LEAD: Large-Scale Edge Cache Deployment Based on Spatio-Temporal WiFi Traffic Statistics

Feng Lyu, *Member, IEEE*, Ju Ren, *Member, IEEE*, Nan Cheng, *Member, IEEE*, Peng Yang, *Member, IEEE*, Minglu Li, *Senior Member, IEEE*, Yaoxue Zhang, *Senior Member, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—Widespread and large-scale WiFi systems have been deployed in many corporate locations, while the backhual capacity becomes the bottleneck in providing high-rate data services to a tremendous number of WiFi users. Mobile edge caching is a promising solution to relieve backhaul pressure and deliver quality services by proactively pushing contents to access points (APs). However, how to deploy cache in large-scale WiFi system is not well studied yet quite challenging since numerous APs can have heterogeneous traffic characteristics, and future traffic conditions are unknown ahead. In this paper, given the cache storage budget, we explore the cache deployment in a large-scale WiFi system, which contains 8,000 APs and serves more than 40,000 active users, to maximize the long-term caching gain. Specifically, we first collect two-month user association records and conduct intensive spatio-temporal analytics on WiFi traffic consumption, gaining two major observations. First, per AP traffic consumption varies in a rather wide range and the proportion of AP distributes evenly within the range, indicating that the cache size should be heterogeneously allocated in accordance to the underlying traffic demands. Second, compared to a single AP, the traffic consumption of a group of APs (clustered by physical locations) is more stable, which means that the short-term traffic statistics can be used to infer the future long-term traffic conditions. We then propose our cache deployment strategy, named LEAD (i.e., Large-scale WiFi Edge cAche Deployment), in which we first cluster large-scale APs into well-sized edge nodes, then conduct the stationary testing on edge level traffic consumption and sample sufficient traffic statistics in order to precisely characterize long-term traffic conditions, and finally devise the *TEG* (Traffic-wEighted Greedy) algorithm to solve the long-term caching gain maximization problem. Extensive trace-driven experiments are carried out, and the results demonstrate that LEAD is able to achieve the near-optimal caching performance and can outperform other benchmark strategies significantly.

Index Terms—Large-scale WiFi system, edge cache deployment, caching gain maximization, Big Data analytics, stationary traffic consumption

1 INTRODUCTION

To accommodate the soaring number of mobile devices and the exponentially growing data traffic demands, most corporate places (including enterprises, airports, campuses, etc.) have deployed large-scale WiFi systems to provide users with full indoor coverage and high-speed Internet experience [1], [2], [3], [4], [5], [6]. According to the most recent forecast by Cisco [7], the number of global public WiFi hotspots will grow from 124 million in 2017 to 549 million in 2022, with about four-fold growth, and WiFi traffic will account for more than 51 percent of total IP traffic in 2022, rising from 43 percent in 2017. Inevitably, massive connections and high-volume traffic by WiFi devices will impose substantial pressure on the backhaul network and thus degrade user quality of experience (QoE).[1] Mobile edge caching [8], [9], [10], [11], [12], [13], [14] has been envisioned as a prominent paradigm to alleviate the unprecedented backhaul pressure and enhance user QoE. Popular contents can be cached at the network edge that are in close proximity to mobile users. Instead of fetching requested contents from cloud center after multi-hop transmission through backhaul and core networks, users can get requested contents with one-hop wireless transmission, which significantly reduces end-to-end latency and improves user QoE. Considering the high penetration and large scale of WiFi systems, WiFi access points (APs) are ideal carriers for edge cache.

To enable efficient caching in large-scale WiFi system, edge cache deployment is the essential building block, which however is quite challenging due to the following

- *F. Lyu, J. Ren, and Y. Zhang are with the School of Computer Science and Engineering, Central South University, Changsha, Hunan 410083, P.R. China. E-mail: {fenglyu, renju, zyx}@csu.edu.cn.*
- *N. Cheng is with the State Key Lab. ISN, China, and also with the School of Telecommunications Engineering, Xidian University, Xian, Shaanxi 710071, P.R.China. E-mail: dr.nan.cheng@ieee.org.*
- *P. Yang is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, Hubei 430074, P.R.China. E-mail: yangpeng@hust.edu.cn.*
- *M. Li is with the College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua, Zhejiang 321004, and also with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, P.R.China. E-mail: mlli@sjtu.edu.cn.*
- *X. Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 21 June 2019; revised 16 Feb. 2020; accepted 24 Mar. 2020. Date of publication 2 Apr. 2020; date of current version 1 July 2021. (Corresponding author: Ju Ren.) Digital Object Identifier no. 10.1109/TMC.2020.2984261

1. In this paper, QoE means the content delivery experience, which is related to the metrics of throughput, delay, jitter, etc.

three reasons. First, deploying edge cache in large-scale WiFi system (e.g., more than 8,000 APs in our system) is labor-intensive and time-consuming. It is unlikely to deploy edge cache at each AP, and an efficient cache placement strategy, i.e., choosing appropriate APs to deploy edge cache, is of capital importance. Second, APs are deployed in a wide area, resulting in differentiated user association and traffic consumption. How to allocate the limited cache storage budget to serve more users and meanwhile alleviate the backhual burden is non-trivial. On one hand, if excessive cache storages are allocated to those APs with less user associations, cache resource will be wasted. On the other hand, for those popular APs where users frequently associate to and consume intensive data traffic, allocating insufficient cache storages there may fail to fully unleash the potential of edge caching. Third, due to the underlying user influx variations and mobilities, the user association and traffic consumption at APs would vary over time while the caching deployment is a static one-shot solution. It is difficult to maintain the highest caching gain in the long run without knowing the future traffic conditions.

Recently, there have been some researches on mobile edge caching, while they mainly focus on content placement and assume that caching servers and cache storages are given ahead. Particularly, in the article [15], Bastug *et al.* explained the basic concepts of proactive caching at edge. Ma *et al.* [16] investigated which contents should be stored by which APs, to maximize the total caching hit rate of all the APs. Also with the target of maximizing the total content hit rate, Yang *et al.* [14] devised a location-aware caching scheme after observing that user requests on certain contents vary significantly at different locations. To fully utilize the limited cache space, Cao *et al.* [17] proposed an auction mechanism to elicit true valuations from the users (incentive compatibility), in order to optimize the content placement. Considering the scenarios where users are covered by multiple base stations (BSs) simultaneously, Zhang *et al.* [9] and Leonardi *et al.* [18] studied the cooperation caching problem to optimize the user caching performance. With the emergence of UAV techniques, some studies have employed UAVs to act as edge nodes to provide caching services [19], [20]. To enhance the caching hit rate, big data and machine learning based techniques are also investigated [21], [22], [23], [24], [25], to benefit the file request prediction. To summarize, these studies mainly concentrate on the content request and replacement at caching server, while neglecting the essential cache deployment problem. Instead, in this paper, we investigate the cache deployment problem in large-scale WiFi system, i.e., where to deploy edge caches and how to allocate cache size for them. The problem is the precondition of edge caching system while rarely seen in the literature.

In this paper, given a limited cache storage budget, we first formulate a large-scale WiFi caching gain maximization (*CGM*) problem, aiming at maximizing the total reduced backhaul traffic in the long term, which is intractable directly since the future user association and traffic consumption are unknown ahead. We then adopt the data analytics to study the problem. Specifically, we collect all user association records within two months in the WiFi system, with totally 41,119,940 association records of 36,952 users. We first conduct statistical analysis on the daily traffic consumption at each AP and two important observations are obtained: 1) the per AP traffic consumption *spans within a rather wide range*, indicating that cache storages should be allocated in a heterogeneous way with respect to the underlying traffic demands; 2) the proportion of APs with extremely large traffic consumption appears quite small, which means that it is possible to achieve quite large rewards when more cache storages are allocated there. To understand how users consume traffic among APs, we then analyze the traffic-weighted entropy (defined as the measure of AP "popularity") of APs and find that *the AP entropy distribution has an exponential decay*. It indicates that the proportion of APs with relatively large entropy values (i.e., important deployment targets) is very limited, which benefits the AP selection to deploy edge cache. After checking the Jaccard similarity score between the AP traffic consumption and popularity ranking, we observe that *the AP traffic consumption and its popularity are highly correlated*, i.e., more traffic consumption at the AP indicates that it is more popular in user association. This inspires us to concentrate on mining the traffic consumption to design our cache deployment strategy.

Based on these insightful observations, we then propose LeaD, i.e., Large-scale WiFi Edge cAche Deployment. In LeaD, we first cluster neighboring APs into well-sized edge nodes according to their physical locations, which can achieve the following three major merits: 1) cutting down the number of deploying locations can save much deploying time and labor; 2) users from multiple APs can share the cache storage with multiplexing gain, which improves the caching resource utility; 3) it is observed that the fluctuation of edge level traffic consumption is more stable than that of a single AP level, benefiting the traffic modeling which is important to the cache deployment. We then perform the stationary testing (i.e., Augmented Dickey-Fuller (ADF) test) on time series of edge traffic consumption and disclose that *the time series of edge traffic consumption is stationary*, which means that the expectation of future long-term traffic consumption can be represented by the short-term mean value. Given the estimation of future traffic consumption, we then devise the *TEG* (Traffic-wEighted Greedy) algorithm to solve the *CGM* problem. At last, we implement our deploying strategy and conduct extensive trace-driven experiments to demonstrate its efficacy. Specifically, LeaD is able to achieve the near-optimal performance and can outperform two benchmark strategies significantly in terms of caching gain and caching resource utility. In addition, as LeaD is a static one-shot solution, to evaluate its temporal robustness, we further collect a new one-month trace after nearly one year, based on which we conduct experiments to verify the robustness.

Our contributions in this paper are three-fold as follows.

- We consider the edge cache deployment in large-scale WiFi system, which is an essential building block for future edge caching system. To allocate the limited cache storage budget, we formulate the *CGM* problem to maximize the long-term cached traffic, which however is intractable directly without knowing future user association and traffic consumption.

Fig. 1. An example of AP deployment.

- We adopt the data-analytics methodology to resolve the problem, and collect huge-volume user association records, which cover 36,952 users and last two months. The trace can also be of great value for other researches such as user performance analysis and enhancement, models tuning, design verification, etc. We have opened the trace for public access.[2] Based on the trace, we conduct extensive statistical studies on user spatio-temporal association and traffic consumption, and achieve several insightful observations, which can direct our road map empirically.
- Given a cache storage budget, we propose LEAD to deploy them in the large-scale WiFi system in order to achieve the maximum long-term caching gain. In LEAD, we integrate three major techniques: 1) AP clustering and edge node formation; 2) traffic stationary testing; and 3) long-term caching gain maximization algorithm design. Extensive trace-driven experiments are carried out and results demonstrate its efficacy with temporal robustness.

The remainder of this paper is organized as follows. Section 2 describes the system and gives the problem definition. We detail our data collection and conduct overall traffic analysis in Section 3. Section 4 elaborates on our LEAD design. In Section 5, we evaluate the performance of the proposed LEAD with trace-driven experiments. Section 6 reviews the related work. Finally, we conclude the paper and direct our future work in Section 7.

