark algorithms, with vary content request intensity, content popularity, transmission speed and weighting factors, respectively.

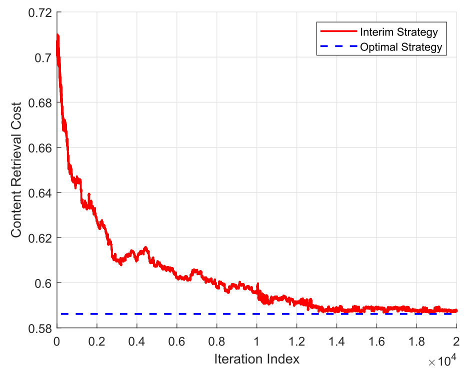


FIGURE 6. Q-learning strategy convergence.

Fig. 7 plots the content retrieval cost, traffic load and content retrieval delay varying with content request intensity. In this simulation, $\lambda^{(1)} = 1$, $\lambda^{(2)}$ varies from 1 to 10, $\alpha^{(1)} = 0.3$, $\alpha^{(2)} = 0.7$, $\omega_T = 0.2$ and $\omega_D = 0.8$. As shown in Fig. 7(a), all algorithms achieve lower content retrieval cost with larger content request intensity. The proposed Q-learning algorithm always outperforms the benchmark algorithms, which verifies that the proposed algorithm is able to learn and therefore forecast the content request dynamics. The Greedy algorithms are the sub-optimal algorithms. The Greedy DL2TL performs better than the Greedy TL2DL. It is because that the content retrieval delay dominates 80% of the overall content retrieval cost, the Greedy DL2TL are likely to selects the caching contents that consumes less content retrieval delay. In addition, the MPC algorithms perform unsatisfactory, because of the dynamic content requests and the distance and transmission speed difference for each content retrieval path. The performance gap between the two MPC algorithms is caused by the request aggregation mechanism. It demonstrates that the request aggregation mechanism not only influences the upstream content popularity distribution, but also changes the sort of the content popularity distributions. Furthermore, it is worth to mention that RND algorithm plays a special role among these benchmark algorithms. It is well known that the performance of RND algorithm does not affected by content request intensity if without request aggregation mechanism. While, the decreasing content retrieval cost of RND algorithm reveals that, by adopting the request aggregation mechanism, larger proportion of the duplicated content requests are aggregated at the EN. Therefore, it verifies that the request aggregation benefits to traffic load and the content retrieval delay reduction. The performance gap between the proposed algorithm to RND algorithm comes from the intelligent content caching decision, targeting the content retrieval cost minimization. In Fig. 7(b) and Fig. 7(c), both traffic load and content retrieval delay decrease with larger content request intensity. Because the content retrieval delay dominates the content retrieval cost, the proposed algorithm prefers to cache the contents that can shorten the content retrieval delay.

The influence of content popularity to the content retrieval cost is demonstrated in Fig. 8. In this simulation, $\lambda^{(1)} = 1$, $\lambda^{(2)} = 10$, $\alpha^{(1)} = 0.3$, $\alpha^{(2)}$ varies from 0.3 to 1.5, $\omega_T = 0.2$ and $\omega_D = 0.8$. From Fig. 8(a), the content retrieval cost of the proposed Q-learning algorithm performs best among all these algorithms, which is no more than 35% with varying skewness factor of Zipf distribution α. However, from a relative uniform content popularity distribution ($\alpha^{(2)} = 0.3$) to a concentrative content popularity distribution ($\alpha^{(2)} = 1.5$), the content retrieval cost only decreases 8%. Given the massive content requests, the request aggregation mechanism already produces a remarkable proportion of the aggregated content requests, which significantly reduces the redundancy transmission. The popularity distribution of the forwarded content requests is flattened, which eliminates the performance difference between caching the popular contents and the non-popular ones. That is the reason why significantly increasing the skewness factor of the Zipf distribution only introduces very little performance gain under a massive content requests circumstance. When the content requests is dense, it is the request aggregation mechanism rather than the content caching technique contributes dominant performance gain in CCN. Fig. 8(b) and Fig. 8(c) represent simulation results of the traffic load and content retrieval delay with varying content popularity. It is worthy to notice that the proposed algorithm achieves no more than 1/5 traffic load and 40% content retrieval delay even when $\alpha^{(2)} = 0.3$.

