Learning-Aided Collaborative Caching in Small Cell Networks

Avik Sengupta, SaiDhiraj Amuru, Ravi Tandon, R. Michael Buehrer, and T. Charles Clancy

***Abstract*—Caching enables reduction of peak backhaul load in modern wireless systems by allowing popular files to be pre-fetched and stored locally and closer to the network edge. With the shift in paradigm from homogeneous to heterogeneous cellular networks, data offloading to small cell base stations (sBS) has garnered recent attention. Caching at the sBS can help avoid bottlenecks in the capacity limited backhaul links to the core network. In this paper, we study collaborative caching strategies in a multi-sBS heterogeneous network with unknown file popularities from a reinforcement learning perspective. We present topology-aware uncoded and coded caching strategies under a learning-aided framework where the file popularities are dynamically learned over time by observing user requests. The uncoded collaborative cache placement problem is NP-hard and we propose a novel weighted graph-coloring and local search-based approximation algorithm with an approximation ratio of $(\frac{1}{3} - \epsilon)$ for some $\epsilon > 0$. Alternately, we formulate a coded cache placement strategy which is shown to be a linear program yielding an optimal solution. Through simulations we show that the uncoded approximate caching algorithm performs close to the optimal coded scheme when integrated with the learning framework in the multi-sBS setting. We also show that for network topologies of practical interest, the collaborative caching strategies outperform local caching strategies.**

I. Introduction

The dynamics of cellular traffic over wireless networks has, in recent times, undergone a paradigm shift to become increasingly content centric with high-data-rate multimedia content distribution holding precedence. Thus it has become imperative to streamline such content distribution over networks to enable maximum utilization of scarce spectrum resources while ensuring high quality service to end-users. To this end, caching has emerged as an important technique for 5G wireless content distribution networks [1], particularly in the paradigm of delay-insensitive video distribution and streaming [2]. Caching can reduce network load by storing popular content locally at network edge caches at the last network hop. The content placement is generally performed during times of low network load so that users can be served directly from local edge caches when the traffic volume over the network is high.

With the proliferation of heterogeneous wireless architectures, caching large volumes of data at the network edge has become a feasible means for load distribution over the network. As a result, benefits of caching in modern wireless network settings has been studied extensively in recent literature [3]–[16]. Recent works like [14]–[16] study caching at mobile end-users from a novel information theoretic framework which has its roots in index coding and multicast delivery to increase efficiency of content delivery over a single server network. Caching in multi-sBS networks from an interference management perspective was studied in [17]–[19] under uniform file (defined as a block of data) popularity distribution. Information theoretic caching for non-uniform file popularity was studied in [20]–[22] However, the problem of determining the optimal content to cache in a practical network setting when file popularity is *unknown* is a difficult one. In this work, we consider this practical problem of caching popular content at the network edge augmented by a framework for learning the varying file popularity profile across users.

Heterogeneous networks, where small-cell base stations (sBSs) like femto or pico-cells are connected to a central cellular base station (BS), lend themselves to caching of data at the network edge by equipping the sBSs with cache memories for local, low-latency content dissemination. Distributed cache placement in sBSs was studied in [2], [23], with an aim to reduce the latency of file delivery to the end-users while assuming that the popularity of files was known apriori at the sBSs. However, such an assumption is unrealistic and learning based caching frameworks were presented in [5]–[9], [11]–[13] and references therein for the case of unknown file popularity profiles. The authors in [5]–[7] presented a multi-armed bandit [24], [25] based reinforcement learning framework for cache placement in a single sBS network. Transfer-learning based approaches for caching in small-cell networks were studied in [11]–[13], where minimization of backhaul load and the evaluation of time to achieve desired learning accuracy under random caching were the main areas of focus. These works, however, do not consider the problem of caching for a multi-sBSs network under practical topologies with overlapping connectivity to users under unknown file popularity distribution. In this work, we study topology-aware cache placement in a multi-sBS wireless network from a reinforcement learning perspective. We ask the following fundamental questions. *Given a multi-sBS network topology, per-sBS cache capacity and random file requests from users based on an unknown file popularity profile - (i) what is the best learning method with finite-time performance guarantees for estimating the file popularity from observations of user requests, and (ii) what is the best collaborative caching strategy, such that a maximum number of requests can be served directly from the sBS caches?*

To address these questions, we formulate the *learning-aided collaborative caching* framework for the multi-sBS environment as shown in Fig. 1. We present novel topology-aware caching strategies using a combinatorial multi-armed bandit (CMAB)-based learning framework [25]–[28]. In real networks, the file popularity profile could change over time and hence it is necessary to learn it dynamically during the

A. Sengupta, T. Charles Clancy and R. M. Buehrer are with Wireless@VT, Dept. of ECE, Virginia Tech, Blacksburg VA USA; S. Amuru is with Samsung, Bengaluru India; Ravi Tandon is with Dept. of ECE, University of Arizona, Tucson, AZ USA. Email: {aviksg,tcc,adhiraj,rbuehrer@vt.edu}, tandonr@email.arizona.edu. This work was presented in part at International Symposium on Wireless Communication Systems, August 2014 [9]. The work of R. Tandon was supported in part by the NSF Grant CCF 1559758.

TABLE I
TABLE OF NOTATIONS USED

$\mathbb{S}$	Set of N sBSs;	$\mathbb{U}$	Set of K users
$\mathbb{F}$	Set of F files;	n	sBS Index
u	User Index;	f	File Index
$\mathcal{S}_n$	$n-$th sBS;	S_f	Size of $f-$th file
$\mathcal{G}(\mathbb{S},\mathbb{U},\mathbb{E})$	Bipartite connectivity graph		
$\mathcal{N}(u)$	sBSs serving user u (neighborhood of u)		
$\mathcal{U}(\mathcal{S})$	Users connected to sBS $\mathcal{S}$		
$d^t_{f,n}$	Average demand for file f at sBS n and time t		
$\theta_{f,n}$	Popularity of file f at sBS n		
Θ_n	Popularity distribution of all files at sBS n		
γ	Skewness of ZipF popularity distribution		
$\mathcal{C}^\pi(t)$	$F \times N$ caching matrix with elements $c^\pi_{f,n} \in (0,1]$		

caching procedure. The files to be cached are modeled as the arms of a combinatorial multi-armed bandit (CMAB) problem [26] in order to learn their popularity over time. The goal of the caching strategy is to pick the best set of files at any time t based on their estimated popularity so that the requests from users can be directly served by the caches without accessing the core network. Based on this network model, the following are the main contributions of our work:

