Distributed Online Learning of Cooperative Caching in Edge Cloud

Xinchen Lyu, Chenshan Ren, Wei Ni, Hui Tian, Ren Ping Liu, and Xiaofeng Tao

Abstract—Cooperative caching can unify storage across edge clouds and provide efficient delivery of popular contents under effective content placement. However, the placement and delivery are non-trivial in cooperative caching due to the decentralized property of edge clouds, as well as the temporal and spatial correlation of the placement. We propose a new distributed online learning approach to jointly optimize content placement and delivery without the a-priori knowledge on file popularity and link availability. Content placement and delivery can be asymptotically optimized in real-time by running distributed online learning at individual edge servers by exploiting stochastic gradient descent (SGD). The proposed approach can allow operations at different timescales by integrating mini-batch learning for farsighted content placement. The optimality loss, stemming from the different timescales, can asymptotically reduce, as the SGD stepsize declines. Simulations confirm that the proposed approach outperforms existing techniques in terms of cache hit ratio and cost effectiveness. Insights are shed on the optimal placement of popular contents.

Index Terms—Cooperative caching, distributed online learning, stochastic gradient descent, mini-batch learning

1 Introduction

The upgrades of network infrastructure are outpaced by the increasingly great demand for data traffic and bandwidth-hungry services, challenging network performance and viability [1]. Smart caching can prefetch popular contents to edge cloud during off-peak times, deliver the contents directly from the network edge, alleviate congestions of backbone network during peak hours, and reduce the costs of file delivery [2], [3]. The edge cloud refers to the network of edge servers spread in cities, e.g., alongside with base stations (BSs) and including the BSs. The memory sizes of the edge servers are comparatively limited (with regards to the large number of contents to be cached). Cooperative edge caching is important for the performance of the system by unifying storage and memories across edge clouds to overcome restrictions of limited storage per server [4], [5].

Cooperative edge caching is not trivial. The spatial and temporal variations of random content request arrivals and background traffic would hinder the acquisition of the a-priori knowledge on file popularity and link availability in practice. Nevertheless, given the large size of edge clouds, centralized coordination is not effective and would undergo high signaling delays and outdated network knowledge [6]. All this prevents the development of optimal online distributed solutions for cooperative smart caching.

- *This work was supported by the National Key R&D Program of China under Grant No. 2018YFE0205502. (Corresponding Author: Hui Tian.)*

- *X. Lyu and X. Tao are with the National Engineering Laboratory for Mobile Network Technologies, Beijing University of Posts and Telecommunications, China.*
- *C. Ren is with Minzu University of China, China.*
- *H. Tian is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China.*
- *W. Ni is with the Digital Productivity and Service (DP&S) Flagship, CSIRO, Australia.*
- *R. P. Liu is with the Global Big Data Technologies Center, University of Technology Sydney, Australia.*

Moreover, in cooperative smart caching, content placement and delivery are required to operate at different timescales. This is because content placement can cause non-negligible delays and cost for an edge server to retrieve the file from the backbone and replace contents in its local memories. It is practically important to avoid frequent cache replacement in cooperative caching and hence frequent disruptions, resulting from the update of network knowledge for request dispatching. The distributed joint optimization of request dispatch, content delivery, and content placement at different timescales has yet to be addressed in the literature [7]–[26].

In this paper, we propose a novel fully distributed online learning approach of cooperative smart caching without the a-priori knowledge on file popularity and link availability. The content placement and delivery (i.e., the request dispatch, content delivery, and content placement) are jointly optimized at different timescales. The key idea is that we first minimize the time-average cost across infinite time horizons, provided the content placement and delivery can run at the same timescale. By exploiting stochastic gradient descent (SGD), the asymptotically optimal placement and delivery can be decoupled between edge servers and across time slots, and achieved by distributed online learning implemented at individual edge servers based on their observations on neighbors.

Another important aspect is that we further diverge the proposed distributed online learning with different timescales for content placement and delivery. This is done by integrating mini-batch learning into the distributed online learning to achieve farsighted content placement. The optimality loss, stemming from the different timescales, is proved to asymptotically diminish, as the stepsize of SGD decreases. The farsighted content placement is important to mitigate the disruptions that frequent content placement would cause to content delivery.

The major contributions of this work include:

- We propose an asymptotically optimal, distributed, online learning approach to jointly optimize the content placement, request dispatch, and content delivery across the edge cloud. In contrast, the existing techniques restricted any cooperative content delivery within only up to two hops [13]–[20], [24]–[27], assumed instantaneous global view [28], or failed to guarantee the (asymptotic) optimality [29].
- We integrate mini-batch learning in the proposed online learning framework to enable farsighted content placement at a large timescale. The mini-batch learning is important to avoid excessively frequent replacement of cached files and, subsequently, significant growth of the cache replacement cost (which is typical in the existing works [24]–[26] that optimized content delivery and replacement at the same timescale).
- We prove that the proposed distributed online learning approach (with the mini-batch learning for farsighted content placement) is asymptotically optimal. Such optimality has not been established in the literature, except for centralized learning [24]–[26] or for content delivery only at the short timescale [29].

Extensive simulations demonstrate the asymptotic optimality and the gains of our technique in cache hit ratio and cost effectiveness (i.e., the caching profit against backbone delivery) over existing techniques. Interesting insights are also shed that the probability of the optimal content placement matches the file popularity under different popularity distributions in the case where each edge server can cache a file. Mismatches can arise when local cache can store multiple files, since cooperative caching can leverage the memories across an edge cloud to efficiently store popular files without sacrificing the cache hit ratio of the files.

Ideally, the centralized approach could have a global network view to improve prediction accuracy. However, the global view can hardly be timely (or instantaneous) as the network size scales under the non-negligible multi-hop signaling delays [6]. Due to the outdated global view, the performance of the centralized approach would be compromised in practice. The proposed distributed online learning approach can operate without the global network view, but still asymptotically achieve the prediction accuracy which the centralized learning would achieve under the ideal setting with the instantaneous global view.

The remainder of this paper is arranged as follows. Existing studies are summarized in Section 2, and the system model is presented in Section 3. The new distributed online learning approach is articulated in Section 4. In Section 5, the mini-batch learning is integrated into the proposed distributed online learning to allow operations at different timescales for content placement and delivery. Section 6 demonstrates simulation results, followed by conclusions in Section 7.

2 Related Work

2.1 Content Placement and Delivery in Edge Caching

By assuming that the decisions of content placement are known in prior, the delivery of contents has been studied in wireless and wired networks [7]–[12]. In [7]–[9], the local memories of the access points were exploited to increase the throughput of wireless system with multicast and cooperative beamforming. In [10]–[12], cache-aware routing was designed for minimizing the cost of content delivery through the network backbone. These approaches were based on offline, centralized optimizations, and unsuitable for cooperative smart caching in edge cloud, given the sheer network scale and multi-hop signaling delays.

In [13]–[20], cooperative content placement was studied under the assumption of perfect knowledge on file popularity and link availability. In [13]–[16], considering the typical hierarchical structure of edge networks, the in-tier and cross-tier content placement was solved by game theory [15], integer programming [13], [14], or analyzing the content popularity [16]. In [17], caching, routing, and channel allocation were jointly optimized to maximize the sum rate of a multi-cell system by exploiting a column generation method. In [18], [19], layered video caching and transcoding were jointly optimized by using offline integer programming techniques in a centralized manner. In [20], the problem of jointly optimizing content routing and placement was NP-complete, and a heuristic algorithm was proposed to reach $(1-1/e)$ of the optimum. However, the perfect network knowledge for offline optimization in [13]–[20] is not available given the stochasticity of request arrivals and link availability.

In light of Gibbs sampling, a sequential content update strategy was proposed in [27] to minimize the cost of downloading files from backhaul in densely deployed cellular networks, where the BSs have overlapping coverage areas and the user can select one of its nearby BSs to retrieve its requested file. The strategy was proved to be asymptotically optimal in both the presence and absence of the knowledge on file popularity. The scenario in [27] is substantially different from the multi-hop cooperative edge caching studied in this paper where multi-hop request dispatch and content delivery require distinct and non-trivial designs. In [28], a real-time caching algorithm was proposed to minimize the overall cost of content providers in a cooperative multi-cell network, where the requested files at a BS can be retrieved from its local memories, the other BSs, or the network backbone. The problem was cast as integer linear programming with NP-completeness. However, the algorithm in [28] would require the instantaneous global view of the network at a centralized controller, and would hardly be scalable in large-scale networks where the instantaneous global view cannot be available.

2.2 Applications of SGD in Edge Caching

As the generalization of the celebrated Lyapunov optimization, SGD can decouple a stochastic time-varying system across different time slots and achieve asymptotic optimality. SGD can be applied to edge intelligence [30], fog computing [6], [31]–[34], and software-defined networking [35].

SGD (or Lyapunov optimization) has also been applied for the content placement in edge caching under the multi-user single-cell network [22], [23] or hierarchical networks [24]–[26]. In [22] and [23], video caching, transcoding and wireless transmission were jointly optimized for the profit

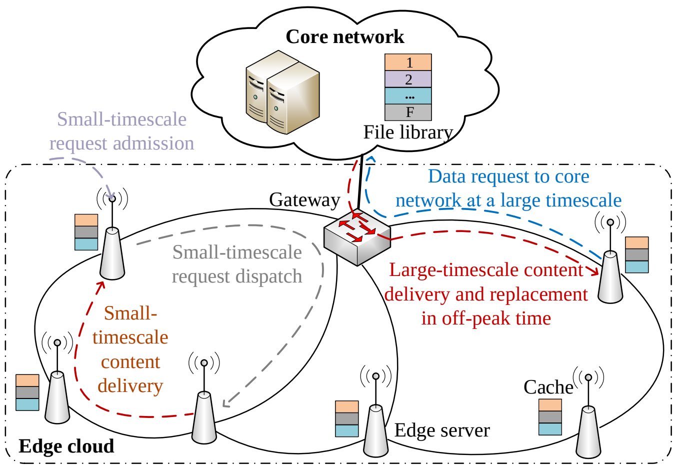

Core network
1
2
...
F
File library
Small-timescale request admission
Gateway
Data request to core network at a large timescale
Small-timescale request dispatch
Large-timescale content delivery and replacement in off-peak time
Small-timescale content delivery
Cache
Edge server
Edge cloud

Fig. 1. An illustration of the proposed cooperative edge caching [29]. The edge servers retrieve data from the backbone (core network) for content replacement during off-peak time (at a large timescale), and the cooperative data delivery between the servers happens from time to time (at a short timescale).