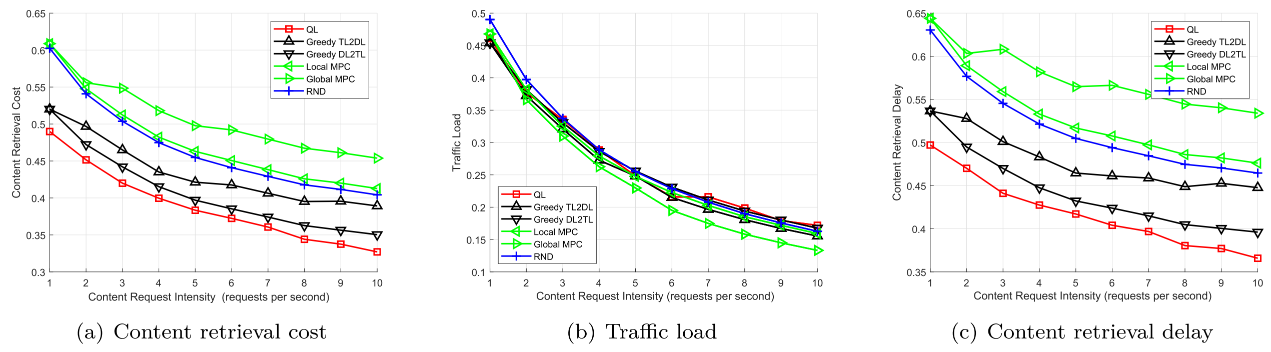
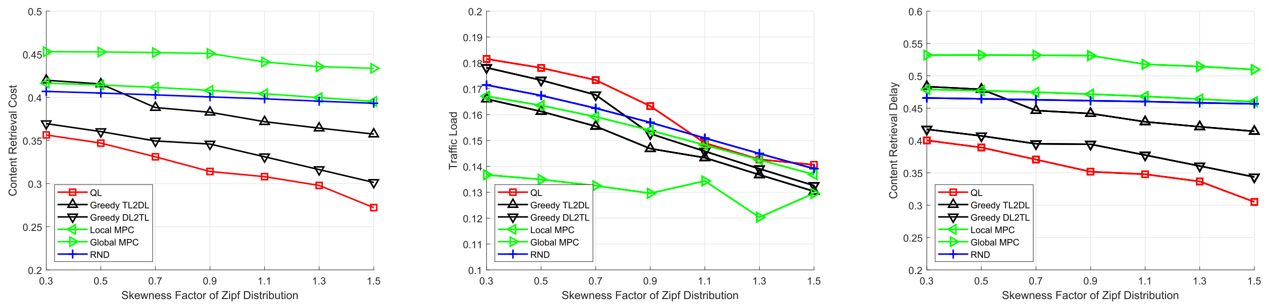
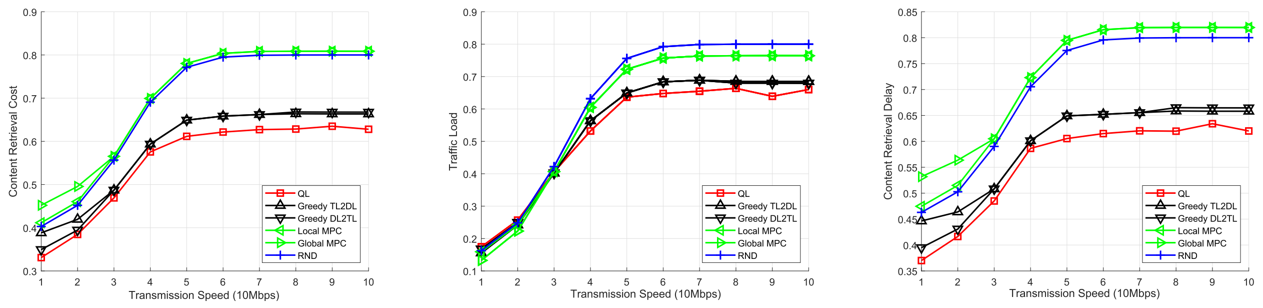


FIGURE 7. Network performance comparison with varying content request intensity.