1. *Learning Framework* - We present a CMAB-based learning framework for dynamically learning the file popularity distributions in a multi-sBS network setting. We show that the bound on the sub-optimality gap of the proposed learning algorithm at each sBS at time t scales as $O(\log(t))$ i.e., the sub-optimality gap scales logarithmically with time.
2. *Collaborative Caching Framework* - We formulate a novel network topology-aware *uncoded caching* strategy where entire files are cached at the sBSs. We show that the uncoded caching problem is NP-hard and propose a novel graph-coloring based algorithm which provides a $\left(\frac{1}{3}-\epsilon\right)$-approximate cache placement solution in polynomial time for some $\epsilon > 0$. We also formulate a *coded caching* strategy which allows fractional file placement. We show that this strategy can be represented as a linear program which can be solved optimally.
3. Through simulations, we present a detailed comparison of the uncoded and coded collaborative caching schemes. We show that for both coded and uncoded schemes, collaborative caching generally outperforms naive strategies that locally optimize the cache content at each sBS without accounting for the network topology. We further show, that in spite of using a greedy approximation algorithm, the uncoded collaborative caching strategy performs similar to the provably optimal coded strategy. This is attributed to the fact that learning popularity using fractional placements is harder than learning from placement of entire files in the multi-sBS setting.

The paper is organized as follows: the network model is detailed in Section II, followed by the learning-aided collaborative caching framework in Section III. Section IV presents the CMAB-based distributed file popularity estimation and the associated regret bounds. The uncoded and coded collaborative caching strategies are designed in Section V. Numerical results are presented in Section VI and finally Section VII concludes the paper.

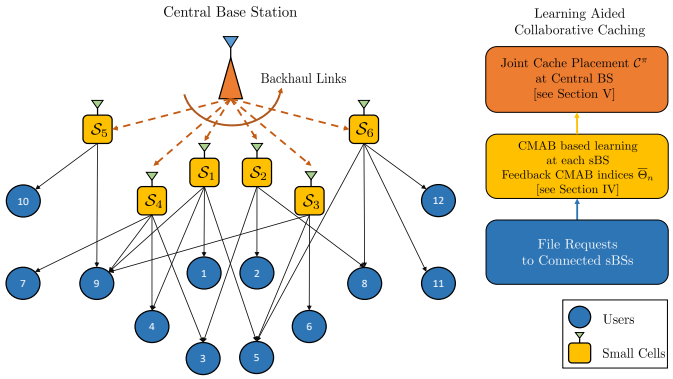


Fig. 1. Network model for learning-aided collaborative caching in a small cell network with a central BS, $N = 6$ sBSs and $K = 12$ users.

II. NETWORK MODEL

We consider a small cell network with a set of N sBSs, $\mathbb{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N\}$, connected to a central Base Station (BS). A set of K users $\mathbb{U} = \{U_1, U_2, \ldots, U_K\}$ are located in an area covered by the small cell network. Users can make random requests from a directory of F files, $f \in \mathbb{F} = \{1, 2, \ldots, F\}$, where file f has size S_f units[1]. The central BS is considered to have enough memory to store the entire file directory $\mathbb{F}$. Each sBS in the network has a cache memory that can store M units of data. We assume that $M \geq \min_{f\in\mathbb{F}} S_f$ i.e., the sBS caches are large enough to store at least one whole file. Each user can be served from the cache of one or more sBSs in the network. If a user's request cannot be serviced from the caches of the sBSs to which it is connected, then it can be downloaded directly from the central BS. This ensures that all the users' requests are serviced. However, the objective is to enable loacalized content delivery with minimal use of backhaul resources. Therefore, the optimal caching policy ensures that maximum number of user requests can be serviced locally from the sBS caches at the network edge, thereby alleviating load at the central BS. An example small cell wireless network architecture with $N = 6$ sBSs and $K = 12$ users is shown in Fig. 1. The central BS is connected to sBSs via wireless (or wired) backhaul links.

A. Network Connectivity

The connectivity of the users and sBS within the network can be modeled as a bipartite graph $\mathcal{G} = (\mathbb{S}, \mathbb{U}, \mathbb{E})$, where the edges $(\mathcal{S}_n, u) \in \mathbb{E}$ if there exists a communication link, subject to physical layer constraints (i.e., the user is not in *outage* based on path loss, shadowing etc.), between user u and the n-th sBS $\mathcal{S}_n$. $\mathcal{N}(u) \subseteq \mathbb{S}$ denotes the neighborhood of user u i.e., the sBSs that can serve the user (or conversely, the sBSs to which the user is connected). For example, from Fig. 1, $\mathcal{N}(U_1) = \{\mathcal{S}_1\}$ while $\mathcal{N}(U_3) = \{\mathcal{S}_2, \mathcal{S}_4\}$. On the other hand, the neighborhood of the nth sBS i.e., the number of users connected to sBS $\mathcal{S}_n$ is denoted by $\mathcal{U}(\mathcal{S}_n)$. For example, in Fig. 1, $\mathcal{U}(\mathcal{S}_1) = \{U_1, U_4, U_5, U_9\}$ and $\mathcal{U}(\mathcal{S}_2) = \{U_2, U_3, U_8\}$. We also assume that the central BS has complete network knowledge.

[1]The unit of file size can be Megabytes or Gigabytes depending upon the networks under consideration. We keep it generic for our exposition,

B. Model for File Popularity

The popularity of files (e.g., videos) in a multimedia content distribution network is generally modeled as a ZipF distribution [29], [30] where the mean demand (number of requests) for a file f at any sBS $\mathcal{S}_n$ can be modeled as:

$$\theta_{f,n} = \frac{f^{-\gamma}}{\sum_{k=1}^{F} k^{-\gamma}}, \tag{1}$$

where $\theta_{f,n} \in [0,1]$ and γ models the skewness of the popularity profile. For example, $\gamma = 0$ models a uniform file popularity profile and as γ increases the skewness increases. The file popularity distribution at $\mathcal{S}_n$ is denoted by $\Theta_n = \{\theta_{1,n}, \theta_{2,n}, \ldots, \theta_{F,n}\}$.