2 System Description and Problem Definition

In this section, we first describe the large-scale WiFi system by detailing its network architecture. Then, we give the problem formulation of cache deployment, followed by the problem traceability analysis.

2.1 Large-Scale WiFi System

Full WiFi coverage systems are now widely deployed in corporate places, while edge caching is essential in such systems to meet future explosive traffic demand. We consider deploying edge cache in our campus WiFi system, which contains 8,000 APs, to provide full WiFi coverage for more

2. Online available. https://github.com/Intelligent-WiFi/DataSet

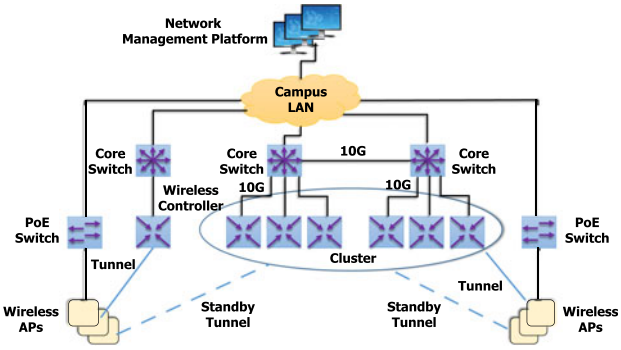


Fig. 2. The architecture of the large-scale WiFi system.

than 40,000 active users in the area of 3.0925 km^2. In the system, APs are scattered in all teaching buildings, dining halls, office buildings, libraries and different departments. As shown in Fig. 1, APs are mounted on room ceilings, and each room normally has one AP, except for the most common and social places such as dinning halls and libraries where multiple APs are installed in a room to cover high-density users. Fig. 2 shows the architecture of the large-scale WiFi system, in which each Power over Ethernet (PoE) switch connects to multiple APs to power them up and route their data, and each wireless controller is in charge of a cluster of APs by providing IP and authentication services, sending remote controlling commands and collecting daily management logs, etc. Both PoE switches and core switches are connected to the campus Local Area Network (LAN), which connects to the Internet via the backhaul network. In this paper, we investigate edge caching under the current end-to-end network architecture, i.e., just pushing contents to the edge node without enabling cooperation among them. Particularly, when a user requests the data, the edge node will return the content directly if it has stored the data proactively, otherwise, the user has to fetch the data from the Internet via backhaul. In this way, the caching system can be fast implemented without additional modification in the current end-to-end network architecture.

2.2 Problem Definition and Tractability Analysis

The large-scale WiFi system imposes substantial pressure on the backhual network as there are a large number of user associations and huge data traffic consumption, which would inevitably incur high latency and deteriorate user performance. Enabling content delivery at the network edge can effectively reduce backhaul burden and directly contribute to user performance enhancement. Yet, the deployment is labor-intensive and costly, and cache storages are limited. Hence, maximizing the cache deploying gain is the core concern of this paper. In particular, we aim at allocating the cache storages to a limited number of APs in order to maximize the long-term caching gain. Therefore, we cast our cache deployment problem as follows.

Definition 1 (Caching Gain Maximization, CGM). *Given the total cache storage budget C, how to allocate them to the APs in a large-scale WiFi system, such that the total reduced backhaul traffic is maximized in the long term?*

We partition the long-term duration into T consecutive time slots, and suppose there are N APs in the system,

denoted as $\Psi = \{AP_1, AP_2, \ldots, AP_N\}$. In addition, for AP_i ($AP_i \in \Psi$), we suppose there are M_i^t users associated to it during the time slot t, and denote by $\mathcal{U}_i^t = \{u_1, u_2, \ldots, u_{M_i^t}\}$ the set of associated users. Let $\mathcal{C} = \{c_1, c_2, \ldots, c_i, \ldots, c_N\}$ be the set of allocated cache sizes for N APs. Thus, the *CGM* problem can be formulated as follows

$$\begin{aligned} \max_{\{c_i\}} \quad & \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} f(c_i)\sum_{j=1}^{M_i^t} V_{ij}^t, \\ \text{s.t.} \quad & \sum_{i=1}^{N} c_i \le C, \end{aligned} \tag{1}$$

where V_{ij}^t is the traffic consumption of user j at the AP i during the time slot t, and $f(c_i)$ is the caching ratio function. The caching ratio function is related to the cache size c_i and the content placement strategy. As this paper focuses on the cache size allocation rather than content placement, given the cache size, we assume that the content popularity follows a Zipf distribution and the least frequently used (LFU) content replacement strategy is adopted. Such assumptions have been widely adopted in existing literature [9], [12], [14]. Specifically, let $\mathcal{F} = \{1, 2, \ldots, f, \ldots, F\}$ be the file set and F be the total number of files. Denote by $\mathcal{Q} = \{q_1, q_2, \ldots, q_f, \ldots, q_F\}$ the file popularity distribution (sorted in descending order, $\sum_{f=1}^{F} q_f = 1$), where q_f is the probability that the requested content is file f. Therefore, according to the Zipf distribution,

$$q_f = \frac{1/f^{\alpha}}{\sum_{h=1}^{F} 1/h^{\alpha}}, \tag{2}$$

where α represents the skewness of popularity distribution, and larger α means more concentrated file requests, which normally ranges from 0.4 to 1 [9]. Assume the size of each file is s and the cache size are integer multiple of s. Note that, for files with different sizes, the proposed methodology can be easily applied by dividing each file into chunks of equal size. Then the *CGM* problem can be rewritten as

$$\begin{aligned} \max_{\{c_i\}} \quad & \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{f=1}^{c_i}\frac{1/f^{\alpha}}{\sum_{h=1}^{F} 1/h^{\alpha}}\sum_{j=1}^{M_i^t}\frac{V_{ij}^t}{s}, \\ \text{s.t.} \quad & \sum_{i=1}^{N} c_i \le C,\ 0 \le c_i \le F. \end{aligned} \tag{3}$$

Directly tackling the *CGM* problem is difficult, since in the large-scale WiFi system, the future long-term user association behaviors (i.e., M_i^t) vary dynamically and traffic consumptions (i.e., V_{ij}^t) are unavailable ahead. Meanwhile, there is no well established model that could capture such large-scale dynamics and future long-term uncertainties. Therefore, in this paper, we study this problem by means of data analytics and elaborate on the process in the following sections.

3 Data Collection and Overall Traffic Analysis

In order to understand realistic traffic statistics and design an informed cache deployment strategy, it is essential to study empirical traffic data in terms of distribution and variation in the large-scale WiFi system. In this section, we describe our data collection campaign and conduct overall traffic analysis, so as to identify the crucial factors that affect the cache deployment gain.

3.1 User Association Records Collection

For cache deployment, user traffic consumption pattern is of significance, i.e., where do users consume the most traffic and how much traffic is consumed. To capture this characteristic, we collect all user association records including the association time, disassociation time, consumed traffic volume, etc., from the large-scale WiFi system. Specifically, we implement a data collection program running in *JAVA* to call the data management APIs (Application Programming Interfaces) of the network management platform. The platform can collect raw management data from all APs under the simple network management protocol (SNMP) and generate traffic statistics in real time, and therefore it is able to log all association records of each user, which can be stored in the platform for one week. By calling the API, we download all records every week to obtain the complete association records, and Listing 1 shows an association record sample of a user. It is shown that, each association record contains a unique association ID, the associated AP ID and name, used bytes during the association, connection time, disconnection time, radio association mode, etc. Particularly, in the AP name format of "XXX-xxx-3F-303", "XXX" and "xxx" are the abbreviation of the campus and building, respectively, and "3F" indicates that the AP is installed on the third floor of the building. The radio mode "ac" means the IEEE 802.11ac standard which operates at the 5 GHz frequency band, and APs can simultaneously support IEEE 802.11n connection by another radio (operating at the 2.4 GHz). The data collection lasts for more than two months, starting from May 8 to Jul. 11 in 2018, and we collect total 41,119,940 association records across 7,710 APs covering 36,952 users (with total data size over 35 GB).[3]