(a) Content retrieval cost (b) Traffic load (c) Content retrieval delay

FIGURE 8. **Network performance comparison with varying content popularity.**



(a) Content retrieval cost (b) Traffic load (c) Content retrieval delay

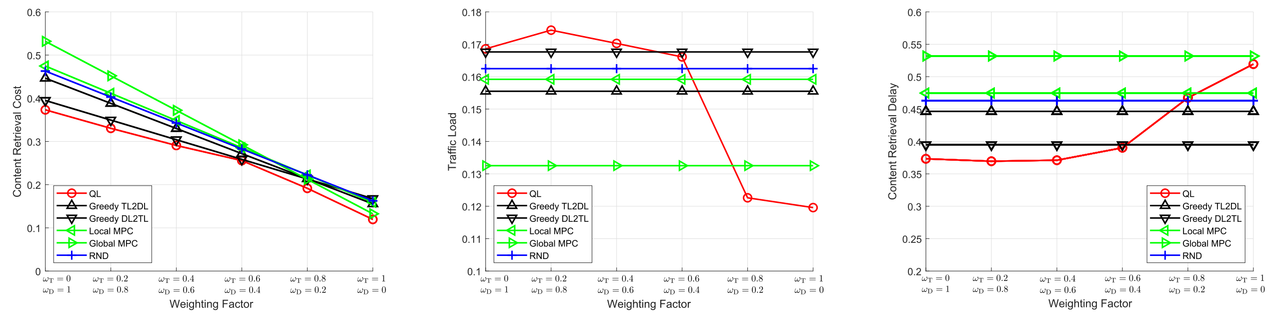
FIGURE 9. **Network performance comparison with varying transmission speed.**

The simulation result indicates that even the content popularity is relatively flat, the proposed algorithm is able to provide good enough performance.

Increasing the transmission speed introduces performance variation of the proposed caching algorithm, as shown in Fig. 9. In this simulation, $\lambda^{(1)} = 1$, $\lambda^{(2)} = 10$, $\alpha^{(1)} = 0.3$, $\alpha^{(2)} = 0.7$, $\omega_{\mathrm{T}} = 0.2$ and $\omega_{\mathrm{D}} = 0.8$. Transmission speed increase from 10Mbps to 100Mbps. Fig. 9(a) represents that the content retrieval cost raises from 33% to 63% when transmission speed increase from 10Mpbs to 100Mbps. When transmission speed exceeds 70Mbps, the content retrieval cost of all algorithms becomes nearly constant. At this time, the request aggregation rarely occurs, and the cost gain comes from content caching. Comparing Fig. 9(b) with Fig. 9(c), we indicate that with high transmission speed, the local MPC overlaps global MPC algorithm, because the global content popularity distribution is same as the local content popularity distribution without the influence from the request aggregation. However, when the transmission speed is relatively low, their performances become distinguished, since their content popularity distribution are changed by request aggregation mechanism.

In Fig. 7, 8 and 9, the weighting factors are set as $\omega_{\mathrm{T}} = 0.2$, $\omega_{\mathrm{D}} = 0.8$, which means that the content retrieval delay cost dominates the proposed content retrieval cost. That is the reason why the proposed caching algorithm outperforms all other algorithms with respect to content retrieval delay as shown in Fig. 7(c), 8(c) and 9(c). Meanwhile, because of the network heterogeneity, the caching strategy that achieves the lowest content retrieval delay usually does not achieve the lowest traffic load at the same time. Therefore, the proposed Q-learning algorithm performs worse than the benchmarks sometimes, as shown in Fig. 7(b), 8(b) and 9(b).

The importance of the traffic load and the content retrieval delay in the total content retrieval cost is tuned by the weighting factors, i.t. ω_{T} and ω_{D}. Fig. 10 represents the content retrieval cost, traffic load, and content retrieval delay influenced by variant weighting factors, respectively. It is seen from Fig. 10(a) that the proposed Q-learning algorithm achieves the lowest content retrieval cost among all these algorithms. Except the proposed algorithm, all the caching performance of the benchmark algorithms drops linearly with increasing the weighing factor of the traffic load. Jointly analyzing Fig. 10(b) and Fig. 10(c), the benchmark algorithms obtain constant traffic load and content retrieval delay, because their caching decisions are not affected by the weighting factors at all. However, the proposed Q-learning algorithm has the capability to track the optimization objective and adjust its caching algorithm according to it. The simulation contains two extreme situations. When $\omega_{\mathrm{T}} = 1$ and $\omega_{\mathrm{D}} = 0$, the proposed algorithm derives the minimum traffic load, which is 12% as shown in Fig. 10(b).