In practical settings, the distribution $\Theta_n,\ \forall n \in \{1, \ldots, N\}$ is unknown. Therefore, in this work, we empirically estimate the mean demand for each file at every sBS based on observations of user requests over a period of time. To this end, let $d_{f,n}^{t,u} \in \{0,1\}$ be an indicator variable such that $d_{f,n}^{t,u} = 1$ when a user u in the neighborhood of sBS $\mathcal{S}_n$ i.e., $u \in \mathcal{U}(\mathcal{S}_n)$ requests the file f at time t. Let

$$d_{f,n}^{t} = \frac{1}{|\mathcal{U}(\mathcal{S}_n)|} \sum_{u \in \mathcal{U}(\mathcal{S}_n)} d_{f,n}^{t,u}, \tag{2}$$

indicate the instantaneous demand for file $f \in \mathbb{F}$, averaged over all users served by sBS $\mathcal{S}_n$. $|\mathcal{U}(\mathcal{S}_n)|$ is the cardinality of the set $\mathcal{U}(\mathcal{S}_n)$. The instantaneous demand $d_{f,n}^{t}$ is an i.i.d random variable with an empirical mean

$$\widehat{\theta}_{f,n} = \mathbb{E}\left[d_{f,n}^{t}\right] \tag{3}$$

which is bounded in support $\left[0,1\right]$ and the expectation is over the random user requests at time instants $1, 2, \ldots, t$. In other words, the mean $\widehat{\theta}_{f,n}$ signifies the empirical popularity of the f-th file at the n-th sBS $\mathcal{S}_n$ i.e., the average number of per-user requests for the file f at $\mathcal{S}_n$ (from the users in $\mathcal{U}(\mathcal{S}_n)$) till time t. Thus, although the true popularity distribution Θ_n is unknown, we estimate the file popularity distribution $\widehat{\Theta}_n = \{\widehat{\theta}_{1,n}, \widehat{\theta}_{2,n}, \ldots, \widehat{\theta}_{F,n}\}$ at $\mathcal{S}_n$ based on the user requests up to time t. In the next section we introduce the learning framework for collaborative caching in a multi-sBS environment.

III. Learning-Aided Collaborative Caching

In multimedia content distribution networks, files generally have varying popularities i.e., a few files in the system are highly popular (e.g., viral YouTube videos) while others are requested less frequently[20], [31]. The resulting popularity profile is accurately modeled by the heavy-tailed ZipF distribution [20], [29], [31]. In order to design efficient caching and delivery policies such that majority of the requested content can be delivered locally, reliable estimation of the file popularity over time is required at each sBS. To address this, we introduce a topology-aware learning-aided collaborative caching framework. Over learning iterations (time), the central BS receives feedback from the sBSs about the *performance* of the caching policy. Performance is measured in terms of a *reward* i.e., the amount of data which is downloaded from an sBS in servicing the requests of connected users. Similarly, the sub-optimality gap of the learning process is defined as *regret* [25]. Traditional learning techniques such as Q-Learning, ϵ-greedy learning etc. provide only asymptotic convergence guarantees [32]. However, in a caching framework, such asymptotic guarantees (when the number of observed user demands goes to infinity) are not sufficient.

To this end, we model the popularity estimation and cache placement as a Multi-Armed-Bandit (MAB) problem [25], allowing us to present finite time guarantees on the learning performance. In this setting, the files $f \in \mathbb{F}$ are treated as the arms and caching a file at a sBS is equivalent to pulling an arm in a MAB framework. The caching of files at each sBS enables estimation of file popularity by observing the rewards obtained in return i.e., by the amount of data which can be used from the local caches to serve user requests. Since each sBS cache can typically store multiple files, it is appropriate to model the problem as a combinatorial MAB (CMAB) problem [5]–[9], [26]–[28]. Under a combinatorial setting, a set of arms, called a *super-arm*, can be pulled at a given time instant. Thus for a cache placement policy which places multiple files in the cache of $\mathcal{S}_n$, the strategy at any time t is equivalent to pulling a CMAB super-arm. A single-sBS CMAB based caching framework was studied in [5]. Our work extends it to a topology-aware multi-sBS network.

For CMAB-based learning, there is a tradeoff between the exploration of new arms (i.e., caching new files to estimate their popularity) and the exploitation of the known arms (caching files that are known to have high popularity and hence give higher rewards). In this work, we utilize the exploration-exploitation tradeoff by use of the Combinatorial Upper Confidence Bound (CUCB) algorithm [26], [27], which provides finite time convergence guarantees. We propose a distributed learning framework detailed in Algorithm 1, which has two main parts:

1. *Learning the File Popularity Distribution:* A CMAB based distributed popularity estimation runs at each sBS $\mathcal{S}_n$ to estimate the empirical file popularity distribution $\widehat{\Theta}_n$, associated with its users $\mathcal{U}(\mathcal{S}_n)$. At time t, based on user demands up to $t-1$, it updates the popularity of every file that is cached in the sBS. The updated popularity profile is used in the subsequent cache placement by the central BS. The learning framework is discussed in Section IV.
2. *Collaborative Cache Placement (CCP):* The central BS takes the CMAB based popularity profile from the sBSs as input and designs cache placement strategies by jointly maximizing the sum reward of the sBSs. The cache placement can be *uncoded* where entire files are cached at the sBSs or *coded* where files are sub-packetized and fractions of files are allowed to be cached. Details on the design of both uncoded and coded collaborative cache placement are discussed in Section V.

We next present a detailed analysis of the learning-aided collaborative caching algorithm.

IV. Learning the File Popularity Distribution

In this section, we present the CMAB based framework for learning the file popularity distribution. First, we define the caching model, present the CUCB based learning algorithm and finally, derive finite time performance guarantees for the learning step in Algorithm 1.