TABLE 1
Definitions of Notations

Notation	Definition
$\mathbf{F}$	Set of files
$\mathbf{N}$	Set of edge servers
$\mathbf{E}$	Set of inter-server links
C_{ij}	Capacity of link (i, j)
ζ_{ij}	Unit cost of content delivery over link (i, j)
$Q_{i,f}^{s}$	Request queue of server i for file f from server i
D_{i}^{s}	Data queue of server i destined for server s
$R_{i,f}$	Size of requests for file f at server i
$r_{i,f}$	Admission of requests for file f at server i
$\zeta_{i,f}$	Unit cost for server i of retrieving file f from backbone
$q_{ij,f}^{s}$	Dispatch of requests from server s from servers i to j
d_{ij}^{s}	Delivery of the file destined for server s via link (i, j)
$b_{i,f}^{s}$	Grant of requests from server s for file f at server i
F_i	Maximal request grant of server i per slot
$c_{i,f}$	Caching decision of file f for server i
ϵ	Stepsize of SGD
T	The interval of content refreshment in Section 5

of video caching in a single-cell network. By exploiting the computing resources of edge servers, the high-quality video segments are online transcoded to the user-requested quality to save the storage of caching multi-resolution copies of the same video [23]. In [24] and [25], considering a two-tier network, cache rental and placement were jointly optimized to achieve the maximum profit of the content provider. In [26], the joint optimization of content placement and delivery was proposed for maximizing the throughput of a three-tier network. These approaches considered only a single server [22], [23] or limited content delivery within two hops [24]–[26], and cannot scale to the cooperative smart caching with multi-hop content delivery in this paper.

In our recent work [29], a distributed online learning technique was proposed to optimize only the request dispatch and content delivery, and establish profitable caching regions in the edge cloud. Only a single timescale was considered. Cooperations could only take place within the profitable caching regions. Despite that the profitable caching regions could operate in conjunction with the popular Least Frequently Used (LFU) or Least Recently Used (LRU) content replacement strategies, the algorithm developed in [29] did not guarantee the optimality. In contrast, the new approach proposed in this paper jointly optimizes the content placement together with the request dispatch and content delivery across the edge cloud. The new approach integrates mini-batch learning at a large timescale to plan content placement farsightedly and avoid excessively frequent replacement of cached files. Moreover, the proposed approach is rigorously proved to be asymptotically optimal.

3 System Model

Fig. 1 shows the smart edge caching network of N number of edge servers with data memories. Let $\mathbf{N} = \{1, \cdots, N\}$ collect the indices to the N edge servers, and $\mathbf{F} = \{1, \cdots, F\}$ collect the files. Without loss of generality, the files are pre-partitioned (e.g., into blocks in real systems) to have unit size. Each edge server caches some of the files based on the size of its local memories. The requests for a file received by a server is either delivered/satisfied from the backbone or the other edge servers which store the file in its local memory. The network operates in a slotted structure. The duration of a time slot can range from 1ms (e.g., the length of TTI in LTE) to tens of milliseconds. Table 1 summarizes the notations in this paper, where $\cdot(t)$ to specify time slot t of the notation is suppressed for brevity.

The edge servers are connected through shared or dedicated network links. Let $\mathbf{G} = \{\mathbf{N}, \mathbf{E}\}$ be the network topology of the edge cloud, where $\mathbf{N}$ and $\mathbf{E}$ are the sets of edge servers and inter-server links in the network, respectively. For example, the inter-server links can be X2 interfaces (between BSs) in 4G networks. The capacity of the network links ranges typically from 1 Gbps (e.g., Ethernet category 6 cable) to 10 Gbps (e.g., optical fiber), and can support cooperative data delivery between the edge servers. The capacity of the links does not change during the slot, and can only vary between different slots given the random background traffic. $C_{ij}(t) \leq C_{ij}^{\max}$ is the link capacity for bi-directional data transmissions over link (i, j) at slot t. At time slot t, $\zeta_{i,f}$ and $\zeta_{ij}(t)$ are the costs of delivering unit size of data (for file f) from the network backbone to server i and over link (i, j), respectively.

$R_{i,f}(t)$ denotes the amount of requests for the f-th file captured by server i. It specifies the amount of data to be delivered to the requested user at slot t. Without the a-priori knowledge on the file popularity[1], $R_{i,f}(t)$ is assumed to be a stochastic process satisfying $R_{i,f}(t) \leq R_{i,f}^{\max}$. Let $r_{i,f}(t)$ be the size of admitted requests for the f-th file at server i

1. The content popularity can vary drastically between the edge servers. The a-priori knowledge of the content popularity at individual edge servers may not be available in practice. This is because the content popularity needs to be evaluated based on a large number of (or a set of pre-known) users and their requests. For an edge server (e.g., a BS), the number of users in its coverage is typically not big enough (e.g., up to a few hundred) to provide an unbiased estimation of the content popularity. Moreover, the popularity can even change over time. Learning online the variation of content popularity among the edge servers and the changes over time is expected to be effective, as compared to offline (or posteriori) measurement.

during slot t to be retrieved from other edge servers within the edge cloud. The remainder of the request, $\big(R_{i,f}(t) - r_{i,f}(t)\big)$, is delivered from the network backbone. We have

$$0 \leq r_{i,f}(t) \leq R_{i,f}(t), \forall i, f, t. \tag{1}$$

At each edge server, NF queues are used to store the requests for the F files received by N edge servers, and another N queues are required for the files to be delivered to the users in the coverage of N edge servers. To facilitate the retrieval of F files to N servers requires the placement of $N(F+1)$ queues at each edge server [6], [31], [32].

Let $Q^s_{i,f}(t)$ and $D^s_i(t)$ be the queue backlogs at server i at slot t of the requests from server s for file f and the files destined for server s, respectively. $\mathbf{Q}(t) = \{Q^s_{i,f}(t), D^s_i(t), \forall i, f, s\}$. The following constraints need to be satisfied:

$$\sum\nolimits_{s\in\mathbf{N}}[d^s_{ij}(t) + d^s_{ji}(t)] \leq C_{ij}(t), \forall(i,j), t; \tag{2a}$$
$$\sum\nolimits_{s\in\mathbf{N}, f\in\mathbf{F}}[q^s_{ij,f}(t) + q^s_{ji,f}(t)] \leq C_{ij}(t), \forall(i,j), t; \tag{2b}$$
$$q^s_{ij,f}(t) \geq 0, d^s_{ij}(t) \geq 0, \forall i, j, f, s, t. \tag{2c}$$

Here, $q^s_{ij,f}(t)$ and $d^s_{ij}(t)$ are the sizes of the requests for file f to be dispatched and the files to be delivered, respectively, from servers i to j during slot t. The dispatch of requests between edge servers in (2b) only involves the transmission of a short packet[2], which is negligible compared to the delivery of contents/files of up to Gigabytes in (2a). As a result, constraint (2a) captures the capacity (or data rate) of link (i, j), and constraint (2b) is to avoid the dispatch of too many requests and stop the requested files from being returned to congest link (i, j).

Let $b^s_{i,f}(t)$ be the size of requests for file f from server s and served at edge server i during the t-th slot, and $F_i(t) \leq F^{\max}$ denote the maximum of request grant during slot t at the server. $F_i(t)$ is time-varying given the random background services at the edge servers. $c_{i,f}(t) \in \{0, 1\}$ is the content placement decisions of file f at edge server i at slot t. Server i caches the f-th file in its local memory at slot t, if $c_{i,f}(t) = 1$. The content placement and request granting need to satisfy

$$\sum\nolimits_{f\in\mathbf{F}, s\in\mathbf{N}} b^s_{i,f}(t) \leq F_i(t), \forall i, t; \tag{3a}$$
$$0 \leq b^s_{i,f}(t) \leq c_{i,f}(t)F_i(t), \forall i, f, s, t, \tag{3b}$$
$$c_{i,f}(t) \in \{0, 1\}, \forall i, f, t, \tag{3c}$$
$$\sum\nolimits_{f\in\mathbf{F}} c_{i,f}(t) \leq M_i, \forall i, t, \tag{3d}$$

where (3b) indicates that server i serves (or satisfies) the requests for the files in its local memory. Constraint (3d) specifies that each edge server only has a finite memory and edge server i can cache up to M_i files in its local memory.

Note that the content placement needs to be carried out at much longer intervals than the request dispatch and content delivery. This is because content placement can cause non-negligible delays and cost for an edge server to retrieve the file from the backbone and replace the contents in its local memories. It is of practical importance to avoid frequent cache replacement in cooperative caching, and hence the substantial increase of the costs for collecting files from the backbone and replacing the local memory.

2. For example, a 4-byte unsigned integer (e.g., the uint32 data type in C++ that can represent 2^{32} integers within [0,4294967295]) is sufficient to represent the queue backlogs of up to 4.3GB.

The backlog of request queue $Q^s_{i,f}(t)$ is updated by

$$\begin{aligned} Q^s_{i,f}(t+1) = [Q^s_{i,f}(t) - \sum\nolimits_{j\in\mathbf{N}} q^s_{ij,f}(t) - b^s_{i,f}(t)]^+ \\ + \sum\nolimits_{j\in\mathbf{N}} q^s_{ji,f}(t) + r^s_{i,f}(t). \end{aligned} \tag{4}$$

where $[\cdot]^+ = \max\{\cdot, 0\}$, $r^i_{i,f}(t) = r_{i,f}(t)$, and $r^s_{i,f}(t) = 0, \forall s \neq i$. Here, $[Q^s_{i,f}(t) - \sum_{j\in\mathbf{N}} q^s_{ij,f}(t) - b^s_{i,f}(t)]^+$ is the size of requests in $Q^s_{i,f}$ by slot t, after granting and dispatching part of the requests at server i. The second and third terms are the arrivals of requests from the users or dispatched from other servers.