Listing 1. An Association Record Sample

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
  <client mac="00:00:C5:66:F1:46">
    <association id="40459632">
      <ap id="1547">"XXX-xxx-3F-303"</ap>
      <bytes_used>"230633700"</bytes_used>
      <conn_time>"2018-05-17T13:49:50"</conn_time>
      <disconn_time>"2018-05-17T15:44:06"</disconn_time>
      <radio_mode>"ac"</radio_mode>
      <rssi>"46"</rssi>
      <ssid>"campus"</ssid>
      <username>"Bob8062"</username>
    </association>
    <association id="36080348">
      ......
    </association>
  </client>
```

3. Note that, as there is a small portion of APs (about 800) managed by another management platform, we didn't collect the association data on those APs.

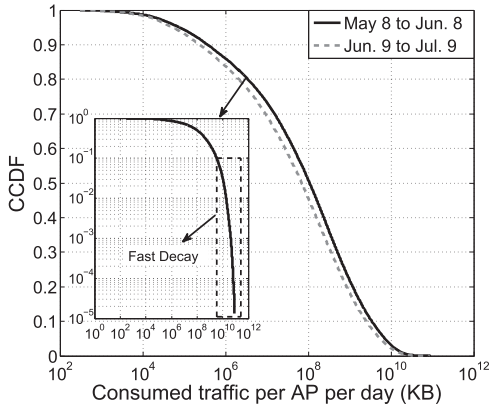


Fig. 3. CCDFs of AP traffic consumption.

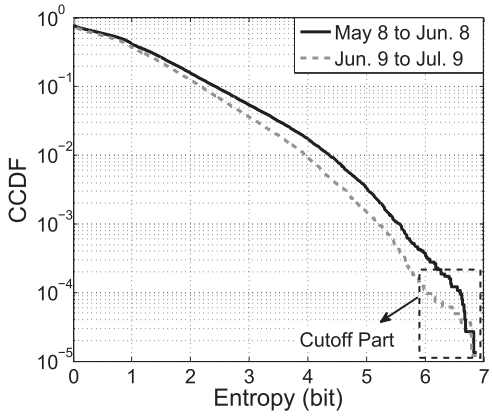


Fig. 4. CCDFs of AP traffic-weighted entropy.

3.2 Overall Traffic Consumption Analysis

3.2.1 Widely Spanned Traffic Consumption

To gain an overall picture, we first examine the complementary cumulative distribution function (CCDF) of daily traffic consumption at each AP. Fig. 3 shows the CCDF results of data sets during May 8 to Jun. 8 and Jun. 9 to Jul. 9, respectively, and we have the following three major observations related to cache deployment. First, the CCDF results in two different months vary quite analogously as two curves are highly close to each other. It means that the caching gain can sustain for a long run if the initial deployment strategy is well designed as the traffic consumption is shown to be similar everyday. Second, the AP traffic consumption spans widely, and the proportion is evenly distributed within a relatively wide traffic range. For instance, in both CCDF curves, the proportion of the daily traffic consumption between 10^4 and 10^{10} KB accounts for more than 90 percent, and the proportion distributes rather evenly within the traffic range. Widely spanned traffic consumption indicates that APs are heterogeneous in terms of traffic generation, calling for customized cache size design in order to maximize the caching gain. Third, when the traffic consumption becomes extremely large, e.g., more than 10^9 KB in the result of May 8 to Jun. 8 (in log-log scale), the proportion has a very fast decay. Specifically, the total proportion is no more than 10 percent when the traffic consumption ranges from 10^9 to 10^{11} KB, which means that it is possible to achieve rather impressive caching gain if the cache can be properly deployed at those few APs whose traffic consumption are extremely large.

3.2.2 Exponential Distribution of AP Popularity

With the overall traffic consumption knowledge, we then investigate how the traffic is consumed by users, i.e., how many users associate to the AP and how much traffic they consume. To this end, we define the *traffic-weighted entropy* to measure the *AP popularity* in terms of user association and traffic consumption [26], [27]. Specifically, given an AP_i ($AP_i \in \Psi$), let V_i be the set of traffic consumption (during each association) at AP_i, and $v_i = |V_i|$ be the total traffic consumption at AP_i. Also, let U_i be the set of distinct users that consumed traffic at AP_i, and $V_{i,u}$ be the set of traffic that user u has consumed at AP_i. Thus, $v_{i,u}$ means the total traffic consumption of user u at AP_i. The probability that the traffic consumption at AP_i is contributed by the user u, is $p_{i,u} = \frac{v_{i,u}}{v_i}$, indicating the fraction of total traffic consumption at AP_i belonging to user u. We define the traffic-weighted entropy for AP_i as follows

$$H(i) = H(p_{i,u_1}, p_{i,u_2}, \ldots, p_{i,u_{|U_i|}}) = \sum_{u \in U_i} p_{i,u} \log_2 \frac{1}{p_{i,u}} \ . \tag{4}$$

An AP with higher traffic-weighted entropy means that the traffic consumption distributes more evenly by more associated users, and vice versa.

We calculate the daily traffic-weighted entropy of each AP, and plot their CCDF results in Fig. 4. It can be seen that in both CCDF curves, *the traffic-weighted entropy has an exponential decay* since clear linear plot appears under linear-log scale. The tail cutoff part appears due to the insufficient sampling, and can be ignored since its proportion is negligible, which has been also pointed out in other researches, e.g., characterizing the distribution of vehicular inter-connection time [28], [29]. The exponential decay of the entropy indicates that the proportion of APs that have the relatively large traffic-weighted entropy values (i.e., being highly popular) is small. This inspires us to deploy more cache resource at only a small number of APs that have the largest traffic-weighted entropy values, to provide caching services. In addition, the similarity of two CCDF curves also indicates the stationary feature of AP popularity.

3.2.3 Strong Positive Correlation of AP Traffic Consumption and Popularity

We then examine the correlation of AP traffic consumption and its popularity. The correlation could affect the cache deployment strategy design, i.e., caching for more traffic or serving more users. Let $R_k^t(tra)$ and $R_k^t(pop)$ be the set of the first k APs ranked (in the descending order) by the traffic consumption and popularity during the time slot t, respectively. We compute the Jaccard similarity coefficient [30] between $R_k^t(tra)$ and $R_k^t(pop)$ as follows

$$J(R_k^t(tra), R_k^t(pop)) = \frac{|R_k^t(tra) \bigcap R_k^t(pop)|}{|R_k^t(tra) \bigcup R_k^t(pop)|}. \tag{5}$$

Intuitively, higher Jaccard similarity score indicates that there is a large amount of overlap among APs between the two top k AP sets. We calculate the Jaccard similarity score for each day and plot their cumulative distribution function (CDF) results in Fig. 5, where k is set to be 100, 200, and 300,

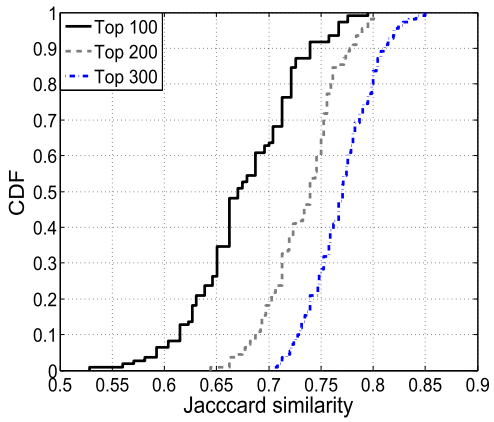


Fig. 5. CDF of Jaccard similarity of AP traffic consumption and popularity.

respectively. We can see that the Jaccard similarity score between the traffic consumption and popularity ranking is rather high. For instance, the median similarity score (with the CDF value of 0.5) is about 0.67, 0.74 and 0.77 when k is 100, 200, and 300, respectively, and the minimum similarity score also reaches up to 0.53, 0.65 and 0.7, respectively. We can conclude that *the traffic consumption of AP is positive correlated to its popularity*. Therefore, when the cache deployment strategy is designed to cache more traffic generated at APs, it is also able to provide the caching services to more users. In addition, as the system provider cares more about the backhaul traffic mitigation, in the remainder of this paper, we then mainly concentrate on mining traffic consumption, to design our cache deployment strategy.

4 Design of LeaD

In this section, we elaborate on the design of LeaD. Particularly, we first depict its architecture and present the work flow. Then, we concentrate on the major technical components of LeaD.

4.1 Overview

Fig. 6 illustrates the work flow and overall architecture of LeaD. As shown in the left part of Fig. 6, with the inputs of system necessary information and available deployment budget, the *CGM* problem can be formulated. By collecting WiFi usage trace and conducting data analytics, LeaD can be adopted to resolve the problem and output the cache deployment results (including both the optimal deployment locations and cache size allocation).

In detail, the right part of Fig. 6 shows the design of LeaD. As deploying cache at large-scale APs (e.g., about 8,000 in our case) is labor- and time-consuming, we first cluster neighboring APs into well-sized edge nodes. In each edge node, the clustered APs can share the cache resource with resource utilization enhancement. Then, we study the edge level traffic consumption over time, and identify that the time series of edge level traffic consumption is stationary after checking the summary statistics and conducting the ADF test. Given the stationary property, the future long-term traffic consumption can be represented by the short-term statistical mean value, and then the *CGM* problem can be transformed to a readily solvable one. We then devise the *TEG* algorithm to achieve the optimal solution of the transformed problem. In the following subsections, we will elaborate on the three technical components.

4.2 Edge Nodes Formation

A straightforward deployment strategy is to deploy cache at each AP, which however may lead to significant deploying cost. Besides, to cover all APs, quite limited caching storages could be allocated to each AP, resulting in inefficient caching performance for users, i.e., with a low caching hit ratio. In contrast, if all caching resource are deployed at the cloud center, the caching hit ratio is optimistic while the end-to-end delay performance is hard to guarantee. Considering the deploying feasibility in realistic conditions, we cluster neighboring APs into well-sized edge nodes according to their physical locations. Normally, when the building is small (e.g., with less than 20 APs), we only need to deploy one edge server in this building to cover all APs within the building by connecting the server to the PoE switches of these APs. For large buildings, which could have more than 100 APs (prevalent in our WiFi system), we divide each floor into distinct edge nodes. Specifically, APs in the same floor are clustered into an edge node, as they are physically in the proximity of each other and suitable for applying

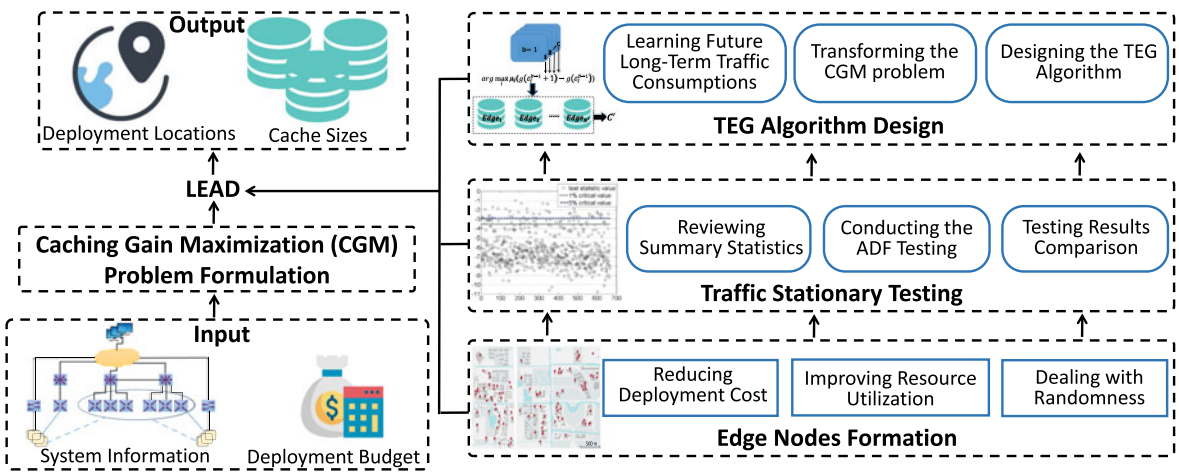


Fig. 6. Architecture of LeaD.

Fig. 7. The physical location of edge nodes.

edge node deployment. Edge nodes formation naturally leads to the following exemplary merits.

1) *Reducing Deployment Cost*. According to the name flag of APs in the system, we partition them into edge nodes by their building and floor distribution, resulting in a total of 667 edge nodes across 201 buildings. Fig. 7 shows the physical location distribution of edge nodes within the system. Note that, we manually collect their Global Positioning System (GPS) locations to label them into the digital map. As the indoor GPS signals are unavailable, all edge nodes in the same building are represented by only one GPS location, i.e., the position of the building. Compared with deploying cache at 7,710 locations, in LEAD, we only need to deploy edge cache at 667 locations at most, which significantly reduces the deployment cost in such a wide area.