Algorithm 1 LEARNING-AIDED COLLABORATIVE CACHING

DISTRIBUTED POPULARITY ESTIMATION AT SBS:

for Each sBS $\mathcal{S}_n \in \mathbb{S}$ at time t **do**
  for Each file $f \in \mathbb{F}$ **do**
    if File f is cached i.e., $c^{\pi}_{f,n}(t) \geq 0$ **then**
      $$T_{f,n} = T_{f,n} + 1$$
      where, $T_{f,n}$ is the number of times file f (or a fraction of file f) is cached in $\mathcal{S}_n$ upto t.
    else if for any $u \in \mathcal{U}(\mathcal{S}_n)$, $\exists\ \mathcal{S}_{n'} \in \mathcal{N}(u) \setminus \{n\}$, $c^{\pi}_{f,n'}(t) \geq 0$, update $T_{f,n} = T_{f,n} + 1$.
    end if
    Update the mean demand:
      $$\widehat{\theta}_{f,n} = \frac{\sum_{i=1}^{t} d^{i}_{f,n}}{t}$$
      where, $\sum_{i=1}^{t} d^{i}_{f,n}$ is the cumulative sum of requests for file f from users $u \in \mathcal{U}(\mathcal{S}_n)$ until time instant t.
    Calculate CMAB Index:
      $$\overline{\theta}_{f,n} = \widehat{\theta}_{f,n} + \sqrt{\frac{\Psi_{f,n} \log\left(|\mathcal{U}(\mathcal{S}_n)|t\right)}{2T_{f,n}}}$$
      $|\mathcal{U}(\mathcal{S}_n)|$ indicates cardinality of $\mathcal{U}(\mathcal{S}_n)$
  end for
  Feedback $\overline{\Theta}_n = [\overline{\theta}_{1,n}, \overline{\theta}_{2,n}, \ldots, \overline{\theta}_{F,n}]$ to central BS.
end for

TOPOLOGY BASED COLLABORATIVE CACHING:

Caching Strategy at time $t+1$:
  $$[\mathcal{C}^{\pi}(t+1)]_{F\times N} = \text{CCP}\left(\overline{\Theta}_1, \overline{\Theta}_2, \ldots, \overline{\Theta}_N\right)$$
Observe actual user requests $d^{t+1}_{f,n}$ at time $(t+1)$
Return to Step 1 and update $T_{f,n}$'s and $\overline{\Theta}_n$'s for the next time step $(t+1)$

A. The Cache Placement Model

In this work, we consider a Collaborative Cache Placement (CCP) policy π which determines the optimal content to be cached at each of the N sBSs in the network. The objective of the policy π is to place files in the caches of the sBSs based on the network topology and empirical history of user demands (file popularity) such that maximum number of file requests can be downloaded directly from the sBS caches. Let the $F \times N$ binary matrix $\mathcal{C}^{\pi}(t)$, denote the collaborative cache placement by the policy π at time step t. Let

$$[\mathcal{C}^{\pi}(t)]_{F\times N} = [\mathbf{c}^{\pi}_1(t), \mathbf{c}^{\pi}_2(t), \ldots, \mathbf{c}^{\pi}_N(t)], \tag{4}$$

where $\mathbf{c}^{\pi}_n(t)$ is the $F \times 1$ cache placement vector for the sBS $\mathcal{S}_n$ and $c^{\pi}_{f,n} \in [0,1]$ are the elements of $\mathbf{c}^{\pi}_n(t)$ such that, at a time step t,

$$c^{\pi}_{f,n}(t) \begin{cases} \geq 0 & \text{if file } f \text{ is cached at } \mathcal{S}_n \\ = 0 & \text{otherwise} \end{cases} \tag{5}$$

In our problem formulation, we define *caching* as placement of a *whole file* or any *fraction of a file* in the cache of the sBS i.e., $c^{\pi}_{f,n}$ denotes the fraction of file f cached at $\mathcal{S}_n$. At $t = 0$, we consider that the caches of all sBSs are empty i.e., $\mathcal{C}^{\pi}(0) = [\mathbf{0}]_{F\times N}$.

B. CMAB aided File Popularity Estimation

The CMAB based file popularity estimation at each sBS gets as reward, the amount of data downloaded from the sBS cache to serve the requests of its users. By tracking the reward for cache placement over time, the sBS aims to learn the optimal cache placement policy π. The algorithm is initialized by sequentially placing each file once in the cache of each sBS. At time t, sBS $\mathcal{S}_n$ learns the file popularity distribution i.e., the empirical mean of the instantaneous demands $\widehat{\Theta}_n = \{\widehat{\theta}_{1,n}, \widehat{\theta}_{2,n}, \ldots, \widehat{\theta}_{F,n}\}$ based on the history of instantaneous demands, $d^{1}_{f,n}, d^{2}_{f,n}, \ldots, d^{t}_{f,n}$, upto time t and the cache placement by the CCP policy π at time t i.e., $c^{\pi}_{f,n}(t) \in \mathcal{C}^{\pi}(t), f \in \mathbb{F}$. To this end, a *CMAB index* $\overline{\theta}_{f,n}$ is calculated for each file $f \in \mathbb{F}$. The CMAB index has two components:

1. The *empirical mean* of the instantaneous demand $\widehat{\theta}_{f,n}$ derived from the observation of user requests over time.
2. An additive *perturbation factor*
$$\sqrt{\frac{\Psi_{f,n} \log\left(|\mathcal{U}(\mathcal{S}_n)|t\right)}{2T_{f,n}}}, \tag{6}$$
where $|\mathcal{U}(\mathcal{S}_n)|$ is the number of users connected to $\mathcal{S}_n$ and $T_{f,n}$ denotes the number of times a file $f \in \mathbb{F}$ has been cached in $\mathcal{S}_n$ until time t. The factor
$$\Psi_{f,n} = \frac{\delta}{|\mathcal{U}(\mathcal{S}_n)|}\left(\frac{|\mathcal{U}(\mathcal{S}_n)|S_f}{F^{\gamma}}\right)^2 \tag{7}$$
is directly proportional to $|\mathcal{U}(\mathcal{S}_n)|$ and a roll-off factor δ (which is usually set to 3 [5], [25]), and inversely proportional to the skewness factor γ of the ZipF based model of the file popularity profile. The factor γ can be empirically estimated as in [29] from the observations of the instantaneous file demands $d^{t}_{f,n}$ over time. We show in Section IV-C that the sub-optimality of the learning process scales linearly with $\Psi_{f,n}$ for the proposed collaborative caching algorithm.

The perturbation factor in the CMAB index in (6) promotes exploration and exploitation based on the number of connected users, the size of each file and the Zipf parameter γ. It promotes exploration by forcing the CCP to place less-often-cached sets of files (for which $T_{f,n}$ is low) in the caches by increasing their index value. This promotes sufficient sampling of lesser requested files in order to accurately evaluate their popularity. It also promotes exploitation when $|\mathcal{U}(\mathcal{S}_n)|$ is large i.e., $\mathcal{S}_n$ has a large user-set and also when the popularity profile is skewed i.e., when γ is large and there are few popular files in the system. Once, the sBSs calculate the local index values $\overline{\Theta}_n$, this information is fed back to the central BS where the CCP policy π determines the caching strategy $\mathcal{C}^{\pi}(t)$.