The backlog of data queue $D^s_i(t)$ is updated by

$$\begin{aligned} D^s_i(t+1) = [D^s_i(t) - \sum\nolimits_{j\in\mathbf{N}} d^s_{ij}(t)]^+ \\ + \sum\nolimits_{j\in\mathbf{N}} d^s_{ji}(t) + \sum\nolimits_{f\in\mathbf{F}} b^s_{i,f}(t), \forall s \neq i, \end{aligned} \tag{5}$$

where $D^i_i(t) = 0$, as server i acts as the sink of the files.

4 Distributed Online Learning of Cooperative Smart Caching

This section presents the proposed distributed online learning to asymptotically optimize content placement, request dispatch, and content delivery, provided that the content placement can be implemented per time slot. As discussed in Section 3, the content placement needs to be implemented at large time intervals to avoid frequent cache replacement and hence the substantially increasing cost of cache replacement. Section 5 will show that the proposed approach can be integrated with mini-batch learning to facilitate farsighted content placement without compromising the optimality.

4.1 Problem Statement and Reformulation

Cost is taken as the generic measure of smart edge caching. Let $\Psi(\mathbf{x}^t)$ denote the cost in edge caching, as given by

$$\Psi(\mathbf{x}^t) = \sum_{i,j\in\mathbf{N}} \psi_{ij}(t) + \sum_{i\in\mathbf{N}, f\in\mathbf{F}} \zeta_{i,f}(R_{i,f}(t) - r_{i,f}(t)), \tag{6}$$

where $\mathbf{x}^t = \{c_{i,f}(t), b^s_{i,f}(t), q^s_{ij,f}(t), d^s_{ij}(t), r_{i,f}(t), \forall i, j, f, s\}$ collects the variables of slot t. Here, $\zeta_{i,f}(R_{i,f}(t) - r_{i,f}(t))$ and $\psi_{ij}(t) = \zeta_{ij}(t)\sum_{s\in\mathbf{N}} d^s_{ij}(t)$ are the costs of retrieving file f from the network backbone to server i and delivering files from servers i to j at slot t, respectively.

The stability of the system is written as

$$\overline{Q^s_{i,f}(t)} < \infty, \overline{D^s_i(t)} < \infty, \forall i, f, s, \tag{7}$$

where $\overline{X(t)} = \lim_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1} \mathbb{E}[X(\tau)]$ is the time-average of a random process $X(t)$.

We minimize the time-average cost of smart caching, i.e., $\overline{\Psi(\mathbf{x}^t)}$, under the system stability constraint (7) without the priori knowledge of request arrivals and link availability. Let $\mathbb{X} = \{\mathbf{x}^t, \forall t\}$ be the set of all the variables of content delivery, request dispatch, and content placement, across all time slots. The problem can be given by

$$\begin{aligned} \Psi^* = \min_{\mathbb{X}} \overline{\Psi(\mathbf{x}^t)} \\ \text{s.t. (1) – (5), (7), } \forall t. \end{aligned} \tag{8}$$

According to queuing theory [36], the system stability constraint (7) specifies that the time-average input rates of the queues cannot exceed the corresponding time-average output rates of the queues. (7) can be rewritten as

$$\overline{\sum\nolimits_{j\in\mathbf{N}}(q^s_{ji,f}(t)-q^s_{ij,f}(t))+r^s_{i,f}(t)-b^s_{i,f}(t)}\leq 0,\forall i,f,s;$$
$$\overline{\sum\nolimits_{j\in\mathbf{N}}(d^s_{ji}(t)-d^s_{ij}(t))+\sum\nolimits_{f\in\mathbf{F}}b^s_{i,f}(t)}\leq 0,\forall i,s. \tag{9}$$

Problem (8) can be reformulated as

$$\Psi^*=\min_{\mathbb{X}}\overline{\Psi(\mathbf{x}^t)} \quad \text{s.t. (1) – (3), (9), } \forall t. \tag{10}$$

Given the time-average objective and constraint, i.e., $\overline{\Psi(\mathbf{x}^t)}$ and (9), the offline optimum to (10) requires the knowledge of the network dynamics of request arrivals and link availability over an infinite time horizon, which is not available in practice. We proceed to exploit SGD of machine learning to learn the optimal solution online from the observed network dynamics at each edge server over time. The details are provided in the following.

4.2 SGD-based Online Learning

SGD is an established method in machine learning widely applied in logistic regression and neural networks [37]. SGD is effective where exact gradients are computationally expensive given the large size of datasets (or the prevalent randomness in this paper). As a result, stochastic gradient can be used to approximate the exact gradient based on a subset of sampled datasets (or the observed network dynamics at each time slot) over time.

By taking SGD, problem (10) can be decoupled between time slots, i.e., eliminate (9), and reformulated into solving (11) at each slot t:

$$\max_{\mathbf{x}^t}\alpha(\mathbf{r}^t)+\beta(\mathbf{q}^t)+\gamma(\mathbf{d}^t)+\eta(\mathbf{b}^t,\mathbf{c}^t) \quad \text{s.t. (1), (2), (3).} \tag{11}$$

The reformulation is achieved by associating (9) with Lagrange multipliers and applying SGD to update the multipliers at each time slot, with the details in Appendix A. In (11), $\mathbf{r}^t=\{r_{i,f}(t),\forall i,f\}$, $\mathbf{b}^t=\{b^s_{i,f}(t),\forall i,f,s\}$, $\mathbf{q}^t=\{q^s_{ij,f}(t),\forall i,j,f,s\}$, $\mathbf{d}^t=\{d^s_{ij}(t),\forall i,j,s\}$, and $\mathbf{c}^t=\{c_{i,f}(t),\forall i,f\}$ are the decisions of admission, request grant, request dispatch, content delivery, and content placement at slot t, respectively. We have

$$\alpha(\mathbf{r}^t)=\sum\nolimits_{i\in\mathbf{N},\mathbf{f}\in\mathbf{F}}[\zeta_{i,f}-\epsilon Q^i_{i,f}(t)]r_{i,f}(t); \tag{12a}$$
$$\beta(\mathbf{q}^t)=\epsilon\sum_{i,j,s\in\mathbf{N},\mathbf{f}\in\mathbf{F}}Q^s_{i,f}(t)(q^s_{ij,f}(t)-q^s_{ji,f}(t)); \tag{12b}$$
$$\gamma(\mathbf{d}^t)=\epsilon\sum_{i,j,s\in\mathbf{N}}D^s_i(t)(d^s_{ij}(t)-d^s_{ji}(t))-\zeta_{ij}(t)d^s_{ij}(t); \tag{12c}$$
$$\eta(\mathbf{b}^t,\mathbf{c}^t)=\epsilon\sum_{i,s\in\mathbf{N},\mathbf{f}\in\mathbf{F}}b^s_{i,f}(t)c_{i,f}(t)(Q^s_{i,f}(t)-D^s_i(t)); \tag{12d}$$

and ϵ is the stepsize of SGD. The Lagrange multipliers can be interpreted as the product of the stepsize and the queue backlogs given the stepsize ϵ; and are suppressed in (12); see Appendix A for details.

We can see in (11) that the optimization variables of $\mathbf{r}^t$, $\mathbf{b}^t$, $\mathbf{q}^t$, $\mathbf{d}^t$, and $\mathbf{c}^t$ are decoupled both in the objective and constraints. Moreover, the decisions on content placement $\mathbf{c}^t$ and request admission $\mathbf{r}^t$ can be decoupled between different edge servers, and the decisions of request dispatch $\mathbf{q}^t$ and content delivery $\mathbf{d}^t$ can be decoupled between links. Problem (11) can be decomposed to the problems of optimizing request admission, request dispatch, content delivery, as well as request grant and content placement, as respectively given by (13a), (13b), (13c) and (13d):

$$\max_{r_{i,f}(t)}\alpha_{i,f}(t)r_{i,f}(t), \text{ s.t. (1);} \tag{13a}$$
$$\max_{\tilde{\mathbf{r}}_{ij}(t)}\sum_{s\in\mathbf{N},f\in\mathbf{F}}\beta^s_{ij,f}(t)q^s_{ij,f}(t)+\beta^s_{ji,f}(t)q^s_{ji,f}(t), \text{ s.t. (2b), (2c);} \tag{13b}$$
$$\max_{\tilde{\mathbf{d}}_{ij}(t)}\sum_{s\in\mathbf{N}}\gamma^s_{ij}(t)d^s_{ij}(t)+\gamma^s_{ji}(t)d^s_{ji}(t), \text{ s.t. (2a), (2c);} \tag{13c}$$
$$\max_{\mathbf{b}_i(t),\mathbf{c}_i(t)}\sum_{f\in\mathbf{F}}\eta^s_{i,f}(t)b^s_{i,f}(t)c_{i,f}(t), \text{ s.t. (3);} \tag{13d}$$

where $\alpha_{i,f}(t)=\zeta_{i,f}-\epsilon Q^i_{i,f}(t)$, $\beta^s_{ij,f}(t)=\epsilon(Q^s_{i,f}(t)-Q^s_{j,f}(t))$, $\gamma^s_{ij}(t)=\epsilon[D^s_i(t)-D^s_j(t)]-\zeta_{ij}(t)$, and $\eta^s_{i,f}(t)=\epsilon(Q^s_{i,f}(t)-D^s_i(t))$. $\tilde{\mathbf{r}}_{ij}(t)=\{q^s_{ij,f}(t),q^s_{ji,f}(t),\forall f,s\}$ and $\tilde{\mathbf{d}}_{ij}(t)=\{d^s_{ij}(t),d^s_{ji}(t),\forall s\}$ collect the decisions of request dispatch and content delivery on link (i,j) at slot t, respectively.