2) *Improving Caching Resource Utilization.* Unlike caching at every AP, where only associated users can access the caching resource, we deploy cache at edge nodes such that all users associated to APs (belong to the same edge node) can share the caching resource. It can significantly improves the caching resource utilization. In addition, as the scale of each edge node is small, i.e., with a limited number APs, there is no need to build the specialized cloud center (requiring large space, environmental monitoring, resource management system, etc). Specifically, in Fig. 8, we plot the CDF of the number of APs in edge nodes and buildings, respectively. It can be seen that for the proposed AP clustering strategy, the maximum number of APs in edge nodes is no more than 40, and the median number is about 10. In contrast,

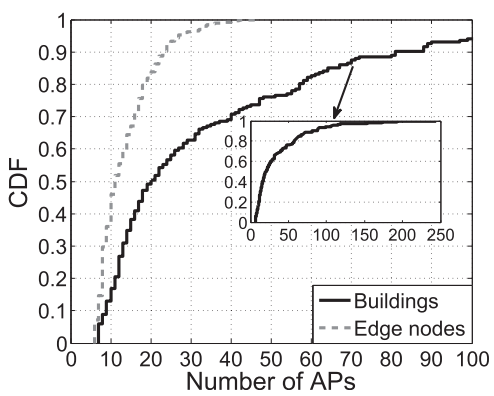


Fig. 8. CDFs of the number of APs in edge nodes and buildings.

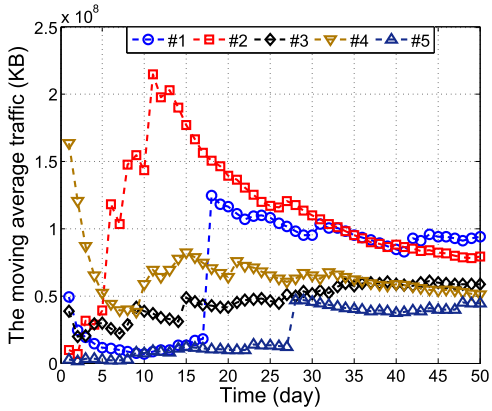


Fig. 9. The empirical mean traffic consumption over days at AP level.

for buildings, the maximum number reaches up to 250, and more than 20 percent buildings have more than 50 APs. Comparing with equipping a data center in each building, the lightweight edge node caching is easy to implement and maintain.

3) *Dealing with Randomness of AP Traffic Consumption.* Another reason for clustering APs is to cope with the randomness of traffic consumption at AP level, which is reasonable since the traffic consumption at a single AP level is limited and thus sensitive to noises. To explain it, given the traffic consumption V_i at each time slot i, we investigate the metric of empirical mean traffic consumption V_{avg}^t, i.e.,

$$V_{avg}^t = \frac{1}{t}\sum_{i=1}^{t} V_i, \tag{6}$$

which indicates the traffic consumption during the long-term duration t. We randomly choose several APs and edge nodes, and plot their empirical mean of traffic consumption over days in Figs 9 and 10, respectively. It can be seen that at AP level, V_{avg}^t can hardly stabilize and oscillate dramatically, especially at the initial and middle stage. On the contrary, at edge level, V_{avg}^t becomes stable rapidly within 15 days at all edge nodes. Fig. 11 further shows the time series of the empirical mean traffic consumption of one hundred randomly-chosen edge nodes, in which the last column (i.e., the 50th column) is sorted and

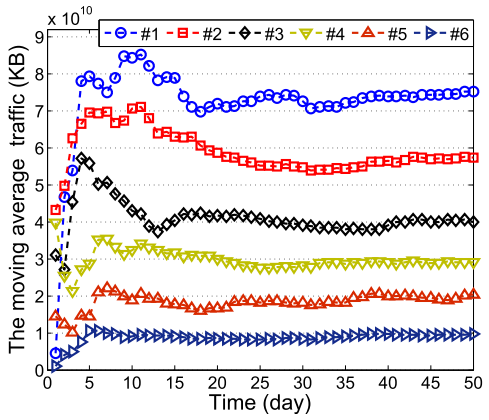


Fig. 10. The empirical mean traffic consumption over days at edge level.

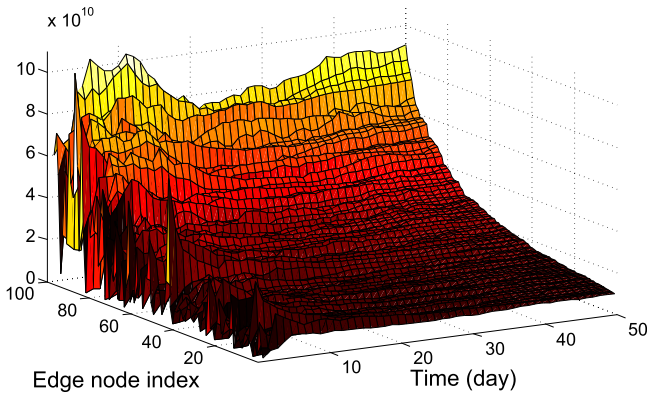


Fig. 11. The empirical mean traffic of one hundred randomly-chosen edge nodes.

placed in descending order. We can also observe that, in each row, the evident color variation mainly appears at the initial stage, and the color keeps stable afterwards. The fast stabilization of the empirical mean traffic consumption at edge level, benefits the traffic consumption modeling.

4.3 Edge Traffic Stationary Testing

Reviewing Summary Statistics. As observed above, the time series of edge level traffic consumption seems to be stationary. To test its underlying stationary condition, we first take a glance at the summary statistics, i.e., the mean and variance variation. We split the time series (on a daily basis) of edge traffic consumption into two partitions, denoted as $X_i' = (V_i^1, V_i^2, \ldots, V_i^l)$ and $X_i'' = (V_i^{l+1}, V_i^{l+2}, \ldots, V_i^{|V_i|})$ for edge node i, where $l = \lfloor \frac{|V_i|}{2} \rfloor$. We compare the mean and variance between X_i' and X_i'', and plot their CDFs of all edge node results in Figs. 12 and 13, respectively. In both figures, it can be seen that the difference of the mean and variance between X_i' and X_i'' are statistically negligible, since both two curves are highly intertwined which almost have the same distribution.

Augmented Dickey-Fuller Testing. The summary statistics results provide the evidence of being stationary, we then adopt the ADF test [31] to explicitly comment on whether the time series of edge traffic consumption is stationary. Generally, the ADF test is a type of unit root test, which tests a null hypothesis that a unit root is present in the time series sample (i.e., it has a certain time-dependent structure and is not stationary), against an alternative hypothesis (i.e.,

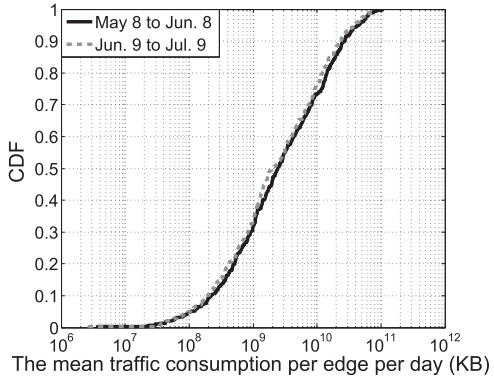


Fig. 12. CDFs of the mean of edge traffic consumption.

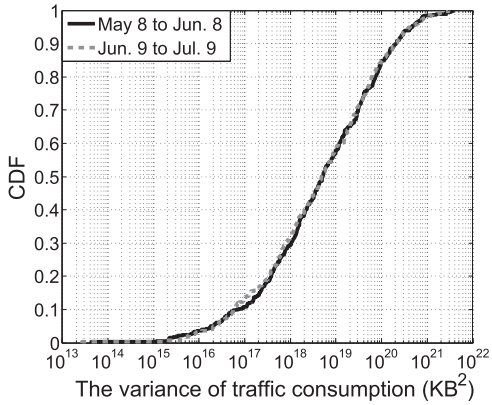


Fig. 13. CDFs of the variance of edge traffic consumption.

rejecting the null hypothesis) that there is no unit root and the time series is stationary. Specifically, we adopt an autoregressive (AR) model to represent the random process of traffic consumption and the testing procedure can be described as follows [31]. We use the AR(p) model:

$$V_t = \sum_{i=1}^{p} \theta_i V_{t-i} + \xi_t, \tag{7}$$

to represent the time series variation, where V_t is the traffic consumption variable, t is the time index, p is the number of lags included in the test, $\theta_1, \ldots, \theta_p$ are the parameters of the model, and $\xi_t \sim N(0, \sigma^2)$ (white noise) is the error term at the time index t [31]. For this model, the characteristic polynomial θ is given by

$$\theta(L) = 1 - \sum_{i=1}^{p} \theta_i L^i, \tag{8}$$

where L is a testing parameter. With a unit root, it means that the characteristic polynomial evaluated in 1 equals to 0, i.e., $\theta(1) = 0$, which is the same as $1 - \sum_{i=1}^{p} \theta_i = 0$. We can test the null hypothesis that there is a unit root, i.e., $H_0 : \sum_{i=1}^{p} \theta_i = 1$, and test it against the alternative hypothesis, i.e., $H_1 : -1 < \sum_{i=1}^{p} \theta_i < 1$. For the original AR($p$) model, we are interested in the difference between V_t and V_{t-1}, and we can rewrite it by subtracting V_{t-1} on both sides, i.e.,

$$\underbrace{V_t - V_{t-1}}_{\Delta V_t} = (\theta_1 - 1)V_{t-1} + \theta_2 V_{t-2} + \cdots + \theta_p V_{t-p} + \xi_t. \tag{9}$$

To express it by the difference of all successive time series, we can transform it by successively adding and subtracting $\theta_p V_{t-p+1}$, $(\theta_{p-1} + \theta_p)V_{t-p+2}, \ldots, (\theta_2 + \cdots + \theta_p)V_{t-1}$ on the right-hand side, i.e.,

$$\begin{aligned} \Delta V_t &= (\theta_1 - 1)V_{t-1} + \theta_2 V_{t-2} + \cdots + \theta_p V_{t-p} + \xi_t \pm \theta_p V_{t-p+1} \\ &= (\theta_1 - 1)V_{t-1} \cdots + (\theta_{p-1} + \theta_p)V_{t-p+1} - \theta_p \Delta V_{t-p+1} + \xi_t \\ &= \cdots \\ &= \gamma V_{t-1} + \delta_1 \Delta V_{t-1} + \cdots + \delta_{p-1} \Delta V_{t-p+1} + \xi_t, \end{aligned} \tag{10}$$

where γ equals $(\theta_1 + \theta_2 + \cdots + \theta_p - 1)$ and $\delta_i = -(\theta_{i+1} + \cdots + \theta_p)$ [31].

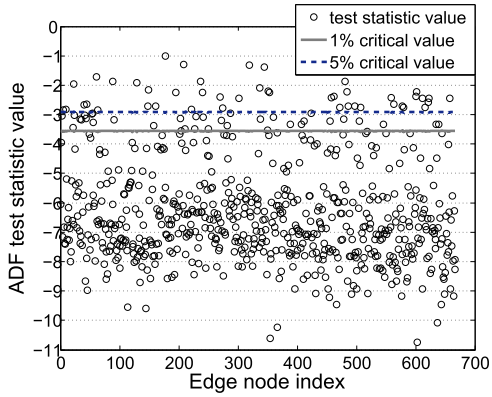


Fig. 14. Comparison between the ADF test statistic and percentile critical value.

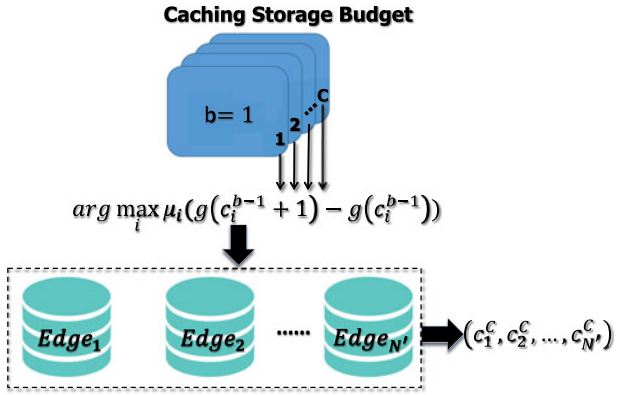


Fig. 16. The work flow of *TEG* algorithm.

By this equivalent representation of the model, we can test the same null hypothesis as $H_0 : \gamma = 0$, against the alternative hypothesis $H_1 : -2 < \gamma < 0$. According to the time series values, we can calculate the value of ADF test statistic, i.e., $\hat{t}$, as

$$\hat{t} = \frac{\hat{\gamma}}{SE(\hat{\gamma})}, \tag{11}$$

where $\hat{}$ means the estimation of the parameter, and $SE(\hat{\gamma})$ is the standard error of the $\hat{\gamma}$.

The testing result can be achieved by comparing the test statistic value with the relevant critical value in the Dickey-Fuller look-up table, which is known ahead. The smaller the test statistic value is, the more likely it is to reject the null hypothesis of $\gamma = 0$ (i.e., the time series is stationary). Fig. 14 shows the comparison between the ADF test statistic value and percentile critical values of traffic consumption series at all edge nodes, where the 1 and 5 percent percent critical values are shown. We can easily observe that the most ADF test statistic values of edge traffic consumption are less than the value around -2.92 at 5 and -3.56 at 1 percent. For the edge node with the ADF test statistic value less than -3.56 at 1 percent, it suggests that we can reject the null hypothesis with a significance level of less than 1 percent, which is a rather low probability that the result is a statistical fluke. We can see that most edge nodes are able to achieve such level of stationary traffic consumption, as their ADF test statistic values are less than -3.56 and could range from -5 to -8. To make it clear, we can simply compute the p-value [32], which means the probability of H_0 being true after observing at least the same ADF test statistic values. If $p\text{-value} \leq 0.05$, the null hypothesis H_0 should be rejected, and the time series does not have a unit root. We plot the CDF of p-value of all edge nodes in Fig. 15. It can be seen that, normally, the p-value of edge nodes are far less than the critical value 0.05, and the median value is only 10^{-8}. In addition, given the critical value of 0.05, about 95 percent of edge nodes can pass the stationary testing. We can conclude that the *time series of edge traffic consumption is statistically stationary*, which is suitable for applying one-shot cache deployment strategy.

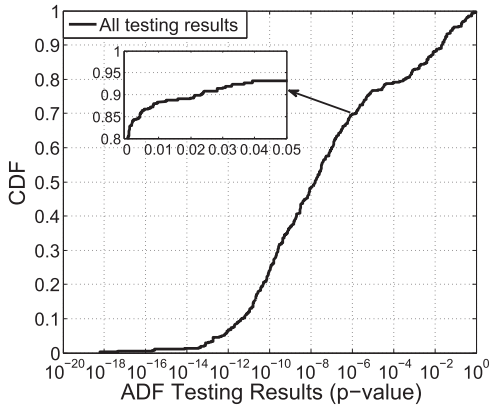


Fig. 15. CDF of p-value.

4.4 Traffic-wEighted Greedy (TEG) Algorithm

Given the stationary property of edge traffic consumption, the expectation of future long-term traffic consumption at the edge node i, i.e., $E[V_i^t]$, can be substituted by its short-term constant mean value μ_i (denote by $\boldsymbol{\mu}$ the set of short-term mean values for all edge nodes). Thus the *CGM* problem can be reformulated as

$$\begin{aligned} &\max_{\{c_i\}} \quad \sum_{i=1}^{N'} \sum_{f=1}^{c_i} \frac{1/f^\alpha}{\sum_{h=1}^{F} 1/h^\alpha} \frac{\mu_i}{s}, \\ &\text{s.t.} \sum_{i=1}^{N'} c_i \leq C, \ 0 \leq c_i \leq F, \end{aligned} \tag{12}$$