Remark 1 (*Topology-Aware $T_{f,n}$ Update*)**.** In this work, we use a unique topology-aware $T_{f,n}$ update procedure which helps in capturing the interaction of users who are connected to multiple sBSs. At $\mathcal{S}_n$, let a user u request a file $f \in \mathbb{F}$. If the file is not cached at $\mathcal{S}_n$ i.e., $c^{\pi}_{f,n}(t) = 0$, then we consider the caches of all *other* sBSs in the neighborhood of u: $\mathcal{S}_{n'} \in \mathcal{N}(u) \setminus \{\mathcal{S}_n\}$. If f is cached at any of these sBSs i.e., if the user's request is satisfied by any other sBS, then we update the $T_{f,n}$ for the sBS $\mathcal{S}_n$ as well. As opposed to the single sBS learning in [5], this leads to a topology-aware learning framework which can account for the network connectivity. Note that at any time-step t, $T_{f,n} \leq t$ still holds. ■

C. Upper Bounds on the Regret for Algorithm 1

In order to derive finite time performance guarantees on the learning step of Algorithm 1, we first formally define the concept of reward and regret (sub-optimality gap) for the caching framework under consideration.

1) Reward: For the policy π, with a caching strategy $\mathbf{c}_n^{\pi}(t)$ at sBS $\mathcal{S}_n$ at time step t, we define the *instantaneous reward* for caching a file f in $\mathcal{S}_n$ as:

$$r_{f,n}^{t} = \sum_{u \in \mathcal{U}(\mathcal{S}_n)} d_{f,n}^{t,u} c_{f,n}^{\pi}(t) S_f = |\mathcal{U}(\mathcal{S}_n)| d_{f,n}^{t} c_{f,n}^{\pi}(t) S_f = |\mathcal{U}(\mathcal{S}_n)| d_{f,n}^{t} \tilde{S}_{f,n}, \quad (8)$$

which indicates that a reward of $\tilde{S}_{f,n} = c_{f,n}^{\pi}(t) S_f$ is obtained when $c_{f,n}^{\pi}(t)$ fraction of a file f requested by a user u is available for download from a local cache. The reward is proportional to the amount of data downloaded from the cache. At time t, the *expected accumulated reward*, $R_{\widehat{\Theta}_n}(\mathbf{c}_n^{\pi}(t))$, is obtained by taking an expectation over the instantaneous rewards up to time t:

$$R_{\widehat{\Theta}_n}(\mathbf{c}_n^{\pi}(t)) \overset{\text{(a)}}{=} \mathbb{E}\left[\sum_{f: c_{f,n}^{\pi}(t)>0} r_{f,n}^{t}\right] = \mathbb{E}\left[\sum_{f: c_{f,n}^{\pi}(t)>0} |\mathcal{U}(\mathcal{S}_n)| d_{f,n}^{t} \tilde{S}_{f,n}\right] \overset{\text{(b)}}{=} |\mathcal{U}(\mathcal{S}_n)| \sum_{f: c_{f,n}^{\pi}(t)>0} \widehat{\theta}_{f,n} \tilde{S}_{f,n}, \quad (9)$$

where, in (a), the expectation is taken over the i.i.d instantaneous file demands $d_{f,n}^{t}$ and (b) follows from the definition of empirical estimate of mean demand in (3). The objective of Algorithm 1 is to maximize the expected reward for all time instants $1, 2, \ldots, t$. We define the *optimal* reward, $R_{\Theta_n}^{\text{opt}}$, as the expected reward obtained when the CCP policy π caches the *optimal* set of files when the *true* popularity profile $\Theta_n = \{\theta_{1,n}, \theta_{2,n}, \ldots, \theta_{F,n}\}$ is perfectly known at the BS.

2) Regret: The regret of the CCP policy π at a time instant t is defined as the difference between the expected accumulated reward obtained by the caching policy and the optimal expected reward $R_{\Theta_n}^{\text{opt}}$. Assuming that policy π uses an $(\alpha, \beta)-$approximation cache placement algorithm i.e., $\Pr\left[R_{\widehat{\Theta}_n}(\mathbf{c}_n^{\pi}(t)) > \alpha \cdot R_{\Theta_n}^{\text{opt}}\right] \geq \beta$, the regret at a finite time horizon $t = T$ is given by:

$$\text{Reg}_{\Theta_n,\alpha,\beta}^{\pi}(T) = T\alpha\beta R_{\Theta_n}^{\text{opt}} - \mathbb{E}\left[\sum_{t=1}^{T} R_{\widehat{\Theta}_n}(\mathbf{c}_n^{\pi}(t))\right], \quad (10)$$

where the expectation is over policy π and all the rewards generated by files cached by $\mathbf{c}_n^{\pi}(0), \mathbf{c}_n^{\pi}(1), \ldots, \mathbf{c}_n^{\pi}(T)$.

A caching strategy $\mathbf{c}_n^{\pi}(t)$ is defined to be α-*sub-optimal* if the reward obtained by the strategy is less than α fraction of the optimal reward:

$$R(\mathbf{c}_n^{\pi}(t)) < \alpha \cdot R_{\Theta_n}^{\text{opt}}.$$

Let $\Delta_{n,\max}^{f}$ be the difference in expected reward between the optimal caching strategy and the *worst* α-sub-optimal caching strategy in which file f is cached at $\mathcal{S}_n$ i.e., it is the sub-optimality gap of the strategy which yields the lowest reward when file f is cached at $\mathcal{S}_n$. Similarly, let $\Delta_{n,\min}^{f}$ be the sub-optimality gap of the *best* α-sub-optimal caching strategy i.e., the best strategy with reward less than $\alpha \cdot R_{\Theta_n}^{\text{opt}}$ in which file f is cached at $\mathcal{S}_n$. Also, let

$$\Delta_{n,\max} = \max_{f \in \mathbb{F}} \Delta_{n,\max}^{f}.$$

be the *worst* sub-optimality gap across all files $f \in \mathbb{F}$. Now, define a counter $N_b(t)$ which counts the number of times a α-sub-optimal caching strategy is employed by the CMAB algorithm upto a time instant t. Using techniques similar to [26], we can derive an upper bound on the expected number of α-sub-optimal periods at a finite time horizon $t = T$:

$$N_b(T) \leq (1-\beta)(T-F) + 2F\left[\frac{\zeta(\Psi_{f,n}-1)}{|\mathcal{U}(\mathcal{S}_n)|^{\Psi_{f,n}}} + \Delta_{n,\max} \sum_{f \in \mathbb{F}} \frac{\Psi_{f,n} \log(|\mathcal{U}(\mathcal{S}_n)|T)}{(g^{-1}(\Delta_{n,\min}^{f}))^2}\right], \quad (11)$$

where $g(\cdot)$ is a strictly increasing, invertible function[2] such that for any two popularity distributions Θ and $\Theta^{'}$, we have $|R_{\Theta}(\mathbf{c}^{\pi}(t)) - R_{\Theta^{'}}(\mathbf{c}^{\pi}(t))| \leq g(\Lambda)$ if $\max_{f \in \mathbf{c}^{\pi}(t)} |\theta_f - \theta_f^{'}| \leq \Lambda$. From (11), we can see that the number of α-sub-optimal periods scales logarithmically with time T. The following Lemma gives an upper bound on the regret for the n-th sBS for the Collaborative Caching Algorithm (Algorithm 1).