(a) Content retrieval cost (b) Traffic load (c) Content retrieval delay

FIGURE 10. **Network performance comparison with varying weighting factors.**

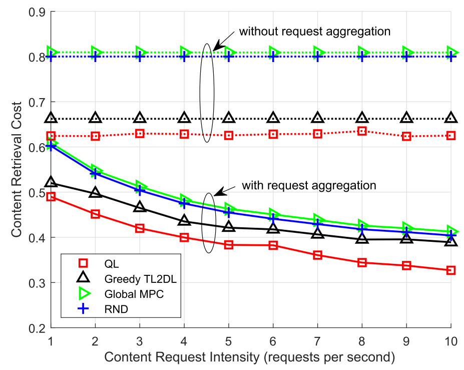


FIGURE 11. **Content retrieval cost comparison between with and without request aggregation mechanism.**

In contrast, when $\omega_T = 0$ and $\omega_D = 1$, the minimum content retrieval delay is obtained, which is 37% as illustrated in Fig. 10(c). Apparently, the proposed algorithm outperforms in network traffic offloading than content retrieval delay reduction. The reason is that the aggregated content requests do not generate traffic load to the upstream network just like the caching hit. However, the aggregated content requests can not be satisfied immediately as the caching hit content requests. They have to wait the previous required contents to be transmitted back from is content providers. Thus, the content retrieval delay of the aggregated requests can not be totally removed. Instead, the delay is only partly reduced.

Fig. 11 represents an intuitive simulation result to compare the system performance with and without request aggregation mechanism, with varying content request intensity. The content retrieval cost without request aggregation remains constant with varying content request intensity. It is because, in this scenario, every caching missed content request is forwarded to upstream networks. Given the traffic load and transmission delay for transmitting every content remains stable, increasing content request intensity only raises the total amount of the content requests, while the content retrieval cost per each content request keeps unchanged. On the contrary, when adopting request aggregation mechanism, the performance significantly declines with the increasing content request intensity, taking advantage of the request aggregation mechanism. The simulation results represent that the performance with request aggregation mechanism is always better than the one without request aggregation. When the content request intensity is small, request aggregation occurs occasionally, therefore the request aggregation gain is small consequently. Meanwhile, when the content request intensity is getting larger, more content requests are aggregated. Therefore, less network traffic load and content retrieval delay are consumed to satisfy each content request. The simulation result confirms that taking into request aggregation mechanism is able to reduce the content retrieval cost. Moreover, the content retrieval cost with request aggregation mechanism outperforms the one without request aggregation especially in heavy traffic networks.

VII. CONCLUSION

In this paper, we proposed a proactive edge caching algorithm to handle the massive and highly dynamic content requests to ENs. We adopted CCN architecture at the edge caching, not only to exploit the in-network content caching, but also to utilize its inherent request aggregation mechanism to further reduce the amount of content request. The more accurate cost function was explored to represent the performance gain for CCN architecture considering the request aggregation mechanism. The request aggregation mechanism dramatically reduced the content retrieval cost in the scenario of massive content requests. It is capable of providing a solution to supplement the content caching performance gain which is limited by caching space restriction. The Q-learning caching algorithm was developed to derive the optimal caching strategy in the scenario of highly dynamic content requests. Simulation results represented that implementing CCN architecture at ENs can dramatically reduce the content retrieval cost in the scenario of massive content request. Our research also demonstrated that the proposed Q-learning based caching algorithm obtained the minimized content retrieval cost among the Greedy, MPC and RND algorithms.

REFERENCES

[1] N. Hassan, K.-L.-A. Yau, and C. Wu, "Edge computing in 5G: A review," *IEEE Access*, vol. 7, pp. 127276–127289, 2019.

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[3] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1473–1499, 3rd Quart., 2015.

[4] T. Zhang, X. Fang, Y. Liu, and A. Nallanathan, "Content-centric mobile edge caching," *IEEE Access*, vol. 8, pp. 11722–11731, 2020.

[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[6] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: A survey," *Comput. Sci. Rev.*, vol. 19, pp. 15–55, Feb. 2016.

[7] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in information centric networking," *J. Netw. Comput. Appl.*, vol. 56, pp. 48–59, Oct. 2015.