where N' (denote by $\boldsymbol{N'}$ the set of edge nodes) is the number of edge nodes. Ignoring the constant $(\sum_{h=1}^{F} 1/h^\alpha) \times s$, the objective function becomes $\max_{\{c_i\}} \sum_{i=1}^{N'} \mu_i \sum_{f=1}^{c_i} 1/f^\alpha$. The goal is to assign the cache size such that the total traffic weighted caching gain can be maximized. We devise the *TEG* algorithm to allocate cache storage iteratively in a greedy way. Specifically, as shown in Fig. 16, given the total cache storage budget C, we first allocate the first unit of cache budget, i.e., $b = 1$. The budget will be assigned to the edge (say the kth edge node) who could achieve the largest caching gain if the first cache budget is deployed there, i.e., $k = \arg\max_i \mu_i(g(c_i^0 + 1) - g(c_i^0))$, where $g(x) = \sum_{f=1}^{x} 1/f^\alpha$, and $c_i^{b-1}(c_i^0 = 0)$ is the cache size of the ith node when allocating the $b-$th unit of cache budget. After that, the cache size of the kth edge node is updated to 1, and we then allocate the second unit of cache budget and assign it to the edge who can achieve the largest caching gain. The procedure continues until there is no unit of budget left, and at

last, we can obtain the allocation result $\boldsymbol{C'}$, i.e., $(c_1^C, c_2^C, \ldots, c_{N'}^C)$. *Algorithm 1* presents the pseudo-code for the *TEG* algorithm. From the pseudo-code, it is easy to observe that the proposed *TEG* algorithm has a polynomial time complexity with $O(N'^2C)$. Considering the deployment of cache is a one-shot solution, the complexity is acceptable. In addition, for the optimization problem in Eq. (12), the *TEG* algorithm is able to achieve the optimal solution, which can be easily proved using contradiction. Specifically, if the solution achieved by the *TEG* algorithm is not optimal, there must exist a unit cache budget that is allocated to a different edge node with a larger caching gain. However, it violates the design of *TEG* that the unit cache budget would be allocated to the edge who can achieve the largest caching gain after owning the budget.

Algorithm 1. TEG Algorithm

Input: $\boldsymbol{N'}$, $\boldsymbol{\mu}$, C, α
Output: $\boldsymbol{C'} = (c_1^C, c_2^C, \ldots, c_{N'}^C)$
Initialize: $c_1^0 = 0, c_2^0 = 0, \ldots, c_{N'}^0 = 0$
Initialize: $w_1 = 0, w_2 = 0, \ldots, w_{N'} = 0$
for b in $[1, C]$ **do**
  for i in $\boldsymbol{N'}$ **do**
    $w_i = \mu_i \cdot (g(c_i^{b-1} + 1) - g(c_i^{b-1}))$
  end for
  $k \leftarrow \arg\ \max\ w_i$
  for i in $\boldsymbol{N'}$ **do**
    if $i == k$ **then**
      $c_i^b = c_i^{b-1} + 1$
    else
      $c_i^b = c_i^{b-1}$
    end if
  end for
end for

function gx
  Initialize: $\text{gain} = 0$
  for f in $[1, x]$ **do**
    $\text{gain} = \text{gain} + 1/f^{\alpha}$
  end for
  return gain
end function

5 Performance Evaluation

In this section, we conduct extensive trace-driven experiments to evaluate the performance of LeaD. Particularly, we first elaborate on the evaluation methodology with experiment setup, benchmark strategies design, and metrics definition, then carry out the overall performance comparison, and last investigate the impact of the number of files and caching budget as well as the impact of Zipf skewness parameter. As the goal of LeaD is to achieve the long-term caching gain maximization where the robustness of the algorithm in time dimension is rather important, we further collect up-to-date trace and conduct new experiments to evaluate the temporal robustness of LeaD.

5.1 Methodology

Experiment Setup. As indicated in the previous section, the future long-term traffic consumption can be well learned with a short-term traffic statistics. Therefore, we use the traces from May 8, 2018 to Jun. 8, 2018 as the training data set to learn the value of $\boldsymbol{\mu}$, and then adopt the traces from Jun. 9, 2018 to Jul. 9, 2018 as the testing data set for performance evaluation. We set the size of each file s to 1 GB. The number of files F is ranged from 1,000 to 5,500 with a step size of 500, and the total cache budget C is ranged from 20 to 600 TB. In addition, the Zipf skewness parameter α is set to be 0.56, which is normally adopted for video file popularity modeling [14].

Benchmark Strategies. To compare with the performance of our proposed LeaD, the following three reasonable benchmark strategies are also designed:

- *Oracle:* In this strategy, we assume that traffic consumptions in the testing data are known ahead, and we solve the *CGM* problem to obtain the optimal result. In reality, this strategy cannot be achieved, and we take it as the upper bound reference.
- *Equipartition:* This strategy would be often adopted in practice by system providers, in which there is no prior information on the network traffic and the total cache storages are equally divided to each edge node.
- *Demographics:* In this strategy, we mimic the demographic approach, i.e., allocating the cache budget according to the user density, which is also commonly used in practice. To this end, we use the number of edge association users in one day to measure the user density, based on which the strategy allocates the cache budget proportionally.

Performance Metrics. The following two metrics are defined to evaluate the caching deployment performance.

- *Caching gain ratio:* refers to the average performance gain in deploying large-scale cache, calculated by dividing the successfully cached traffic to the total consumption traffic.
- *Caching resource utility:* refers to the daily successfully cached traffic divided by the cache size. It indicates how much traffic that a unit cache resource can directly provision per day, i.e., the effectiveness of the unit cache resource.

Note that, to ensure the comparison fairness, all strategies adopt the same edge nodes formation strategy as in LeaD. In addition, as this paper investigates the cache deployment rather than the content delivery [33], the delay performance is not assessed, and how to optimize the content delivery is out of the scope of this paper. Yet, those problems can be studied upon LeaD to collectively enhance the overall system performance.

5.2 Overall Performance Comparison

We first carry out the overall performance comparison between LeaD and other benchmark strategies. We check the experiments results when F and C are set to be 5,000 and 100 TB, respectively.[4]

Efficient Caching Resource Allocation. Fig. 17 shows CDFs of the cache size allocations by different strategies, and we have the following two major observations. First, both LeaD and the *Oracle* strategy can assign the caching resource according

4. Similar observations can also be achieved when varying the values of F and C.

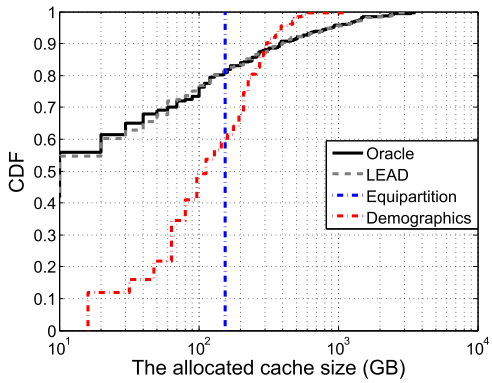


Fig. 17. CDFs of cache size allocations.

to the underlying traffic conditions. Specifically, it can be seen that within the proportion between 90 and 100 percent, the allocated cache sizes are relatively large, which are larger than that allocated by the other two strategies. It fits our data analytics that the proportion has a fast decay when the traffic consumption becomes extremely large (i.e., more caching resource should be allocated to them). Second, we can see that in both LEAD and the *Oracle* strategy, only about 45 percent of edge nodes have been allocated with cache resources, and need to deploy edge caching servers there, reducing much deploying time and labor.

Fig. 18 shows the caching gain ratio over days by different strategies, and it can be easily seen that the proposed LEAD can achieve a quite close performance to the *Oracle* strategy, as two plots are tightly intertwined. In addition, although the LEAD performance oscillates within a small range with time evolving, it can outperform both the *Equipartition* and *Demographics* strategy significantly. In addition, we plot the average caching gain ratio in Fig. 19. We can see that compared with the *Oracle* strategy with the average ratio about 0.35, the average ratio of LEAD can reach 0.346, which is a negligible degradation. However, compared with the *Equipartition* and *Demographics* strategy with the respective average ratio of 0.202 and 0.25, LEAD can improve the performance by 71 and 38 percent, respectively. Note that, more obvious daily variation in LEAD and the *Oracle* strategy (shown by error bars) is mainly caused by the traffic-consumption oscillation of those small-portion edge nodes, who have relatively large traffic consumption and have been assigned with relatively more caching resources. Their traffic-consumption oscillations could affect the

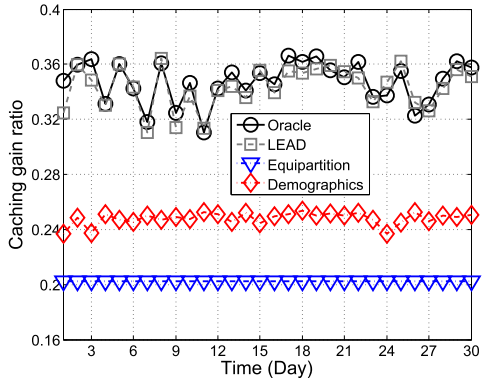


Fig. 18. The caching gain ratio versus time.

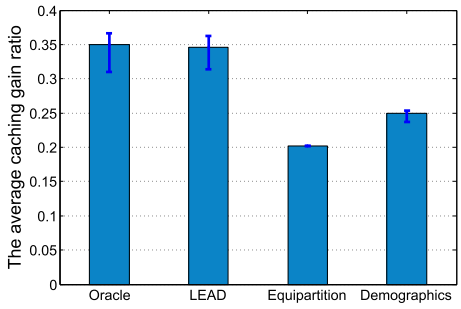
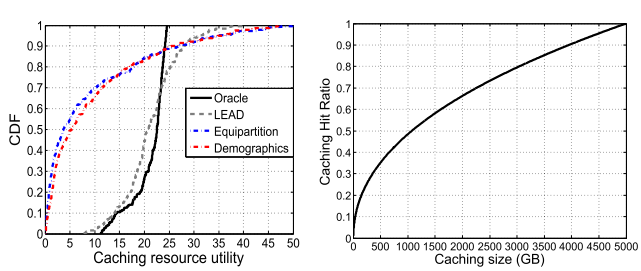


Fig. 19. The average caching gain ratio.

caching performance a lot, but the overall performance gain of both strategies are still significant.

Submodularity Hurts. We then check the caching resource utility and Fig. 20a shows CDFs of caching resource utilities of edge nodes by four strategies. It can be observed that both LEAD and the *Oracle* strategy can outperform the *Equipartition* and *Demographics* strategy dramatically in general. For instance, in the *Equipartition* and *Demographics* strategy, the median utility is about 4 GB and 5 GB per unit per day, respectively, while in LEAD and the *Oracle* strategy, the value increases dramatically to 20.5 GB and 22 GB per unit per day, respectively. It means that with deploying 1 GB cache storage, on average, the proposed LEAD can reduce more than 20.5 GB backhual traffic per day. It is increased by respective 412 and 310 percent when comparing to the *Equipartition* and *Demographics* strategy. However, it can also be observed that when the proportion comes to between 90 and 100 percent, the caching utility in the *Oracle* strategy becomes the lowest, while the *Equipartition* and *Demographics* strategy can achieve a better performance.

To figure out the underlying rationale, as shown in Fig. 20b, we plot the caching hit ratio in the Zipf distribution when varying the cache size from 1 to 5,000 GB (the total number of files is fixed to 5,000). It can be seen that, the caching hit ratio has an obvious submodular property. Specifically, as the cache size increases, the difference in the incremental ratio that a unit cache storage makes, decreases accordingly. For example, with deploying 1,000 GB cache storage, the caching hit ratio reaches about 0.5, while the ratio only increases to 0.65, 0.8 and 0.9 when deploys 2,000, 3,000, and 4,000 GB, respectively. Therefore, in Fig. 20a, for those edges who have extremely large traffic consumption, LEAD and the *Oracle* strategy would allocate them quite more cache storages in order to enhance the caching gain,