Lemma 1. *At each sBS, the regret at time $t = T$, when using an $(\alpha, \beta)-$approximation CCP algorithm, is upper bounded by:*

$$\text{Reg}_{\Theta_n,\alpha,\beta}^{\pi}(T) \leq \sum_{f \in \mathbb{F}, \Delta_{n,\min}^{f}>0} \frac{2\Psi_{f,n} \log\left(|\mathcal{U}(\mathcal{S}_n)|T\right)}{(g^{-1}(\Delta_{n,\min}^{f}))^2} \cdot \Delta_{n,\max}^{f} + \left(\frac{2\zeta(\Psi_{f,n}-1)}{|\mathcal{U}(\mathcal{S}_n)|^{\Psi_{f,n}}} + 1\right) \cdot F \cdot \Delta_{n,\max}, \quad (12)$$

where $\zeta(x)$ is the Riemann Zeta function.

The proof is given in [33, Appendix A]. It can be seen from (12), that the regret at each sBS scales as $O\left(\log\left[|\mathcal{U}(\mathcal{S}_n)|T\right]\right)$. Thus, when $T >> |\mathcal{U}(\mathcal{S}_n)|$, the loss due to lack of knowledge of the true popularity distribution i.e., the regret, grows slowly with T. We next present an example to illustrate the regret bound, noting that the result holds for any arbitrary network.

Example 1 (*Regret Bounds for 2 sBS, 3 User System*)**.** We consider the network in Fig. 2(a). Let there be a library of 3 files $\mathbb{F} = \{F_1, F_2, F_3\}$ with $S_f = 1, \forall f$ and let each sBS have a cache size of $M = 1$. We also define a file popularity profile $\Theta_n = \{\theta_{n,1}, \theta_{n,2}, \theta_{n,3}\}$ such that $\gamma = 2$ and

$$\theta_{n,1} = \theta_{n,2} = \frac{2}{5} \quad \theta_{n,3} = \frac{1}{5}, \; n = 1, 2.$$

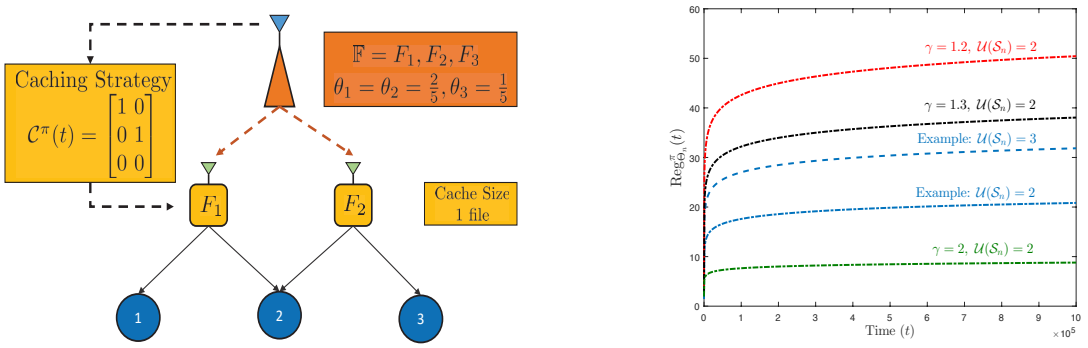
For the system under consideration, the scaling factor $\Psi_{f,n}$ at each sBS is given by

$$\Psi_{f,n} = \Psi_n = \frac{3}{2} \times \left(\frac{2 \times 1}{3^2}\right)^2 = 0.0741; \quad f = 1, 2, 3. \quad (13)$$

We assume a caching strategy where only whole files can be cached i.e., $c_{f,n}^{\pi} \in \{0, 1\}$. Since the file F_3 is the least popular in both sBSs, lets assume that an optimal caching policy is as shown in Fig. 2(a) i.e., $\mathcal{S}_1$ stores F_1 and $\mathcal{S}_2$ stores F_2. The reward in this case is equal to $\theta_{f,n}$ since files have unit size. In this case, by definition, the only non-zero $\Delta_{m,\min}^{f}, \Delta_{m,\max}^{f}$ are those corresponding to file F_3 and given by

$$\Delta_{n,\min}^{3} = \Delta_{n,\max}^{3} = \frac{2}{5} - \frac{1}{5} = \frac{1}{5}. \quad (14)$$

[2]The function $g(\cdot)$ is a *bounded smoothness function*[26] which is an artifact of the proof of the upper bound on the regret bound in Lemma 1.



(a) (b)

Fig. 2. (a) A collaborative caching set-up with $N = 2$ sBSs, $K = 3$ users, (b) Upper bound on regret vs. time.