[8] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in information-centric networking: Strategies, challenges, and future research directions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1443–1474, 2nd Quart., 2018.

[9] A. Kabir, G. Rehman, S. M. Gilani, E. J. Kitindi, Z. Ul Abidin Jaffri, and K. M. Abbasi, "The role of caching in next generation cellular networks: A survey and research outlook," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, p. e3702, Feb. 2020.

[10] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2847–2886, 4th Quart., 2016.

[11] S. Shan, C. Feng, T. Zhang, and J. Loo, "Proactive caching placement for arbitrary topology with multi-hop forwarding in ICN," *IEEE Access*, vol. 7, pp. 149117–149131, 2019.

[12] C. Fang, F. Richard Yu, T. Huang, J. Liu, and Y. Liu, "An energy-efficient distributed in-network caching scheme for green content-centric networks," *Comput. Netw.*, vol. 78, pp. 119–129, Feb. 2015.

[13] B. Nour, K. Sharif, F. Li, H. Moungla, A. E. Kamal, and H. Afifi, "NCP: A near ICN cache placement scheme for IoT-based traffic class," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Dec. 2018, pp. 1–6.

[14] N. Abani, T. Braun, and M. Gerla, "Proactive caching with mobility prediction under uncertainty in information-centric networks," in *Proc. 4th ACM Conf. Inf.-Centric Netw.*, Berlin, Germany, 2017, pp. 88–97.

[15] G. Mauri, M. Gerla, F. Bruno, M. Cesana, and G. Verticale, "Optimal content prefetching in NDN Vehicle-to-Infrastructure scenario," *IEEE Trans. Veh. Technol.*, vol. 66, no. 3, pp. 2513–2525, Mar. 2017.

[16] H. Farahat and H. Hassanein, "Optimal caching for producer mobility support in named data networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[17] Y. Zhang, X. Tan, and W. Li, "PPC: Popularity prediction caching in ICN," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 5–8, Jan. 2018.

[18] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, "Content popularity prediction and caching for ICN: A deep learning approach with SDN," *IEEE Access*, vol. 6, pp. 5075–5089, 2018.

[19] X. He, K. Wang, and W. Xu, "QoE-driven content-centric caching with deep reinforcement learning in edge-enabled IoT," *IEEE Comput. Intell. Mag.*, vol. 14, no. 4, pp. 12–20, Nov. 2019.

[20] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *Proc. 2nd Int. Conf. Information-Centric Netw. (ICN)*, New York, NY, USA, 2015, pp. 69–78.

[21] H. Dai, B. Liu, H. Yuan, P. Crowley, and J. Lu, "Analysis of tandem PIT and CS with non-zero download delay," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[22] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo, and M. Dianati, "Popularity-based video caching techniques for cache-enabled networks: A survey," *IEEE Access*, vol. 7, pp. 27699–27719, 2019.

[23] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.

[24] K. N. Doan, T. Van Nguyen, T. Q. S. Quek, and H. Shin, "Content-aware proactive caching for backhaul offloading in cellular network," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3128–3140, May 2018.

[25] S. O. Somuyiwa, A. Gyorgy, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.

[26] T. Zhang, X. Xu, L. Zhou, X. Jiang, and J. Loo, "Cache space efficient caching scheme for content-centric mobile ad hoc networks," *IEEE Syst. J.*, vol. 13, no. 1, pp. 530–541, Mar. 2019.

[27] G. Panwar, R. Tourani, S. Misra, and A. Mtibaa, "Request aggregation: The good, the bad, and the ugly," in *Proc. 4th ACM Conf. Inf.-Centric Netw.*, New York, NY, USA, 2017, pp. 198–199.

[28] G. Carofiglio, M. Gallo, and L. Muscariello, "On the performance of bandwidth and storage sharing in information-centric networks," *Comput. Netw.*, vol. 57, no. 17, pp. 3743–3758, Dec. 2013.

[29] T. Huang, G. Wang, C. Fang, F. R. Yu, and Y. Liu, "Modeling of miss-probability in content-centric networking," *Sci. China Inf. Sci.*, vol. 58, no. 7, pp. 1–13, Jul. 2015.

[30] A. J. Abu, B. Bensaou, and J. M. Wang, "Interest packets retransmission in lossy CCN networks and its impact on network performance," in *Proc. 1st Int. Conf. Inf.-Centric Netw. (INC)*, New York, NY, USA, 2014, pp. 167–176.