(a) CDFs of caching resource utilities of edge nodes (b) caching hit ratio vs. the cache size in the Zipf distribution

Fig. 20. Caching resource utility.

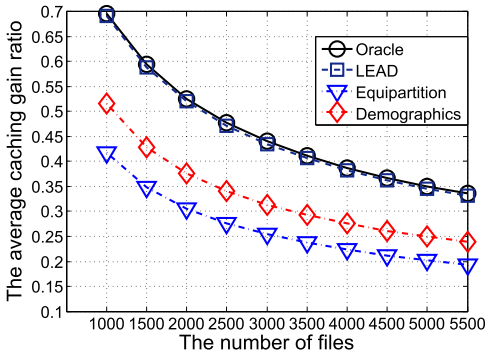


Fig. 21. The average caching gain ratio versus the number of files.

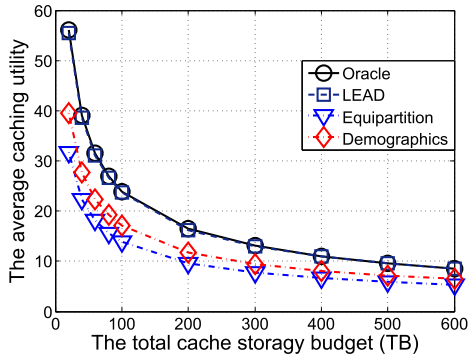


Fig. 23. The average caching resource utility versus the caching budget.

which however pulls the average caching utility down. On the contrary, in the *Equipartition* and *Demographics* strategy, less cache storages are allocated there, resulting in larger caching utilities.

5.3 Impact of the Number of Files and Caching Budget

With the overall performance guaranteed, we then investigate the impact of number of files F and caching budget C on the caching performance.

We start by varying the number of files from 1,000 to 5,500 with the step size of 500 while fixing the total caching budget to 100 TB. As shown in Fig. 21, we plot the average caching gain ratio, and can achieve the following three major observations. First, the proposed LEAD can reach the approximately optimal performance as two curves are quite close to each other. Second, regardless of the files variation, LEAD always outperforms other two candidate strategies with an obvious performance gap. Third, apart from the caching deploying strategy, the caching content replacement strategy is also critical for caching performance, as the number of files has a deep impact on the caching gain ratio. For example, in LEAD, when the number of files increases from 1,000 to 5,000, the ratio can decrease from 0.7 to 0.34. Therefore, to achieve an outstanding caching system, it is also of importance to design an efficient content replacement strategy, to well predict the file request, which however is out of the scope of this paper.

As shown in Fig. 22, we then plot the average caching gain ratio by ranging the caching budget from 20 to 600 TB while fixing the number of files to 5,000. Similar observations such as the superior and close-to-optimal performance of LEAD can also be observed. What should be pointed out is that with deploying more cache storages, although the average caching gain ratio increases, it is also sustained by the *submodular property* that caused by the caching resource utility degradation. As shown in Fig. 23, the average caching resource utility decreases dramatically as the total caching budget increases, especially at the initial stage. For instance, the utility decreases from 57 GB to 24 GB, 17 GB, 10 GB, and 9 GB per unit per day when the total budget increases from 20 TB to 100 TB, 200 TB, 500 TB, and 600 TB, respectively. These results provide insightful principles for system providers to customize their deploying plan when considering the deployment budget. For example, by deploying less cache storages (e.g., less than 100 TB), it is easy to achieve an immediate caching performance with relatively large caching resource utilities. However, to achieve a quite impressive performance, it means a large number of cache storages should be deployed as the caching resource utility decreases significantly.

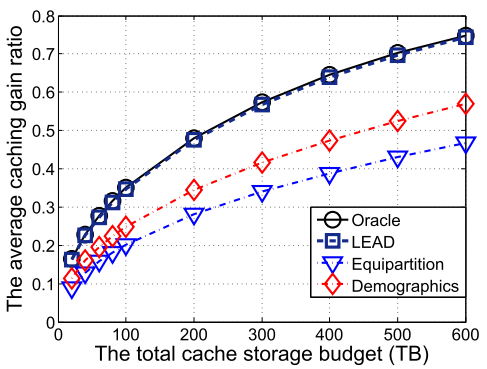
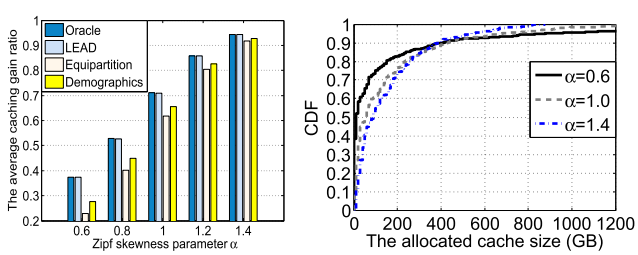


Fig. 22. The average caching gain ratio versus the caching budget.

5.4 Impact of the Zipf Skewness Parameter

In this subsection, we investigate the impact of Zipf Skewness parameter α, which is crucial for the caching system, and may vary with time [34], [35]. Specifically, we vary the value of α from 0.6 to 1.4 with a step size of 0.2, and fix the value of F and C to 5,000 and 100 TB, respectively.

Fig. 24a shows the average caching gain ratio achieved by strategies when varying the value of α, demonstrating three significant observations. First, under all conditions, LEAD can achieve a quite close performance to the *Oracle* strategy, both of which can outperform the other two benchmark strategies significantly. Second, for all strategies, the average caching gain ratio can increase rapidly with the skewness parameter. It is reasonable as a larger α leads to less uncertainties of content requests, and can result in a



(a) The average caching gain ratio vs. α (b) CDFs of cache size allocations vs. α

Fig. 24. Impact of Zipf skewness parameter.

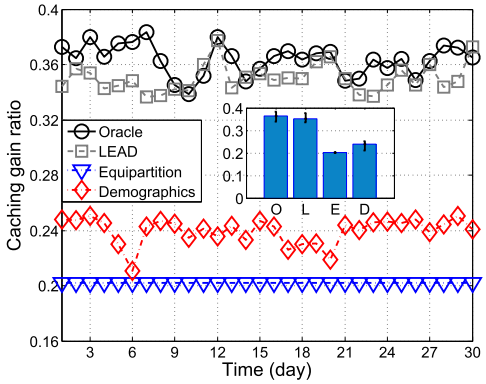


Fig. 25. The caching gain ratio in each day.

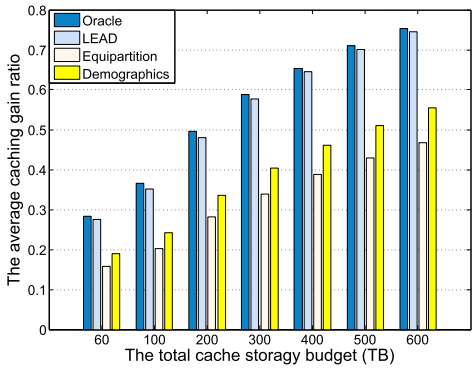


Fig. 27. The average caching gain ratio versus storage budget.

larger content hit ratio. Third, with the increase of the value of α, the performance gap between LEAD and the other two benchmark strategies shrinks. It happens because the larger α brings a better content hit ratio, which can make up the weakness of cache deployment of the *Equipartition* and *Demographics* strategy.

Fig. 24b shows the CDFs of cache size allocation for all edge nodes (achieved by LEAD), when setting the value of α to 0.6, 1.0, and 1.4, respectively. We can observe that, with a small α (e.g., 0.6), the deployment strategy would allocate the cache storages to a small proportion of edge nodes (i.e., popular nodes) with relatively skewed cache sizes. In contrast, with a large α (e.g., 1.4), the cache storages would be allocated to most edge nodes with evenly distributed cache sizes, as all of them are able to achieve considerable caching gain after deploying cache there. Specifically, when the value of α is set to respective 0.6, 1.0, and 1.4, about 39, 18, and 10 percent of edge nodes will not be allocated cache storages, which are free from the cache deployment. In addition, with $\alpha = 1.4$, the maximum allocated cache storage is limited within 850 GB, but the value can reach over 1,200 GB, and 2,000 GB when decreasing α to 1, and 0.6, respectively. It indicates that the uncertainties of content request can affect the underlying cache deployment strategy a lot. In addition, a more skewed parameter α can inversely lead to a more smooth distribution of cache size allocation, while a small α can lead to a skewed distribution.

5.5 Temporal Robustness Experiments

To validate the long-term efficacy of LEAD, we collect the new one-month user association trace from Apr. 26, 2019 to May 28, 2019, with over 23,115,573 association records. The new trace is up-to-date (in terms of when we conduct this experiment) and takes almost one-year interval after our methodology design, which is adequate to evaluate the temporal robustness of LEAD. Except for the *Oracle* strategy, it obtains the cache deploying decisions by solving the *CGM* problem based on the new trace. For other strategies, we adopt the cache deploying decisions that achieved in above experiments (i.e., using the training data set of May 8, 2018 to Jun. 8, 2018 to solve the *CGM* problem) to evaluate the performance of new trace.

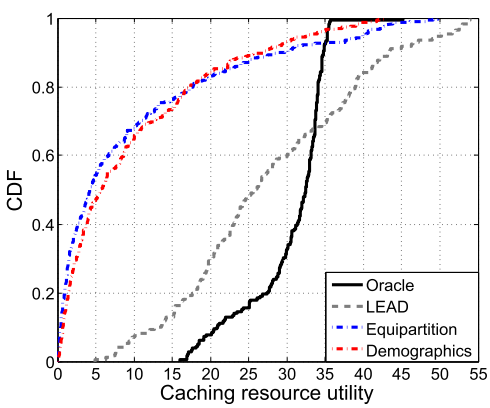


Fig. 26. CDFs of average caching resource utility of edge nodes.

Fig. 25 shows the daily caching gain ratio by four strategies when F and C are 5,000 and 100 TB, respectively, and we can observe that LEAD is still able to guarantee the prosperous performance. Particularly, LEAD can achieve a quite close performance to that of the *Oracle* strategy, and outperforms the *Equipartition* and *Demographics* strategy significantly. Even though, compared with Fig. 18, the gap between LEAD and the *Oracle* strategy slightly increases, which is reasonable as the correlation of time-series traffic consumption decays slightly after a long time interval, the one-month averaged caching gain ratio of LEAD is still rather considerable. Specifically, the *Oracle* strategy guarantee 0.363 of average caching gain ratio, while LEAD obtains 0.353 of average caching gain ratio, with only 2.7 percent of performance degradation. However, compared with the *Equipartition* and *Demographics* strategy with 0.202 and 0.242 of average caching gain ratio, LEAD can improve the performance by 74.8 and 45.8 percent, respectively. Fig. 26 shows the CDFs of average caching resource utility of edge nodes by four strategies, and similar observations (compared with Fig. 20a) can be observed. Differently, more obvious advantage of LEAD and the *Oracle* strategy shows up, which happens due to more active Internet activities by users in the new trace. Fig. 27 shows the one-month averaged caching gain ratio by four strategies when varying the storage budget. Likewise, with enlarged cache budgets, the caching performance improves with submodularity, and LEAD is able to achieve the near optimal performance.

To summarize, LEAD has a strong temporal robustness due to the stationary property of daily traffic consumption. Even though after a long interval, the gap between LEAD and the *Oracle* strategy may slightly increase, enlarging the training data set could further reduce the gap. However, the gap is very limited (no more than 3 percent), and one-month training trace is enough and suggested, since a quite outstanding performance can be guaranteed.

6 Related Work

We review the related work in two categories, i.e., mobile edge caching and data analytics for caching.

6.1 Mobile Edge Caching

Since high-bandwidth file requests are popular among mobile users nowadays, mobile edge caching has attracted a lot of research attention recently, which is a prominent technique to alleviate the backhual burden and deliver fast services to users. Bastug *et al.* [15] explained the basic concepts of proactive edge caching in 5G networks, where predictive capabilities and developments in storage, social networks, and context awareness, are advocated to be harnessed for caching performance enhancement. Yao *et al.* [36] organized a mobile edge caching survey, in which after presenting an overview of mobile edge caching, authors then introduced caching locations, summarized caching metrics, reviewed caching schemes, and then mainly focused on caching content methodologies including request analysis, content exploration, delivery, and replacement.