Again for this example, we use $g(x) = x$ or conversely $g^{-1}(x) = x$. Then at time $t = T$, we have

$$\mathrm{Reg}^\pi_{\Theta_n}(T) \leq \frac{2\Psi_n \log(2T)}{0.2} + \frac{3}{5}\left(\frac{2\zeta(\Psi_n - 1)}{2^{\Psi_n}} + 1\right). \quad (15)$$

The upper bound on the regret is plotted as a function of time t in Fig. 2(b). We also plot a bound for a case where the number of users per sBS increases to 3. These are shown by the blue curves. It is seen that the value of regret increases with increasing users which is expected. But for both cases, the regret grows slowly with large values of t. Thus in both cases, the bound shows that with increasing t, the learning converges and the regret grows only slowly with time. Further we also generate some popularities using the Zipf distribution [29] with varying skewness γ. The optimal placement in this case is to place the most popular file in the cache. It can be seen that as the skewness increases, the regret decreases i.e., as fewer files become highly popular, it gets easier to learn the optimal cache placement. Also, when $\gamma = 0$ i.e., every file is equally popular, the regret grows arbitrarily large and optimal cache placement is difficult to learn. This will be further illustrated in Section VI where we present simulation results. □

Corollary 1 (Overall Regret)**.** *The system regret over all the N sBSs is upper bounded by:*

$$\sum_{n=1}^{N}\left[\sum_{f\in\mathbb{F},\Delta^f_{n,\min}>0}\frac{2\Psi_{f,n}\log\left(|\mathcal{U}(\mathcal{S}_n)|t\right)}{(g^{-1}(\Delta^f_{n,\min}))^2}\cdot\Delta^f_{n,\max} + \left(\frac{2\zeta(\Psi_{f,n}-1)}{|\mathcal{U}(\mathcal{S}_n)|^{\Psi_{f,n}}}+1\right)\cdot F\cdot\Delta_{n,\max}\right] \quad (16)$$

From Corollary 1, we can see that the overall regret for the multi-sBS system with N sBSs is the sum of the regrets of the individual sBSs. Thus, if the CCP policy π intelligently places content such that a user u with multiple connections can download its requested content from *any one* of its connections, then the regret for all the sBS in $\mathcal{N}(u)$ also decreases. This is one of main design goals for the CCP solvers as discussed in the sequel. In the next section we detail the design of the CCP policy π which chooses the joint caching strategy at the sBSs accounting for the topology of the network.

V. Collaborative Cache Placement

In this section, we detail the design of the collaborative cache placement (CCP) policy π which enables the central BS to determine joint caching strategies at the sBSs. The CCP in Algorithm 1 assumes full topological knowledge and takes as input, the CMAB indices, $\overline{\Theta}_1, \overline{\Theta}_2, \ldots, \overline{\Theta}_N$ at time t. The caching strategy $\mathcal{C}^\pi(t)$ determines which files are cached in the sBSs for servicing the user requests at time $t+1$. The CCP in Algorithm 1 produces an $(\alpha, \beta)-$approximate solution i.e., an $\alpha-$approximate solution for at least β fraction of time. Thus, for an *optimal caching scheme*, we have $\alpha = \beta = 1$. However, as discussed in the following subsection, not all caching policies lend themselves to optimal solutions in polynomial time and hence the use of an $(\alpha, \beta)-$approximation algorithm is assumed for generality. It is to be noted that for the regret bound in Lemma 1, the terms $\Delta^f_{n,\max}, \Delta^f_{n,\min}$ are dependent on α while the expected number of α-sub-optimal periods in (11) scales with β. Therefore, we should aim to use cache placement algorithms with $\beta = 1$ and α close to 1. We next highlight two different caching strategies considered in this work and comment on the α, β values for both:

1. The first is an *uncoded caching strategy* where whole files are stored in the sBS caches. We show that this is an NP-hard problem and propose a novel graph-coloring based approximate solution.
2. The second is a *coded caching strategy* where files are encoded using rateless-codes [2], [9], [34] thereby enabling fractions of files to be stored in caches. We show that, in contrast to the uncoded caching problem, this strategy leads to a concave reward function and the cache placement for reward maximization yields an optimal linear programming solution.

In the following discussion the indexing on t is dropped for simplicity. We assume that user requests and the data downloaded from the caches to satisfy these requests, at time t, are observed. The demands are then updated for the cache placement phase at the next time instant $t+1$.

A. Uncoded Collaborative Caching Strategy

We first formulate the optimal uncoded caching strategy where entire files are cached at the sBSs. In order to facilitate learning, the CMAB indices $\overline{\Theta}_n$ are used as a proxy for the

true popularity of all files $f \in \mathbb{F}$ at the sBSs $\mathcal{S}_n$. Based on this acquired knowledge, the average reward for each user u in the network (upto the current time step) can be defined as:

$$\overline{R}_u^f = \max_{n \in \mathcal{N}(u)} \overline{\theta}_{f,n} \cdot S_f \cdot c_{f,n}^{\pi}, \tag{17}$$

where $c_{f,n}^{\pi}$ are constrained to be binary variables i.e., $c_{f,n}^{\pi} \in \{0,1\}$. They are the elements of the $F \times N$ cache assignment matrix $\mathcal{C}^{\pi}$ and $c_{f,n}^{\pi} = 1$ denotes that the file f is cached at $\mathcal{S}_n$. The reward formulation is such that for a user connected to multiple sBSs, the obtained reward is maximized when the requested file is placed in the sBS where it is *most popular* i.e., the sBS with the highest $\overline{\theta}_{f,n}$. Additionally, in case the file is placed in multiple locations, the $\max(\cdot)$ ensures that only the reward based on the *most popular* location is counted. As a result, the proposed reward function implicitly encourages the placement of a file f requested by user u at the sBS in $\mathcal{N}(u)$ where the popularity of the file is the highest.

The optimal uncoded cache placement maximizes the sum reward of the users in the network. The cache placement is formulated as the following binary integer program subject to cache memory constraints at each sBS:

$$\textbf{UNC-1:} \quad \max_{\mathcal{C}^{\pi}} \sum_{u \in \mathbb{U}} \sum_{f=1}^{F} \overline{R}_u^f \quad \text{s.t.} \quad \sum_{f=1}^{F} c_{f,n}^{\pi} \cdot S_f \leq M, \quad \forall n.$$

The optimal cache placement $\mathcal{C}^{\pi}$ in **UNC-1** that maximizes the sum reward of the multi-sBS network ensures that there is minimal repetitive placement of a file f in the system by virtue of the $\max(\cdot)$ function in the reward. The caveat remains that repetition might be necessary to account for similar requests from users not connected to the same sBSs. The formulation is flexible in this regard by allowing repetition since the overall reward is the sum of rewards for each user in the network. Thus our formulation utilizes the network topology to optimize the cache placement at the sBSs.