[31] X. Xu, C. Feng, T. Zhang, J. Loo, and G. Y. Li, "Caching performance of information centric networking with content request aggregation," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[32] M. Ahmadi, J. Roberts, E. Leonardi, and A. Movaghar, "Impact of traffic characteristics on request aggregation in an NDN router," 2019, *arXiv:1903.06419*. [Online]. Available: http://arxiv.org/abs/1903.06419

[33] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[34] M. Amadeo, C. Campolo, A. Molinaro, and G. Ruggeri, "Content-centric wireless networking: A survey," *Comput. Netw.*, vol. 72, pp. 1–13, Oct. 2014.

[35] J. Jiang, S. Zhang, B. Li, and B. Li, "Maximized cellular traffic offloading via Device-to-Device content sharing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 82–91, Jan. 2016.

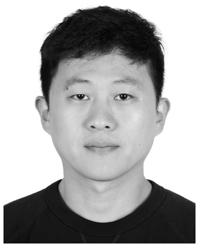
[36] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.

[37] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[38] F. S. Melo, "Convergence of Q-learning: A simple proof," Inst. Syst. Robot., Lisbon, Portugal, Tech. Rep. 1, 2001, pp. 1–4.

XIAOGENG XU (Student Member, IEEE) received the B.S. degree in communications engineering from the Beijing University of Technology, China, in 2005, the M.S. degree in digital communication systems and technology from the Chalmers University of Technology, Sweden, in 2007. She is currently pursuing the Ph.D. degree in communication and information systems from the Beijing University of Posts and Telecommunications. Her current researches focus on caching performance analysis and optimization in content-centric networks.

CHUNYAN FENG (Senior Member, IEEE) received the B.S. degree in communications engineering and the M.S. and Ph.D. degrees in communication and information systems from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China. She is currently a Professor with the School of Information and Communication Engineering, BUPT. Her research interests are in the areas of broadband networks and wireless communication systems. Her current research focuses on cognitive radio and green wireless communications.



SIYANG SHAN (Student Member, IEEE) received the B.S. degree in communications engineering from the Beijing University of Posts and Telecommunications, China, in 2011, the M.S. degree from Hainan University, China, in 2014. He is currently pursuing the Ph.D. degree in communication and information systems from the Beijing University of Posts and Telecommunications. His current research focuses on caching and routing in future Internet architecture.

TIANKUI ZHANG (Senior Member, IEEE) received the Ph.D. degree in information and communication engineering and the B.S. degree in communication engineering from the Beijing University of Posts and Telecommunications (BUPT), China, in 2008 and 2003, respectively. He is currently a Professor with the School of Information and Communication Engineering, BUPT. His research interests include wireless communication networks, mobile edge computing and caching, signal processing for wireless communications, content centric wireless networks. He has been an Associate Editor for IEEE ACCESS, since 2020. He had published more than 100 articles including journal articles on the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE TRANSACTION ON COMMUNICATIONS, and conference papers, such as the IEEE GLOBAL COMMUNICATIONS CONFERENCE (GLOBECOM) and the IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC).

JONATHAN LOO (A.K.A. KOK KEONG LOO) received the M.Sc. degree (Hons.) in electronics and the Ph.D. degree in electronics and communications from the University of Hertfordshire, Hertfordshire, U.K., in 1998 and 2003, respectively. In 2003 and 2010, he was a Lecturer in multimedia communications with the School of Engineering and Design, Brunel University, Uxbridge, U.K. In June 2010 and May 2017, he was an Associate Professor in communication networks with the School of Science and Technology, Middlesex University, London, U.K. From June 2017, he was a Chair Professor in computing and communication engineering with the School of Computing and Engineering, University of West London, U. K. He has successfully graduated 18 Ph.D. students as their Principal Supervisor, and has coauthored over 350 journal and conference papers in the aforementioned specialized areas. His research interests include natural language and image processing, machine learning and AI, the IoT/cyber-physical systems, cyber-security, information centric networking, communication networks, and wireless/mobile systems. He has been an Associate Editor of the Wiley *International Journal of Communication Systems*, since 2011. He was the Lead Editor of the book *Mobile Ad Hoc Networks: Current Status and Future Trends* (CRC Press, 2011).