Therefore, most current technical works concentrate on caching content placement since it becomes tricky when taking the user request diversity, local file popularity, user mobility, channel quality, link connectivity, etc., into consideration. Particularly, Ma *et al.* [16] investigated the caching problem at a large number of APs, to determine the content placements, i.e., which APs should cache which contents. By considering the local content popularity at each AP, they formulated an optimization problem to maximize the total cache hit rate of all APs. After proving its NP-hardness, a local distributed caching algorithm is then devised to solve the problem. Also to maximize the total content hit rate, Yang *et al.* [14] designed a linear model to estimate the future content popularity, and proposed a location-aware caching scheme after observing that users at different locations have unique preferable file lists. Moreover, Vasilakos *et al.* [37] leveraged the joint consideration of user mobility prediction and content popularity information, to further enhance the caching performance. Other technical papers on video popularity prediction could be found in [23], [24], [25]. Given the limited cache storages, to cache the most popular contents, Cao *et al.* [17] proposed an auction mechanism from the perspective of system providers. The mechanism can elicit true valuations of contents from the users (incentive compatibility), and incentivize user participation (individual rationality). With the mechanism, the cache size allocation can be determined for different contents. Regrading to wireless resource allocation (e.g., bandwidth, transmission power) for content delivery, there are two recent cooperative caching proposals, i.e., [9] and [18], in dense BS scenarios where users can be covered by multiple BSs simultaneously. In specific, as covered by multiple BSs (enlarging the cached content set), the bandwidth utilization may degrade when users are served by further BSs. To strike the tradeoff, Zhang *et al.* [9] studied the joint optimization of BS clustering, content placement, and bandwidth allocation, the goal of which is to minimize the average content transmission delay. Without direct communications among BSs or a priori knowledge of content popularity, Leonardi *et al.* [18] focused on cache coordination scheme design, and proposed a class of fully distributed schemes to maximize the overall hit ratio. Recently, UAVs are also investigated, which are employed as flying edges for caching service provision [19], [20], since they have a high agility with low cost. For example, in order to seek suitable user-UAV associations, Chen *et al.* [19] tried to predict the content request distribution by leveraging the human-centric information, such as gender, requested contents, visited locations, etc. In [20], UAVs are adopted to provide services for moving vehicles and how to make offloading decisions is investigated.

On the other hand, Vigneri *et al.* [10] investigated caching at relays, e.g., mobile vehicles, to achieve low cost video streaming, as users can prefetch video chunks from encountered vehicles at low cost or stream from the cellular infrastructure at high cost. They modeled the playback buffer of user and analyzed its idle periods during which data should be downloaded from the infrastructure, and then optimized the content allocation to mobile caches to minimize the expected amount of celluar-downloading data. Likewise, Wu *et al.* [38] also investigated content caching at users and BSs where users can retrieve the requested content from neighboring users via device-to-device links or the neighboring BSs via cellular links. Differently, in the work [39], Somuyiwa *et al.* tried to minimize the long-term average energy cost of the content delivering to users when conduct the proactive caching in wireless networks.

The limitation is that these works research on the content placement/replacement with a common assumption that caching points are determined and cache storages are given ahead, but the investigation on cache deployment is rarely seen in the field. In this paper, instead of designing the content placement/replacement strategy, we concentrate on the cache deployment, which is an essential building block for an efficient caching system. Although there have been some cache size allocation works, e.g., [40], [41], they are applied in content-centric networks. The system model is quite different, since the end-to-end networking architecture is broken and caching servers can cooperate with each other. Besides, the objective function is also different, which is to minimize the number of hops to fetch the content and cannot be applied in our case.

6.2 Data Analytics for Caching

In the work [12], Ma *et al.* studied the two-week real-world data set which contains video (more than 0.3 million unique videos) viewing records by 2 million users, to investigate the user request patterns and their behaviors in mobile video streaming, and then disclosed that user request shows spatio-temporal variation, i.e., users within different locations at different time would request for different contents. These insights benefit content placement and update while neglecting cache deployment strategies. In another data analytics work, Syamkumar *et al.* [42] used a dataset of over 4M cell tower locations in the US, and presented an empirical study of key aspects of cell tower infrastructure to understand their current characteristics and identify future edge deployments. Specifically, they evaluated the geographic characteristics of cell towers and highlight how locations correspond to population density in major metropolitan areas and in rural areas. They focused on the edge

deployment rather than cache deployment, and the cellular scenario is different from our corporate WiFi system. Likewise, Li *et al.* [43] also studied the location distribution of 3,233 BSs in Shanghai city, to investigate the edge deployment where they tried to strike the tradeoff between the coverage holes and energy consumption of edge servers. Considering the important role of big data in caching system, some magazine papers [21], [22], [44], [45] have advocated the combination of big data and machine learning to facilitate the caching performance, while they mainly focus on content request analysis and content popularity prediction, and give high-level guidelines without detailing the in-depth techniques.

In summary, all the mentioned data analytics works do not investigate user association and traffic consumption in the network, which might fail to direct an informed caching deployment. As far as we know, in the literature, there is little work on mining user networking patterns, to benefit caching deployment and cache size allocation, while our work closes this gap by carrying out spatio-temporal analysis of user association and traffic consumption, and designing LeaD to maximize the long-term caching gain. In our previous work [46], we have demonstrated the efficacy of WiFi traffic statistics for edge cache deployment. In this work, we further improve it by presenting the complete architecture design, giving the comprehensive stationary testing on edge traffic consumption with mathematical proof and experiment validation, evaluating the temporal robustness of LeaD with the new collected trace, and supplementing new experiments. In addition, we have surveyed the up-to-date research works and re-organized the related work to well motivate our work.

7 Conclusion and Future Works

In this paper, we have investigated the edge cache deployment in large-scale WiFi system, which is an essential building block for future WiFi caching system. As the formulated *CGM* problem is intractable directly without knowing future traffic conditions, we have conducted spatio-temporal data analytics on WiFi traffic to study the problem, based on which we have proposed a cache deployment strategy, named LeaD. In LeaD, we first cluster neighboring APs into well-sized edge nodes, then conduct the stationary testing on edge traffic consumption and disclose its stationary property, and finally devise the *TEG* algorithm to solve the *CGM* problem. At last, we have conducted extensive trace-driven experiments to demonstrate the efficacy as well as the temporal robustness of the proposed LeaD strategy. For our future work, in addition to the cache deployment, we will also research on the caching content replacement strategy by mining user association and request patterns.

Acknowledgments

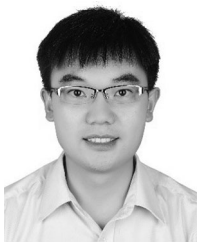
This work was supported in part by the National Key R&D Program of China under Grant No. 2019YFA0706403, National Natural Science Foundation of China under Grant No. 91638204, 61702562, and U19A2067, 111 project (No. B18059), the Young Title Scientists Sponsorship Program by CAST under Grant No. 2018QNRC001, the Young Talents Plan of Hunan Province of China under Grant No. 2019RS2001, and Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] Z. Song, L. Shangguan, and K. Jamieson, "Wi-Fi goes to town: Rapid picocell switching for wireless transit networks," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 322–334.
[2] L. Fang, G. Xue, F. Lyu, H. Sheng, F. Zou, and M. Li, "Intelligent large-scale AP control with remarkable energy saving in campus WiFi system," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst.*, 2018, pp. 69–76.
[3] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 461–474, Feb. 2018.
[4] W. Xu, W. Shi, F. Lyu, H. Zhou, N. Cheng, and X. Shen, "Throughput analysis of vehicular internet access via roadside WiFi hotspot," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3980–3991, Apr. 2019.
[5] F. Mehmeti and T. Spyropoulos, "Performance analysis of mobile data offloading in heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 482–497, Feb. 2017.
[6] D. Gong and Y. Yang, "On-line AP association algorithms for 802.11n WLANs with heterogeneous clients," *IEEE Trans. Comput.*, vol. 63, no. 11, pp. 2772–2786, Nov. 2014.
[7] Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022, White Paper," 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
[8] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, Oct. 2019, Art. no. 125.
[9] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.
[10] L. Vigneri, T. Spyropoulos, and C. Barakat, "Low cost video streaming through mobile edge caching: Modelling and optimization," *IEEE Trans. Mobile Comput.*, vol. 18, no. 6, pp. 1302–1315, Jun. 2019.
[11] T. X. Tran, D. V. Le, G. Yue, and D. Pompili, "Cooperative hierarchical caching and request scheduling in a cloud radio access network," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2729–2743, Dec. 2018.
[12] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1076–1089, May 2017.