Here, (13a) is linear programming of maximizing the weighted sum of the variables [38]. Its optimal solution is as follows.

$$r_{i,f}(t)=\begin{cases}R_{i,f}(t), \text{ if } Q^i_{i,f}(t)<\zeta_{i,f}/\epsilon;\\ 0, \text{ otherwise.}\end{cases} \tag{14}$$

Problems (13b) and (13c) are also linear programming, and their optimal solutions can be obtained by evaluating $\beta^s_{ij,f}(t)$, $\beta^s_{ji,f}(t)$, $\gamma^s_{ij}(t)$ and $\gamma^s_{ji}(t)$. In specific, for (13b), if $\max_{f,s}\{\beta^s_{ij,f}(t)\}<0$ or $\max_{f,s}\{\beta^s_{ij,f}(t)\}<\max_{f,s}\{\beta^s_{ji,f}(t)\}$, we have $q^s_{ij,f}(t)=0$. Otherwise, the optimal size of dispatched requests is written as

$$q^s_{ij,f}(t)=\begin{cases}C_{ij}(t), \text{ if } (f,s)=\arg\max_{f,s}\beta^s_{ij,f}(t);\\ 0, \text{ otherwise.}\end{cases} \tag{15a}$$

For (13c), if $\max_s\{\gamma^s_{ij}(t)\}<0$ or $\max_s\{\gamma^s_{ij}(t)\}<\max_s\{\gamma^s_{ji}(t)\}$, $d^s_{ij}(t)=0$. Otherwise, the optimal size of content delivery can be given by

$$d^s_{ij}(t)=\begin{cases}C_{ij}(t), \text{ if } s=\arg\max_s\gamma^s_{ij}(t);\\ 0, \text{ otherwise.}\end{cases} \tag{15b}$$

Lastly, (13d) is a mixed-integer linear program, and can be decomposed to two dependent subproblems: (a) optimizing request grant $\mathbf{b}_i(t)$ given the content placement; and (b) optimizing the content placement $\mathbf{c}_i(t)$ based on the optimal request grant of the former subproblem. Given content placement $c_{i,f}(t)$, (13d) is linear programming. If $\max_{f,s}\{\eta^s_{i,f}(t)c_{i,f}(t)\}\leq 0$, $b^s_{i,f}(t)=0$. Otherwise, the optimal solution is

$$b^s_{i,f}(t)=\begin{cases}F_i(t), \text{ if } (f,s)=\arg\max\limits_{f,s}\eta^s_{i,f}(t)c_{i,f}(t);\\ 0, \text{ otherwise;}\end{cases} \tag{16}$$

where only the request queue with the most urgency is selected in (16). As a result, edge server i decides to place file f with the most urgency at the slot, i.e.,

$$c_{i,f}(t) = 1, \text{ if } f = \arg\max_{f,s} \eta^s_{i,f}(t). \tag{17}$$

It is noted that (14) indicates a request is accepted or rejected in whole to preserve the integrity of a file. In (15) and (16), the files are implicitly assumed to be dividable. This is because $C_{ij}(t)$ and $F_i(t)$ may not accommodate integer numbers of files. In this case, (15) and (16) are rounded to the largest integer numbers of files. The rounding of (15) and (16) also preserves the asymptotic optimality of our approach, since the optimality gap (or the difference to the optimal solution in the case where the files can be dividable) is upper bounded by $C_{ij}(t)$ and $F_i(t)$ and can asymptotically diminish as $\epsilon \to 0$. Nevertheless, as will be shown shortly, the continuous results of (15) and (16) can improve the tractability of analyzing the asymptotic optimality and the instantaneous upper bounds of the queues.

4.3 Asymptotic Optimality and Tradeoff

We proceed to analyze the optimality of the new fully distributed online learning approach. Let $\widetilde{\Psi}^*(\mathbf{x}^t)$ be the cost achieved by the proposed approach, and Ψ^* be the optimum to (8) (which needs to be minimized offline based on the priori knowledge of network dynamics over an infinite time horizon). We can establish the typical $[\mathcal{O}(1/\epsilon), \mathcal{O}(\epsilon)]$-tradeoff of SGD in the following theorems by exploiting Lyapunov techniques [39].

Theorem 1. *The proposed approach is asymptotically optimal, i.e., the gap between $\widetilde{\Psi}^*(\mathbf{x}^t)$ and Ψ^* can asymptotically diminish as $\epsilon \to 0$ decreases, as given by*

$$\widetilde{\Psi}^*(\mathbf{x}^t) - \Psi^* \leq \epsilon U, \tag{18}$$

where $U = \frac{1}{2}\big\{\sum_{i\in\mathbf{N}}\big[\sum_{f\in\mathbf{F}} R^{\max}_{i,f} + 2\sum_{j\in\mathbf{N}} C^{\max}_{ij} + 2NF^{\max}\big]\big\}^2$ is a constant.

Proof. See Appendix B. □

Theorem 2. *The queue backlogs are upper bounded by $Q^s_{f,\max}$ and $D^s_{\max}$, such that $Q^s_{i,f}(t) \leq Q^s_{f,\max}$ and $D^s_i(t) \leq D^s_{\max}$, $\forall i, f, s, t$. The upper bounds $Q^s_{f,\max}$ and $D^s_{\max}$ can be given by*

$$Q^s_{f,\max} = \frac{\beta_{s,f}}{\epsilon} + R^{\max}_{s,f} + \theta_s; \tag{19a}$$

$$D^s_{\max} = \max_f \left\{R^{\max}_{s,f} + \frac{\beta_{s,f}}{\epsilon}\right\} + 2\theta_s + F^{\max}; \tag{19b}$$

where $\theta_s = \sum_{j\in\mathbf{N}_s} C^{\max}_{js}$ is the maximum arrivals of requests and files at edge server s. $\mathbf{N}_s$ is the set of the immediate neighbors of server s.

Proof. See Appendix C. □

It is important to reduce content retrieval latency and backhaul usage (which are tightly coupled). The performance metric in (6) captures both the cost of backhaul usage (when a content request is not cached in the edge cloud and must be retrieved from the backbone) and the cost of edge content delivery (when the content can be retrieved from the servers in the edge cloud). Here, "cost" is a generic measure to evaluate the performance of a system, and can be translated to monetary cost or energy consumption. For example, $\sum_{i,f}\zeta_{i,f}(R_{i,f}(t) - r_{i,f}(t))$ in (6) is the energy consumption of data retrieval over backhaul, where $\zeta_{i,f}$ is the energy of retrieving a unit size of file f from the backbone to edge server i. $\sum_{i,f}\zeta_{i,f}(R_{i,f}(t) - r_{i,f}(t))$ in (6) can also be translated to the consumption of backhaul bandwidth resources, if we set $\zeta_{i,f} = 1$. On the other hand, the content retrieval latency can be measured by the backlogs of unsatisfied content requests, according to Little's Law [36]. Captured in (7), the backlogs are also part of problem (8) and stabilized by the proposed approach. As shown in Theorem 2, by adjusting the SGD stepsize ϵ, we can balance between the content retrieval cost (accounting for the backhaul usage and the delivery of cached contents) and the content retrieval latency in the proposed approach.

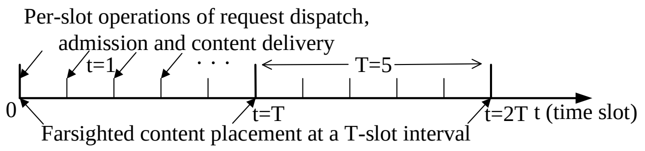

Per-slot operations of request dispatch,
admission and content delivery
t=1
· · ·
T=5
0
t=T
t=2T
t (time slot)
Farsighted content placement at a T-slot interval

Fig. 2. An illustrative example of two-timescale operations for content placement and delivery, where $T = 5$ time slots.

5 Two-timescale Mini-batch Learning for Farsighted Content Placement

This section considers a practical scenario where content placement needs to be conducted at a T-slot interval, as opposed to Section 4. Replacing cached files can incur overhead, i.e., for retrieving new files from the network backbone and updating the files in the local memories of the edge servers. The cost can increase substantially if cached files are excessively frequently replaced at the edge servers. As shown in Fig. 2, by running the proposed two-timescale mini-batch learning for farsighted content placement, the cached files are only replaced every T time slots (at slot $t = mT$, $m = 1, 2, \cdots$). The decisions of request dispatch, admission and content delivery are made at each time slot. As a result, the cache replacement cost of the network is inversely proportional to the value of T.

5.1 Farsighted Content Placement via Mini-Batch Learning

The two-timescale learning and optimization of content caching and delivery can operate as follows:

- *Content placement at large time intervals:* At slot $t = mT$, each edge server i decides the content placement $\mathbf{c}_i(t)$ by maximizing the expectation of the overall objective (13d) across $t = \{mT, \cdots, (m+1)T - 1\}$, i.e.,

$$\max\Big\{\mathbb{E}\big[\sum_{t=mT}^{(m+1)T-1}\sum_{f\in\mathbf{F}}\eta^s_{i,f}(t)b^s_{i,f}(t)c_{i,f}(t)|\omega^t\big]\Big\}, \text{ s.t. (3).} \tag{20}$$

- *Optimization of request dispatch, admission and content delivery per time slot:* At each slot $t = mT + \tau$, edge server i admits/dispatches requests and delivers contents based on (14) and (15), and grants the requests by (16), given the decisions of content placement in problem (20).

Note that the optimal content placement of (20) requires the explicit a-priori knowledge on ω^t across $\{mT, \cdots, (m+1)T-1\}$, including future requests and link availability, which is not practical in smart edge caching. We now derive the content placement by approximating the future queue backlogs (or Lagrange multipliers) as the current backlogs as of slot $t = mT$:

$$\begin{aligned}&\widehat{Q}^s_{i,f}(mT+\tau) = Q^s_{i,f}(mT);\\ &\widehat{D}^s_i(mT+\tau) = D^s_i(mT),\end{aligned} \tag{21}$$

where $\tau \in \{0, \cdots, T-1\}$, and $\widehat{Q}^s_{i,f}(t)$ and $\widehat{D}^s_i(t)$ are the approximate backlogs of the queues.

Here, (21) is typically adopted in the mini-batch learning of SGD in machine learning [40]. In specific, the stochastic gradients are summed up over a mini-batch (or the T-slot time window in this paper) and updated less frequently (at the interval of the size of a mini-batch). This helps reduce the variance of SGD resulting from the randomness of network dynamics.

By taking the approximation of (21), problem (20) can be reformulated as a sequence of per-slot subproblems; see (13d). Edge server i selects file f with the maximum urgency $\widehat{\eta}_{i,f}(t) = \max_{s\in\mathbf{N}}\{\eta^s_{i,f}(t)\}$ to be placed, as articulated in Section 4. Due to the randomness across $\{mT, \cdots, (m+1)T-1\}$, edge server i would cache the M_i files with the highest urgency $\widehat{\eta}_{i,f}(t)$ among all the files $f \in \mathbf{F}$. The urgency of the files $\widehat{\eta}_{i,f}(t)$ are only related to the queue backlogs (or Lagrange multipliers) without the a-priori statistical information of file popularity. This is achieved by SGD and mini-batch learning, where the Lagrange multipliers adapt to the pattern of randomness (request arrivals and link availability) and gradually converge to the optimum.

Fig. 3 shows the flowchart of the proposed approach at an edge server at slot t. Recall that the proposed approach is fully distributed, and all the edge servers run identical processes. Take edge server i for an example. The proposed approach optimizes the decisions on content placement and on request dispatch and content delivery at different timescales. Specifically, at each slot t, edge server i collects its local queue backlogs and the conditions of its connected links, exchanges the information of the queue backlogs with its neighbors (e.g., server j), and decides the optimal request admission, grant and dispatch, and content delivery based on (14)–(16). At every T-slot interval (i.e., if $t = mT$, $m = 1, 2, \cdots$), edge server i optimizes the content placement farsightedly by approximating the future backlogs according to (21) and replacing the files in its local memory, as stated in Section 5.1.

Note that the neighboring servers are expected to exchange (or synchronize) the information on their queue backlogs every time slot to compare their queue differences for the optimal decisions. Such signaling overhead of exchanging a 4-byte unsigned integer (which is sufficient to represent the queue backlogs of up to 4.3GB) is negligible as compared to the capacity of the network links between the edge servers.

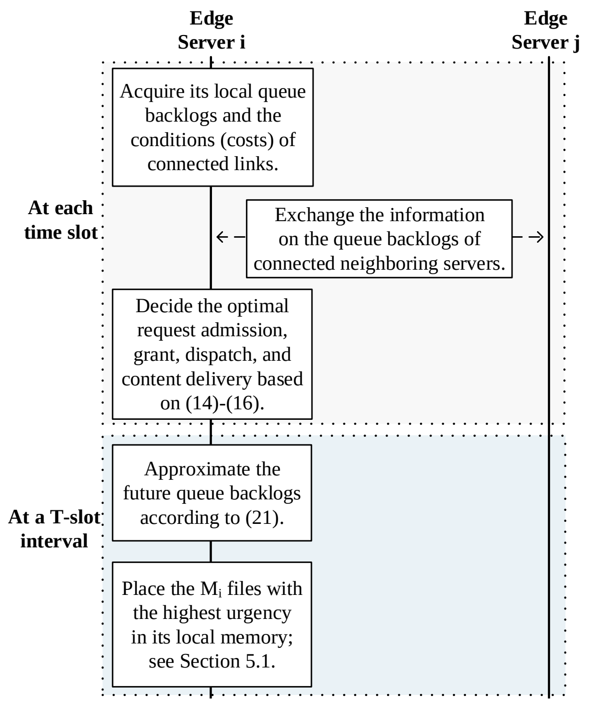

Edge Server i
Edge Server j
At each time slot
Acquire its local queue backlogs and the conditions (costs) of connected links.
Exchange the information on the queue backlogs of connected neighboring servers.
Decide the optimal request admission, grant, dispatch, and content delivery based on (14)-(16).
At a T-slot interval
Approximate the future queue backlogs according to (21).
Place the M_i files with the highest urgency in its local memory; see Section 5.1.

Fig. 3. The flowchart of the proposed approach at an edge server.

The proposed distributed online learning algorithm learns the file popularity and link availability in a distributed manner at each edge server. Given the learned knowledge, the proposed approach can jointly and asymptotically optimize the operations of content placement and delivery of cooperative caching. Both the file popularity and link availability are critical for the effectiveness of a cooperative caching algorithm, but they are hardly available in prior at the edge servers in practice, due to background traffic and user behaviors. The file popularity and link availability may be predicted in a centralized manner (e.g., by using deep neural networks), provided an instantaneous global view of the complete network. Unfortunately, the instantaneous global view of the network is barely available in practice, given the sheer size of the edge cloud and the non-negligible signaling delays. The proposed distributed online learning approach has a low time-complexity and is particularly designed to operate in real-time at individual resource-restrictive edge servers, as will be discussed in Section 5.2.

5.2 Complexity Analysis

The time-complexity of the approach is $\mathcal{O}(NF\delta_i)$ per edge server at the short timescale (per time slot) for optimal request dispatch, admission, and content delivery, and is $\mathcal{O}(F\log F)$ per edge server at the large timescale (per M time slots) for content placement. Here, δ_i is the degree of edge server i (or in other words, the number of immediate neighbors of the edge server).

At the short timescale (per time slot), the decisions of request dispatch, admission, and content delivery are optimized at each edge server in a distributed manner, according to (14)–(16). The time-complexity of solving (14)–(16)

depends on the complexity of evaluating and comparing the cost-effectiveness measures of $\alpha_{i,f}(t)$, $\beta^s_{ij,f}(t)$, $\gamma^s_{ij}(t)$ and $\eta^s_{i,f}(t)$. Each measure can be calculated in $\mathcal{O}(1)$ time. Each edge server i needs to acquire the measures for each file f, each requested server s, and all its immediate neighbors j in a total of $\mathcal{O}(NF\delta_i)$ time. Comparing the measures and generating the optimal decisions, i.e., executing (14), (15) and (16), would require $\mathcal{O}(F)$, $\mathcal{O}(NF\delta_i)$, and $\mathcal{O}(NF)$ time, respectively. The overall time-complexity of optimal request dispatch, admission, and content delivery is $\mathcal{O}(NF\delta_i)$ per time slot per edge server i.

At the large timescale (every T time slots), by using the approximation (21), the T-slot content placement problem (20) can be reduced to (13d), where the optimal solution for edge server i is to cache the M_i most urgent files in its storage. This needs to sort F files in the descending order of $\widehat{\eta}_{i,f}(t)$, and undergoes $\mathcal{O}(F\log F)$ time-complexity by using the quick-sort algorithm [41]. Therefore, the time-complexity of the optimal content placement at the large timescale is $\mathcal{O}(F\log F)$.

The time-complexity (or runtime) of the new approach is only linear to the numbers of edge servers and files at the short timescale and sub-quadratic to the number of files at the large timescale. As a result, the new distributed online learning technique can be optimized in real-time.

5.3 Performance Analysis

This section proves that the optimality loss resulting from the approximation of (21) asymptotically disappears as the stepsize $\epsilon \to 0$. For analytical tractability, an upper bound of the optimality loss is analyzed, under the assumption that the decisions of request dispatch, admission, and content delivery are optimized at a T-slot interval by extending the approximation of (21) to (14)-(16). This provides the lower bound of the performance offered by the two-timescale distributed learning.

Lemma 1. *The differences between the approximate and actual queue backlogs in (21) are upper bounded by*

$$|Q^s_{i,f}(t) - \widehat{Q}^s_{i,f}(t)| \le T\delta_i; \tag{22a}$$

$$|D^s_i(t) - \widehat{D}^s_i(t)| \le T\delta_i, \tag{22b}$$

where $\delta_i = \max_{f\in\mathbf{F}}\{F^{\max}_i + \sum_{j\in\mathbf{N}} C^{\max}_{ji}, R^{\max}_{i,f} + \sum_{j\in\mathbf{N}} C^{\max}_{ij}\}$ *is a constant.*

Proof. We can see from (21) that $\widehat{Q}^s_{i,f}(t) = Q^s_{i,f}(t-\tau)$, where $\tau \in \{0,\cdots,T-1\}$. According to the queue dynamics (4), we have $|Q^s_{i,f}(t+1) - Q^s_{i,f}(t)| \le \delta_i$. δ_i is the maximum difference of $Q^s_{i,f}(t)$ between consecutive time slots. As a result, $|Q^s_{i,f}(t) - \widehat{Q}^s_{i,f}(t)| = |Q^s_{i,f}(t) - Q^s_{i,f}(t-\tau)| = |\sum_{t_0=t-\tau}^{t-1} Q^s_{i,f}(t_0+1) - Q^s_{i,f}(t_0)|$. By applying $|a+b| \le |a|+|b|$, $|Q^s_{i,f}(t) - \widehat{Q}^s_{i,f}(t)| \le \tau\delta_i \le T\delta_i$ in (22a) is proved. Likewise, (22b) can be proved. □

Let $\widehat{\alpha}_{i,f}(t)$, $\widehat{\beta}^s_{ij,f}(t)$, $\widehat{\gamma}^s_{ij}(t)$ and $\widehat{\eta}^s_{i,f}(t)$ denote the approximate parameters in (13) under the approximation of (21) for farsighted content placement.

Lemma 2. *The differences between the actual and approximate parameters in (13) are upper bounded by*

$$|\widehat{\alpha}_{i,f}(t) - \alpha_{i,f}(t)| \le \epsilon T\delta_i; \tag{23a}$$

$$|\widehat{\beta}^s_{ij,f}(t) - \beta^s_{ij,f}(t)| \le \epsilon T(\delta_i + \delta_j); \tag{23b}$$

$$|\widehat{\gamma}^s_{ij}(t) - \gamma^s_{ij}(t)| \le \epsilon T(\delta_i + \delta_j); \tag{23c}$$

$$|\widehat{\eta}^s_{i,f}(t) - \eta^s_{i,f}(t)| \le 2\epsilon T\delta_i. \tag{23d}$$

Proof. According to (12), we have $|\widehat{\alpha}_{i,f}(t) - \alpha_{i,f}(t)| = \epsilon|Q^s_{i,f}(t) - \widehat{Q}^s_{i,f}(t)|$. As a result, we can prove (23a) according to Lemma 1.