[13] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the Edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Aug. 2017.
[14] 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.
[15] E. Bastug, M. Bennis, and M. Debbah, "Living on the Edge: The role of proactive caching in 5G Wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.
[16] G. Ma, Z. Wang, J. Ye, and W. Zhu, "Wireless caching in large-scale Edge access points: A local distributed approach," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netwo.*, 2018, pp. 726–728.
[17] X. Cao, J. Zhang, and H. V. Poor, "An optimal auction mechanism for mobile Edge caching," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 388–399.
[18] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1276–1285, Jun. 2018.
[19] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the Sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.
[20] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[21] E. Zeydan *et al.*, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Magazine*, vol. 54, no. 9, pp. 36–42, Sep. 2016.
[22] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the Big Data era," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 28–35, Jun. 2018.
[23] J. Xu, M. van der Schaar, J. Liu, and H. Li, "Forecasting popularity of videos using social media," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 2, pp. 330–343, Mar. 2015.
[24] Z. Li, G. Xie, J. Lin, Y. Jin, M. Kaafar, and K. Salamatian, "On the geographic patterns of a large-scale mobile video-on-demand system," in *Proc. EEE Conf. Comput. Commun.*, 2014, pp. 397–405.
[25] T. Trzciński and P. Rokita, "Predicting popularity of online videos using support vector regression," *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2561–2570, Nov. 2017.
[26] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh, "Bridging the gap between physical location and online social networks," in *Proc. 12th ACM Int. Conf. Ubiquitous Comput.*, 2010, pp. 119–128.
[27] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1082–1090.
[28] H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. M. Ni, "Recognizing exponential inter-contact time in VANETs," in *Proc. IEEE IEEE INFOCOM*, 2010, pp. 1–5.
[29] F. Lyu et al., "Characterizing urban vehicle-to-vehicle communications for reliable safety applications," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2019.2920813.
[30] Z. Liu, Y. Shen, and Y. Zhu, "Where will dockless shared bikes be stacked?-parking hotspots detection in a new city," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 566–575.
[31] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *J. Amer. Statist. Assoc.*, vol. 74, no. 366a, pp. 427–431, 1979.
[32] J. G. MacKinnon, "Approximate asymptotic distribution functions for unit-root and cointegration tests," *J. Business Econ. Statist.*, vol. 12, no. 2, pp. 167–176, 1994.
[33] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, 2012, pp. 1107–1115.
[34] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Caching-aware recommendations: Nudging user preferences towards better caching performance," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.
[35] N. Carlsson and D. Eager, "Ephemeral content popularity at the edge and implications for on-demand caching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1621–1634, Jun. 2017.
[36] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, Third Quarter 2019.
[37] X. Vasilakos, V. A. Siris, and G. C. Polyzos, "Addressing niche demand based on joint mobility prediction and content popularity cachings," *Comput. Netw.*, vol. 110, pp. 306–323, 2016.
[38] W. Wu, N. Zhang, N. Cheng, Y. Tang, K. Aldubaikhy, and X. Shen, "Beef up mmWave dense cellular networks with D2D-assisted cooperative edge caching," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3890–3904, Apr. 2019.
[39] S. O. Somuyiwa, A. Gyorgy, and D. Gunduz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.
[40] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *Proc. 21st IEEE Int. Conf. Netw. Protocols*, 2013, pp. 1–10.
[41] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *Proc. IEEE INFOCOM Workshops*, 2012, pp. 280–285.
[42] M. Syamkumar, P. Barford, and R. Durairajan, "Deployment characteristics of "The Edge" in mobile edge computing," in *Proc. Workshop Mobile Edge Commun.*, 2018, pp. 43–49.
[43] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput.*, 2018, pp. 66–73.
[44] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov./Dec. 2018.
[45] M. Zeng *et al.*, "Temporal-spatial mobile application usage understanding and popularity prediction for edge caching," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 36–42, Jun. 2018.
[46] F. Lyu *et al.*, "Demystifying traffic statistics for edge cache deployment in large-scale WiFi system," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 965–975.

Feng Lyu (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018, and the BS degree in software engineering from Central South University, Changsha, China, in 2013. He is currently a professor with the School of Computer Science and Engineering, Central South University, Changsha, China. During respective September 2018–December 2019 and October 2016–October 2017, he worked as a postdoctoral fellow and was a visiting PhD student in BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include mobile networking and computing, big data driven application design, could/edge computing, and space-air-ground integrated network. He is a member of the IEEE Computer Society, Communication Society, and Vehicular Technology Society.



Ju Ren (Member, IEEE) received the BSc, MSc, and PhD degrees in computer science from Central South University, China, in 2009, 2012, and 2016. During 2013-2015, he was a visiting PhD student in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Currently, he is a professor with the School of Computer Science and Engineering, Central South University, China. His research interests include Internet-of-Things, wireless communication, network computing and cloud computing. He is the recipient of the Best Paper Award of IEEE IoP 2018 and IEEE ICC 2019, as well as the Most Popular Paper Award (2015-2018) of the Chinese Journal of Electronics. He currently serves/served as an associate editor for the *IEEE Transactions on Vehicular Technology* and the *Peer-to-Peer Networking and Applications*, and a TPC member of many international conferences including IEEE INFOCOM19/18, Globecom17, WCNC17, WCSP16, etc. He also served as the TPC chair for IEEE BigDataSE'19, a poster co-chair for IEEE MASS'18, a track co-chair for IEEE ICCC'19, I-SPAN'18 and IEEE VTC17 Fall, and an active reviewer for more than 20 international journals.

Nan Cheng (Member, IEEE) received the BE and MS degrees from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, in 2016. He is currently a professor with the School of Telecommunication Engineering, Xidian University, Shaanxi, China. He worked as a post-doctoral fellow with the Department of Electrical and Computer Engineering, University of Toronto and Department of Electrical and Computer Engineering, University of Waterloo, from 2017 to 2018. His current research focuses on space-air-ground integrated system, Big Data in vehicular networks, and self-driving system. His research interests also include performance analysis, MAC, opportunistic communication, and application of AI for vehicular networks.

Peng Yang (Member, IEEE) received the BE degree in communication engineering and PhD degree in information and communication engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2013 and 2018, respectively. He was with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, as a Visiting PhD Student from September 2015 to September 2017, and a postdoctoral fellow from September 2018 to December 2019. Since Jan. 2020, he has been a faculty member with the School of Electronic Information and Communications, HUST. His current research interests include mobile edge computing, video streaming, and analytics.

Minglu Li (Senior Member, IEEE) received the PhD degree in computer software from Shanghai Jiao Tong University, in 1996. He is currently a full professor and the director of Network Computing Center at Shanghai Jiao Tong University. He has published more than 350 papers in academic journals and international conferences. He was the chairman of Technical Committee on Services Computing (TCSVC) (2004-2016) and Technical Committee on Distributed Processing (TCDP) (2005-2017), of IEEE Computer Society in Great China region. He served as a general co-chair of IEEE SCC, IEEE CCGrid, IEEE ICPADS, and IEEE IPDPS, and a vice chair of IEEE INFOCOM. He also served as a PC member of more than 50 international conferences including IEEE INFOCOM 2009-2016, IEEE CCGrid 2008, etc. His research interests include vehicular networks, Big Data, cloud computing, grid computing, and wireless sensor networks.

Yaoxue Zhang (Senior Member, IEEE) received the BS degree from the Northwest Institute of Telecommunication Engineering, China, in 1982, and the PhD degree in computer networking from Tohoku University, Japan, in 1989. Currently, he is a professor with the School of Computer Science and Engineering, Central South University, China, and a professor with the Department of Computer Science and Technology at Tsinghua University, China. His research interests include computer networking, operating systems, ubiquitous/pervasive computing, transparent computing, and Big Data. He has published more than 200 technical papers in international journals and conferences, as well as nine monographs and textbooks. He is a fellow of the Chinese Academy of Engineering, China, and the editor-in-chief of the Chinese Journal of Electronics.

Xuemin (Sherman) Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, in 1990. He is currently a university professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, social networks, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering Foreign fellow, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award in 2019 from IEEE, Canada, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and Outstanding Performance Award five times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee chair/co-chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the editor-in-chief of the *IEEE Internet of Things Journal* and the vice president on Publications of the IEEE Communications Society.