For (23b), $|\widehat{\beta}^s_{ij,f}(t) - \beta^s_{ij,f}(t)| = \epsilon|[\widehat{Q}^s_{i,f}(t) - Q^s_{i,f}(t)] - [\widehat{Q}^s_{j,f}(t) - Q^s_{j,f}(t)]|$. By applying $|a-b| \le |a|+|b|$, we can obtain

$$|\widehat{\beta}^s_{ij,f}(t) - \beta^s_{ij,f}(t)| \le \epsilon[|\widehat{Q}^s_{i,f}(t) - Q^s_{i,f}(t)| + |\widehat{Q}^s_{j,f}(t) - Q^s_{j,f}(t)|]. \tag{24}$$

By substituting (22a) into the RHS of (24), (23b) can be proved. Likewise, (23c) and (23d) can be proved. □

Theorem 3. *The optimality loss of (11) using the approximation (21) under farsighted content placement is bounded, i.e.,*

$$\begin{aligned}&[\alpha(\mathbf{r}^t) + \eta(\mathbf{b}^t,\mathbf{c}^t) + \beta(\mathbf{q}^t) + \gamma(\mathbf{d}^t)] \\ &- [\alpha(\widehat{\mathbf{r}}^t) + \eta(\widehat{\mathbf{b}}^t,\widehat{\mathbf{c}}^t) + \beta(\widehat{\mathbf{q}}^t) + \gamma(\widehat{\mathbf{d}}^t)] \le \epsilon B\end{aligned} \tag{25}$$

where $\{\mathbf{r}^t,\mathbf{b}^t,\mathbf{c}^t,\mathbf{q}^t,\mathbf{d}^t\}$ *and* $\{\widehat{\mathbf{r}}^t,\widehat{\mathbf{b}}^t,\widehat{\mathbf{c}}^t,\widehat{\mathbf{q}}^t,\widehat{\mathbf{d}}^t\}$ *are the solutions to (11) under real-time and farsighted content placement, respectively, and* $B = 2T\big[\sum_{i,f} R^{\max}_{i,f}\delta_i + (F+1)\sum_{i,j}(\delta_i+\delta_j)C^{\max}_{ij} + 2\sum_i \delta_i F^{\max}_i\big]$ *is a constant.*

Proof. To prove this theorem, we first evaluate the loss of (12a), i.e., $\alpha(\mathbf{r}^t) - \alpha(\widehat{\mathbf{r}}^t)$. According to (12a), we can obtain

$$\begin{aligned}\alpha(\widehat{\mathbf{r}}^t) &= \sum_{i,f}\alpha_{i,f}(t)\widehat{a}_{i,f}(t) \\ &= \sum_{i,f}[\widehat{\alpha}_{i,f}(t)\widehat{a}_{i,f}(t) + (\alpha_{i,f}(t) - \widehat{\alpha}_{i,f}(t))\widehat{a}_{i,f}(t)].\end{aligned} \tag{26}$$

Recall that $\widehat{a}_{i,f}(t)$ is the optimal solution under the approximate parameter $\widehat{\alpha}_{i,f}(t)$. We can obtain

$$\begin{aligned}&\sum_{i,f}\widehat{\alpha}_{i,f}(t)\widehat{a}_{i,f}(t) \ge \sum_{i,f}\widehat{\alpha}_{i,f}(t)r_{i,f}(t) \\ &= \sum_{i,f}[\alpha_{i,f}(t)r_{i,f}(t) + (\widehat{\alpha}_{i,f}(t) - \alpha_{i,f}(t))r_{i,f}(t)] \\ &= \alpha(\mathbf{r}^t) + \sum_{i,f}(\widehat{\alpha}_{i,f}(t) - \alpha_{i,f}(t))r_{i,f}(t),\end{aligned} \tag{27}$$

where the second equality is because $\alpha(\mathbf{r}^t) = \sum_{i,f}\alpha_{i,f}(t)r_{i,f}(t)$ according to (12a). Substituting (23) into the subtraction of (27) and (26), we can obtain

$$\begin{aligned}\alpha(\mathbf{r}^t) - \alpha(\widehat{\mathbf{r}}^t) &\le \sum_{i,f}(\widehat{\alpha}_{i,f}(t) - \alpha_{i,f}(t))(r_{i,f}(t) + \widehat{a}_{i,f}(t)) \\ &\le 2\epsilon T\sum_{i,f} R^{\max}_{i,f}\delta_i.\end{aligned} \tag{28a}$$

Likewise, the optimality losses of (12b), (12c) and (12d) are upper bounded by

$$\beta(\mathbf{q}^t) - \beta(\widehat{\mathbf{q}^t}) \le 2\epsilon FT\sum_{i,j}(\delta_i+\delta_j)C^{\max}_{ij}; \tag{28b}$$

$$\gamma(\mathbf{d}^t) - \beta(\widehat{\mathbf{d}^t}) \le 2\epsilon T\sum_{i,j}(\delta_i+\delta_j)C^{\max}_{ij}; \tag{28c}$$

$$\eta(\mathbf{b}^t,\mathbf{c}^t) - \eta(\widehat{\mathbf{b}}^t,\widehat{\mathbf{c}}^t) \le 4\epsilon T\sum_i \delta_i F^{\max}_i. \tag{28d}$$

Adding up (28), we prove the theorem. □

We can prove that the proposed two-timescale learning and optimization of farsighted content placement and real-time operations is asymptotically optimal based on Theorems 1 and 3, i.e.,

$$\widetilde{\Psi}^*(\mathbf{x}^t) - \Psi^* \leq \epsilon(U + B). \tag{29}$$

It is also noted that the optimality loss of farsighted content placement is expected to increase linearly with T. The optimality loss can be traded for the cost of content replacement.

Theorems 1 – 3 provide an accuracy analysis of the proposed approach. Theorems 1 and 2 demonstrate that the new approach converges to offline optimum asymptotically as the stepsize of SGD $\epsilon \to 0$ under real-time content placement. The asymptotic optimality is due to the convexity of the formulated problem, while SGD can be used to fast converge to the global optimum of convex objective functions [37]. Theorems 3 shows that the SGD-based distributed online learning approach with mini-batch learning for farsighted content placement is also asymptotically optimal.

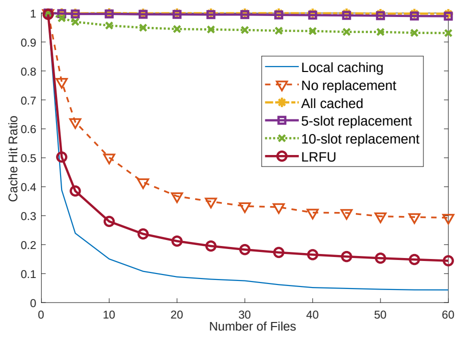
6 Simulation and Numerical Results

This section evaluates the proposed approach in a smart edge caching network of $N = 50$ edge servers and $F = 10$ files. The edge servers are connected to an average of $M = 3$ neighbors and have the storage size of caching one file; unless otherwise specified. The file requests arriving at each edge server are independently and uniformly distributed within $[6, 10]$Mbps, and the requests for each file follow Zipf distribution [42]. The popularity for file i is $p_i = \frac{1/i^{\xi}}{\sum_{f\in\mathbf{F}} 1/f^{\xi}}$ with the skew parameter $\xi = 0.8$; unless otherwise specified. The servers cache the files randomly with the probability of file popularity. The simulation parameters used in our simulation are taken from the existing dataset [18] and the existing work [29] (e.g., 50 edge servers, as well as the network topology/delivery cost). The cost of file delivery from peer servers, ζ_{ij}, is randomly and uniformly distributed from 0.05 to 0.15 per Megabit [18]. The energy consumption of file retrieval from the backbone is typically about 10 times of the energy consumption from peer servers [12], and therefore, the cost is set as $\zeta_{i,f} = 0.5$ per Megabit [29]. $1/\epsilon$ is 80. 50000 slots are tested per run.

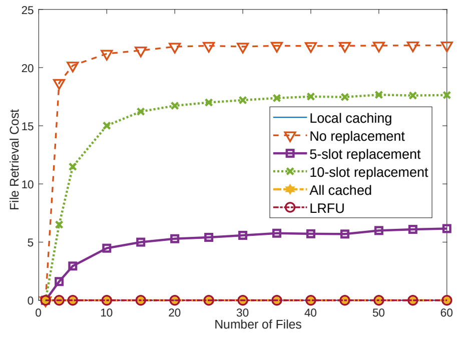
The new online learning and optimization of cooperative caching with a T-slot content replacement interval, as proposed in Section 5, is denoted by "T-slot replacement". We also simulate four existing benchmarks[3], as follows.

- Local caching, where the requests arriving at an edge server can be only retrieved from its local memory or delivered from the backbone.
- A non-cooperative Least Recently/Frequently Used (LRFU) approach, developed in [41], is the state-of-the-art reactive content replacement technique and used as the benchmark in this paper. LRFU [41] integrates the classical LRU and LFU strategies to balance between recent history (i.e., the timeliness of file requests) of LRU and old history (i.e., the frequency of file requests) of LFU. Files can be retrieved from either the local cache (without cooperative file delivery) of the edge servers or the backbone. For fair comparisons with the proposed 10-slot replacement approach, the non-cooperative LRFU approach replaces the cached files in the local memories of the edge servers every 10 time slots.
- All files cached in the local memories (denoted by "all cached" in the simulation), where the edge servers are assumed to have sufficient memories to cache all the files, and can meet the requests locally.
- Cooperative caching with no replacement of cached files at the edge servers (denoted by "no replacement" in the simulation), where the cached files at the edge servers remain unchanged.

3. This paper aims to jointly optimize both content placement and delivery in a distributed manner in the absence of the a-priori knowledge of file popularity and link availability. Existing caching algorithms, e.g., typical offline/one-shot optimization algorithms of cache placement without considering content delivery, are not directly applicable. They are also non-trivial to extend to serve the scenario of interest.



(a) Cache hit ratio

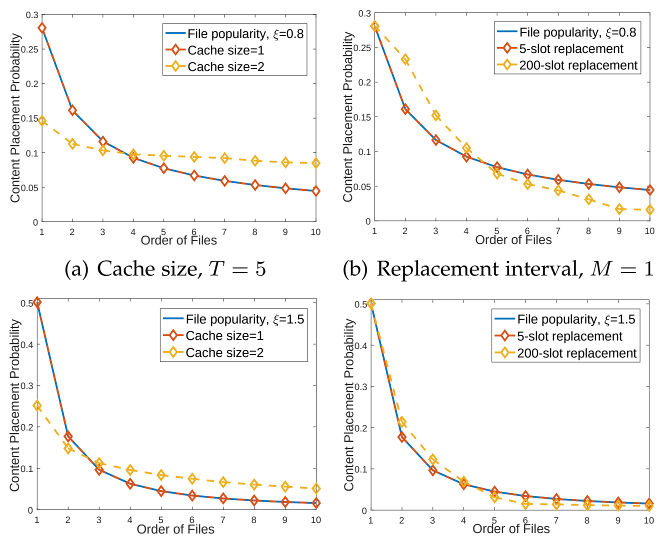


(b) File retrieval cost

Fig. 4. The cache hit ratio and cost of retrieving files within the edge cloud, as F increases. The proposed approach achieves significantly higher cache hit ratios than the other approaches, and the gain of cooperative caching comes at the penalty of increased file retrieval cost.

The distributed asymptotically optimal operations presented in Section 4 are carried out under all the cooperative caching approaches, but with different content replacement interval. In the all-cached approach, the edge servers are assumed to have sufficient memories to cache all the files (as compared to the practical consideration of finite memory per edge server in the proposed approach). This is only used to provide the upper bound of the proposed approach (designed for the edge servers with limited memories) and evaluate the effectiveness of farsighted content placement in the proposed distributed online learning approach.

Fig. 4 evaluates the cache hit ratio and the corresponding



(a) Cache size, $T = 5$ (b) Replacement interval, $M = 1$

(c) Cache size, $T = 5$ (d) Replacement interval, $M = 1$

Fig. 5. The probability of optimal content placement versus the popularity of $F = 10$ files generated by the Zipf distributions with $\xi = 0.8$ (for Figs. 5(a) and 5(b)) and $\xi = 1.5$ (for Figs. 5(c) and 5(d)). The probability of optimal content placement matches the file popularity under different popularity distributions when $M = 1$ and $T = 5$.

cost of retrieving files within the edge cloud of the proposed approach, local caching, caching all files locally, and cooperative caching with no cache replacement, as F increases. The proposed approach achieves significantly higher cache hit ratios than the local caching, LRFU, and no-replacement approaches. Both 5-slot replacement and caching all files locally can grant nearly all the file requests from the edge cloud, and their cache hit ratios are always close to 1. Due to the enlarged replacement interval, the cache hit ratio of 10-slot replacement slightly drops and stabilizes at 0.93, and the cache hit ratio of cooperative caching with no replacement drops to 0.4.

We also see in Fig. 4(a) that the LRFU and local caching approaches are susceptible to the increasing number of files F, and can only achieve the cache hit ratios of 15% and 5% when $F > 50$, respectively. In contrast, the proposed 5/10-slot replacement approaches can achieve a cache hit ratio of over 93%. This is because the requested files are less likely to be cached in the local memory of the edge servers with the increase of F. The LRFU approach replaces the cached contents according to the request arrivals, and achieves higher cache hit ratios than the local caching approach which caches files randomly based on the file popularity. The gain of the proposed cooperative caching approaches (as compared to the LRFU and local caching approaches) comes at the penalty of the increased file retrieval cost within the edge cloud, as demonstrated in Fig. 4(b). Decreasing T (e.g., from $T = 10$ to 5) can increase the cache hit ratio and help decrease the file retrieval cost from the edge cloud with an increasingly large cache replacement cost (which is inversely proportional to T). T can be configured by the network service provider to finetune the tradeoff between the cache hit ratio of edge caching and the cache replacement cost.

Fig. 5 shows the probability of optimal content placement at the edge servers under different cache sizes and replacement intervals, versus the popularity of $F = 10$ files generated by the Zipf distributions of $\xi = 0.8$ and 1.5. The "probability of optimal content placement" defines the probability of the files/contents being cached in the local memories of the edge servers (i.e., the ratio between the number of edge servers that cache a file and the total number of edge servers in the edge cloud) by the proposed asymptotically optimal approach. We show in Fig. 5 that the proposed algorithm can learn the file popularity in a distributed online manner via SGD, and the probability of content placement matches the file popularity in both cases of $\xi = 0.8$ and 1.5, when the replacement interval $T = 5$ and the cache size per server $M = 1$. In Fig. 5(a), when M increases to 2, the probability of caching the most popular file decreases to almost half of that under $M = 1$, and the probability of caching less popular files increases. This is because, with the growth of the cache size, the files with low popularity would be increasingly likely to be cached to reduce the cost of retrieving these files multi-hop away, until all the files are cached locally. In Fig. 5(b), the probability of caching the most popular four files increases with the replacement interval T. This coincides with the well-established Least Recently/Frequently Used (LRFU) policy [43]. When frequently refreshing the cached contents ($T = 5$), LRU policy would place the files according to their arrival (i.e., the same to file popularity); and when the cached contents are static ($T = 200$), LFU policy would cache the most popular contents. Figs. 5(c) and 5(d) for $\xi = 1.5$ show the same phenomenon as Figs. 5(a) and 5(b) for $\xi = 0.8$.

Note that the proposed distributed online learning approach neither takes the Zipf distribution of file popularity as input, nor has any a-priori knowledge on the distribution in the simulation. Over time, our approach learns the distribution from the request arrivals observed at the edge servers, and generates the optimal content placement which exhibits the Zipf-like shapes according to the file popularity, as shown in Fig. 4.

Figs. 6 and 7 evaluate the stabilized caching profit achieved by the four approaches: (1) the proposed T-slot replacement, (2) local caching, (3) caching all files, and (4) cooperative caching with no replacement, as T and $1/\epsilon$ increases. Fig. 6 shows that the stabilized caching profit (i.e., the optimality loss compared to caching all files locally) of the T-slot replacement decreases with the replacement interval T sub-linearly. This is because the linear optimality loss, as analyzed in Section 5.3, provides the upper bound of the mini-batch training. Nevertheless, the optimality loss is bounded by the gap between the cooperative caching with no replacement (which is equivalent to the case of $T \to \infty$) and the all cached approach (which provides the upper bound of our approach). The system cost of the proposed approach decreases with $1/\epsilon$. This is because increasing $1/\epsilon$ (i.e., decreasing stepsize of SGD) helps reduce the optimality loss, as stated in Theorem 1.

Fig. 8 plots the queue backlogs of the new approach, as t evolves. We can see in Fig. 8 that the backlogs of the system first increase and stabilize over time. The stabilized backlogs achieved by the 5-slot and 10/slot replacement approaches are similar, and are shorter than that of cooperative caching with no replacement. 10-slot replacement requires shorter time for stabilization than 5-slot replacement, due to the fact

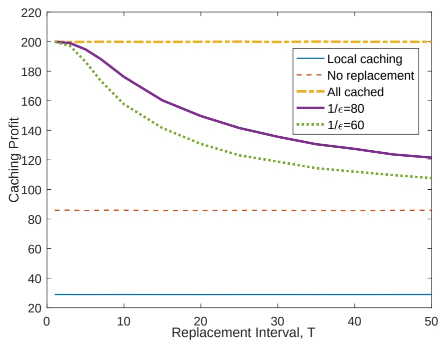


Fig. 6. The stabilized caching profit as the increase of replacement interval, T. The optimality loss resulting from the different two timescales increases sub-linearly with T.

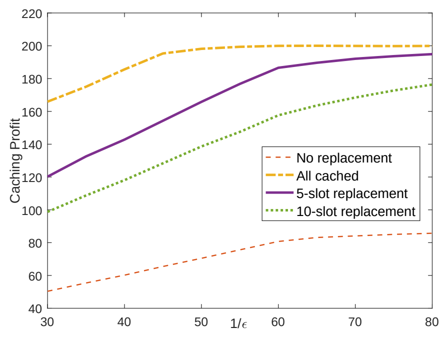


Fig. 7. The stabilized caching profit as the increase of reciprocal of stepsize, $1/\epsilon$. The optimality loss diminishes as the increase of $1/\epsilon$ (i.e., the decrease of stepsize for SGD).

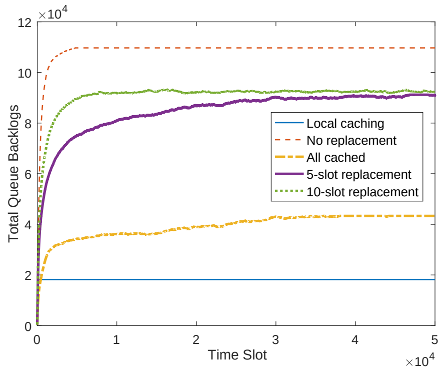


Fig. 8. The total queue backlogs of cooperative caching as t evolves. The total queue lengths of all the approaches first increase and then stabilize.

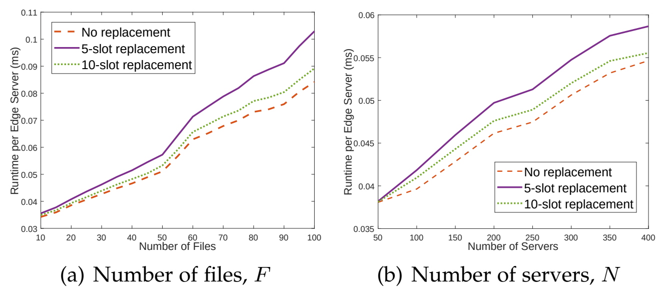


Fig. 9. The runtime of the proposed distributed approach per edge server, as the number of files F and the number of servers N increase.

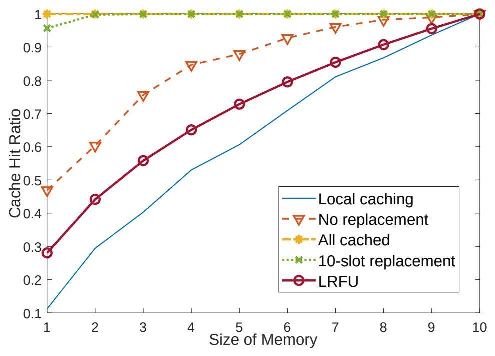


Fig. 10. The cache hit ratio within the edge cloud, as the memory sizes of the edge servers increase. The proposed approach can achieve significantly higher cache hit ratios than its alternatives.

that suitable mini-batch size (i.e., T) can help speed up the convergence of gradient descent in mini-batch learning.

Fig. 9 plots the runtime of the proposed distributed approaches per edge server, as the number of files F and the number of servers N increase. We see that the runtime of the proposed approaches increases sub-quadratically with F and linearly with N (which is consistent with the complexity analysis in Section 5.2), and is shorter than 0.11ms per edge server. This confirms the real-time implementability of our approach and its scalability to the network with large numbers of files and servers. We also see that the runtime of the 5-slot replacement approach is longer than that of the 10-slot replacement and no-replacement approaches. This is because, with the shortening replacement interval, the 5-slot replacement approach executes the content placement increasingly frequently, hence increasing the time-complexity.

Fig. 10 plots the cache hit ratio within the edge cloud, as the memory sizes of the edge servers increase from one file per server to $M_i = 10$ files. The total number of files F is 10. The proposed approach achieves up to 9 times the cache hit ratio of the local caching approach when $M_i = 1$, and can deliver all the contents within the edge cloud cooperatively when $M_i \geq 2$. The cache hit ratios of the alternative no-replacement, local caching, and LRFU schemes increase with M_i, as requested files are increasingly likely to be cached in the local memories of the edge servers. The cache hit ratios of the alternative schemes cannot reach 100% until $M_i = 10$ (i.e., all the files are already cached locally). This validates the gain of cooperative edge caching of the proposed approach.

7 Conclusion

This paper proposed a new distributed online learning technique to jointly optimize of content placement and delivery in edge clouds without prior knowledge on file popularity and link availability. SGD was exploited to design online learning to asymptotically minimize the cost of smart edge caching in a distributed fashion. By integrating mini-batch learning, the proposed approach operates at different timescales for content placement and delivery without compromising the asymptotic optimality. Simulations demonstrated the superiority of the proposed approach to the prior art.

References

[1] C. Xu, P. Zhang, S. Jia, M. Wang, and G. M. Muntean, “Video streaming in content-centric mobile networks: Challenges and solutions,” *IEEE Wireless Commun.*, vol. 24, no. 5, pp. 157–165, October 2017.

[2] D. Liu, B. Chen, C. Yang, and A. F. Molisch, “Caching at the wireless edge: design aspects, challenges, and future directions,” *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, September 2016.

[3] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[4] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, April 2017.

[5] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching and computing in wireless systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, 2017.

[6] X. Lyu, C. Ren, W. Ni, H. Tian, and R. P. Liu, "Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 574–586, March 2018.

[7] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud ran," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6118–6131, Sept 2016.

[8] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6284–6297, Sept 2016.

[9] ——, "Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2956–2970, July 2017.

[10] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," *Computer Communications*, vol. 36, no. 7, pp. 758–770, 2013.

[11] V. Sourlas, P. Flegkas, and L. Tassiulas, "Cache-aware routing in information-centric networks," in *2013 IEEE International Symposium on Integrated Network Management*, May 2013, pp. 582–588.

[12] P. Sena, A. Ishimori, I. Carvalho, and A. Abelém, "Cache-aware interest routing: Impact analysis on cache decision strategies in content-centric networking," in *LANC '16*. New York, NY, USA: ACM, 2016, pp. 39–45.

[13] K. Poularakis and L. Tassiulas, "On the complexity of optimal content placement in hierarchical caching networks," *IEEE Trans. Commun.*, vol. 64, no. 5, pp. 2092–2103, May 2016.

[14] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *2012 IEEE INFOCOM*, March 2012, pp. 2444–2452.

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

[16] Q. Li, W. Shi, X. Ge, and Z. Niu, "Cooperative edge caching in software-defined hyper-cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2596–2605, Nov 2017.

[17] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug 2016.

[18] R. Yu, S. Qin, M. Bennis, X. Chen, G. Feng, Z. Han, and G. Xue, "Enhancing software-defined ran with collaborative caching and scalable video coding," in *2016 IEEE ICC*, May 2016, pp. 1–6.

[19] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *2017 IEEE WONS*, Feb 2017, pp. 165–172.

[20] 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 Trans. Network.*, vol. 25, no. 3, pp. 1635–1648, June 2017.

[21] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug 2018.

[22] Y. Guo, Q. Yang, F. R. Yu, and V. C. Leung, "Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach," *IEEE Trans. Veh. Technol.*, 2018.

[23] G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu, "Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1286–1296, 2015.

[24] L. Sun, H. Pang, and L. Gao, "Joint sponsor scheduling in cellular and edge caching networks for mobile video delivery," *IEEE Trans. Multimedia*, 2018.

[25] J. Kwak, G. Paschos, and G. Iosifidis, "Dynamic cache rental and content caching in elastic wireless cdns," in *IEEE WiOpt*, 2018, pp. 1–8.

[26] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5g wireless networks: Cloud versus edge caching," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3030–3045, 2018.

[27] A. Chattopadhyay, B. Baszczyszyn, and H. P. Keeler, "Gibbsian on-line distributed content caching strategy for cellular networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 969–981, Feb 2018.

[28] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug 2016.

[29] C. Ren, X. Lyu, W. Ni, H. Tian, and R. P. Liu, "Profitable cooperative region for distributed online edge caching," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 4696–4708, July 2019.

[30] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2393–2406, Oct 2019.

[31] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov 2017.

[32] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Distributed online optimization of fog computing for selfish devices with out-of-date information," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7704–7717, Nov 2018.

[33] C. Ren, X. Lyu, W. Ni, H. Tian, W. Song, and R. P. Liu, "Distributed online optimization of fog computing for internet-of-things under finite device buffers," *IEEE Internet Things J.*, p. to appear, 2020.

[34] X. Lyu, C. Ren, W. Ni, H. Tian, and R. P. Liu, "Cooperative computing anytime, anywhere: Ubiquitous fog services," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 162–169, February 2020.

[35] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and Y. J. Guo, "Multi-timescale decentralized online orchestration of software-defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2716–2730, Dec 2018.

[36] A. O. Allen, *Probability, statistics, and queueing theory*. Academic Press, 2014.

[37] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *COMPSTAT*, 2010, pp. 177–186.

[38] G. Dantzig, *Linear programming and extensions*. Princeton University Press, 2016.

[39] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[40] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *ACM SIGKDD*. ACM, 2014, pp. 661–670.

[41] R. Sedgewick, "The analysis of quicksort programs," *Acta Informatica*, vol. 7, no. 4, pp. 327–355, 1977.

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

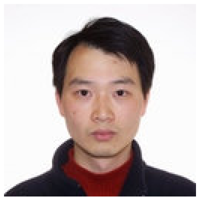
[43] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Computers*, no. 12, pp. 1352–1361, 2001.

[44] T. Chen, A. Mokhtari, X. Wang, A. Ribeiro, and G. B. Giannakis, "Stochastic averaging for constrained optimization with application to online resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3078–3093, June 2017.

[45] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

Xinchen Lyu received the B.E. degree from Beijing University of Posts and Telecommunications (BUPT) in 2014, and his dual Ph.D. degrees from BUPT and University of Technology Sydney in 2019. He is currently an associate researcher at BUPT. His research interests include stochastic optimization and machine learning with applications to fog computing, edge caching, software-defined networking, and resource management.

Chenshan Ren received her B.E. degree from Zhengzhou University, Henan, China, in 2013, and her dual Ph.D. degrees from BUPT and University of Technology Sydney in 2019 and 2020, respectively. She is currently a lecturer at Minzu University of China. Her research interests include fog computing, edge caching, and radio resource management.



Wei Ni received the B.E. and Ph.D. degrees in Electronic Engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively. Currently he is a Senior Scientist, and Team and Project Leader at CSIRO, Australia. He also holds honorary positions at the University of New South Wales (UNSW), Macquarie University (MQ) and the University of Technology Sydney (UTS). Prior to this he was a postdoctoral research fellow at Shanghai Jiaotong University (2005-2008), Research Scientist and Deputy Project Manager at the Bell Labs R&I Center, Alcatel/Alcatel-Lucent (2005-2008), and Senior Researcher at Devices R&D, Nokia (2008-2009). His research interests include optimization, game theory, graph theory, as well as their applications to network and security.

Dr Ni serves as Editor for Hindawi Journal of Engineering since 2012, secretary of IEEE NSW VTS Chapter since 2015, Track Chair for VTC-Spring 2016 and 2017, and Publication Chair for BodyNet 2015. He also served as Student Travel Grant Chair for WPMC 2014, Program Committee Member of CHINACOM 2014, TPC member of IEEE ICC'14, ICCC'15, EICE'14, and WCNC'10.

Hui Tian received her M.S. in Micro-electronics and Ph. D degree in circuits and systems both from Beijing University of Posts and Telecommunications, China, in 1992 and 2003, respectively. Currently, she is a professor at BUPT, the Network Information Processing Research Center director of State Key Laboratory of Networking and Switching Technology and the MAT director of Wireless Technology Innovation Institute (WTI). Her current research interests mainly include radio resource management, cross layer optimization, M2M, cooperative communication, mobile social network, and mobile edge computing.

Ren Ping Liu is a Professor at the School of Electrical and Data Engineering in University of Technology Sydney, where he leads the Network Security Lab in the Global Big Data Technologies Centre. He is also the Research Program Leader of the Digital Agrifood Technologies in Food Agility CRC, a government/research/industry initiative to empower Australia's food industry through digital transformation. Prior to that he was a Principal Scientist at CSIRO, where he led wireless networking research activities. He specialises in protocol design and modelling, and has delivered networking solutions to a number of government agencies and industry customers. Professor Liu was the winner of Australian Engineering Innovation Award and CSIRO Chairman medal. His research interests include Markov analysis and QoS scheduling in WLAN, VANET, IoT, LTE, 5G, SDN, and network security. Professor Liu has over 100 research publications, and has supervised over 30 PhD students.

Professor Liu is the founding chair of IEEE NSW VTS Chapter and a Senior Member of IEEE. He served as Technical Program Committee chair and Organising Committee chair in a number of IEEE Conferences. Ren Ping Liu received his B.E.(Hon) and M.E. degrees from Beijing University of Posts and Telecommunications, China, and the Ph.D. degree from the University of Newcastle, Australia.

Xiaofeng Tao received the B.S. degree in electrical engineering from Xian Jiaotong University, Xian, China, in 1993, and the M.S.E.E. and Ph.D. degrees in telecommunication engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 1999 and 2002, respectively. He is currently a Professor with BUPT. He has authored or coauthored 250 papers and three books in wireless communication areas. He was an Inventor or a Co-Inventor of 74 patents. He currently focuses on 5G research. He is the Chair of the IEEE ComSoc Beijing Chapter.