Lemma 2. *The binary integer program **UNC-1** is NP-hard.*

Proof. The lemma can be proved by showing that specific instances of the problem is NP-hard. Consider the two network examples presented in Fig. 3. Network Example 1 represents a

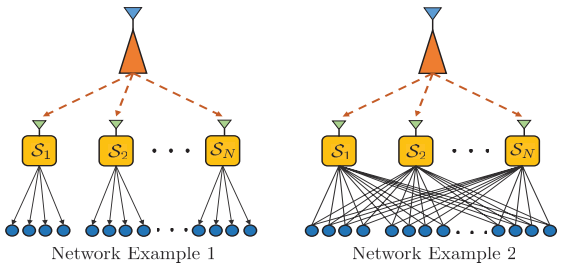


Fig. 3. Network Topology Examples

network topology where every sBS has its own set of users and each user is connected to only one sBS. Under this topology, the multi-sBS cache placement problem reduces to solving N single sBS problems of the following form subject to the cache storage constraint of **UNC-1**:

$$\max_{\mathcal{C}^{\pi}} \sum_{n=1}^{N} \sum_{u \in \mathcal{U}(\mathcal{S}_n)} \sum_{f=1}^{F} \overline{\theta}_{f,n} \cdot S_f \cdot c_{f,n}^{\pi}. \tag{18}$$

Each of the single-sBS placement problems are NP-hard Knapsack Problems [35] with values $\overline{\theta}_{f,n} S_f$ and weights S_f. Again, Network Example 2 shows the case of a fully connected network, where all users are connected to all sBSs. In this case, solving **UNC-1** becomes analogous to solving a single sBS problem but with a cache size of NM units. This is also a Knapsack problem and is NP-hard. Since these two special cases of **UNC-1** are NP-hard, **UNC-1** itself is NP-hard. Specifically, **UNC-1** is an example of a the Generalized Assignment Problem (GAP) [36]. This concludes the proof of the lemma. ■

Since **UNC-1** is NP-hard, we next formulate an approximation algorithm which runs in polynomial time. We propose a novel graph coloring based approach for reducing the given cache placement problems to multiple sub-problems where each sub-problem can be cast as a Separable Assignment Problem (SAP) [37] which yield $\left(1 - \frac{1}{e}\right)$ −approximations but in exponential time. For this work, we use a local-search based ϵ−greedy polynomial time approximation algorithm for each SAP for some $\epsilon > 0$.

B. An Approximation Algorithm for Uncoded Caching

In this section we present a novel approximation algorithm for **UNC-1**. The algorithm stems from two key ideas namely:

1. The N sBSs are divided into groups such that sBSs within each group have a high number of common users. This reduction helps in grouping sBSs such that each group of sBSs can be treated as a single contiguous cache.
2. Within each group, a variant of the **UNC-1** problem is solved with an added constraint that files cannot be replicated across the sBSs. This converts **UNC-1** to an SAP and was originally studied in [37].

Next, we discuss the two steps in detail. To this end, consider a reduction of the original NP-hard GAP problem **UNC-1** to a related problem whose solution upper bounds the solution of **UNC-1** subject to the same constraints:

$$\begin{aligned}
\max_{\mathcal{C}^{\pi}} \sum_{u \in \mathbb{U}} \sum_{f=1}^{F} \max_{n \in \mathcal{N}(u)} \overline{\theta}_{f,n} \cdot S_f \cdot c_{f,n}^{\pi} \\
&\overset{\text{(a)}}{\leq} \max_{\mathcal{C}^{\pi}} \sum_{u \in \mathbb{U}} \sum_{n=1}^{\mathcal{N}(u)} \sum_{f=1}^{F} \overline{\theta}_{f,n} \cdot S_f \cdot c_{f,n}^{\pi} \\
&\overset{\text{(b)}}{\leq} \max_{\mathcal{C}^{\pi}} \sum_{n=1}^{N} \sum_{u \in \mathcal{U}(\mathcal{S}_n)} \sum_{f=1}^{F} \overline{\theta}_{f,n} \cdot S_f \cdot c_{f,n}^{\pi} \\
&\overset{\text{(c)}}{=} \max_{\mathcal{C}^{\pi}} \sum_{n=1}^{N} |\mathcal{U}(\mathcal{S}_n)| \sum_{f=1}^{F} \overline{\theta}_{f,n} \cdot S_f \cdot c_{f,n}^{\pi},
\end{aligned} \tag{19}$$

where, (a) follows from replacing the $\max(\cdot)$ function with the sum over all the sBS $\mathcal{S}_n \in \mathcal{N}(u)$, and (b)-(c) follows from reversing the order of summation in (a). Note that the term within the first summation in (19) is the expected accumulated reward from (9). The problem in (19) is still a GAP and hence NP-hard.

In order to simplify the problem in (19), it is divided into sub-problems such that each sub-problem deals with a group of sBSs. We aim to identify sBSs with common users which

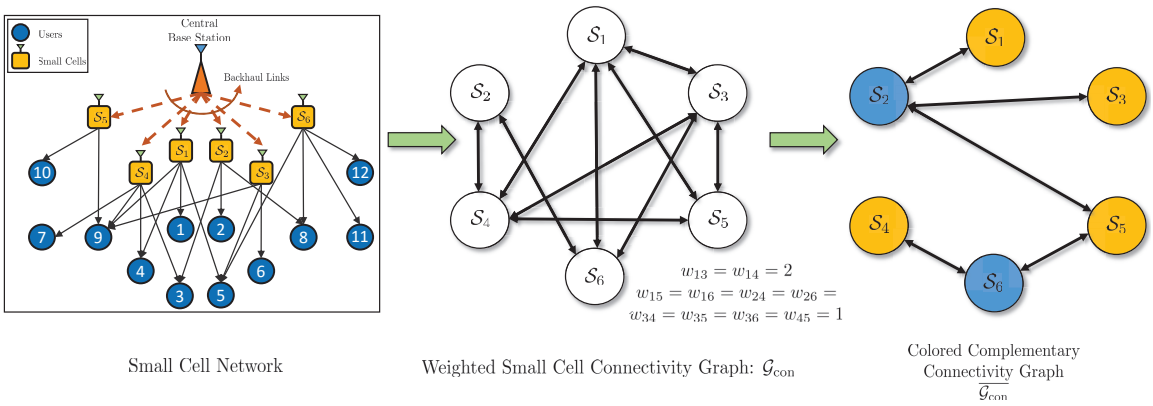


Fig. 4. Illustration of the proposed W-DSATUR algorithm.

can be grouped together to act as a single cache. Thus, the summation over n in (19) is divided into multiple sums:

$$\max_{\mathcal{C}^{\pi}} \sum_{g=1}^{G} \sum_{n \in g} |\mathcal{U}(\mathcal{S}_n)| \sum_{f=1}^{F} \overline{\theta}_{f,n} \cdot S_f \cdot c^{\pi}_{f,n}, , \tag{20}$$

where, G is the total number of groups of sBSs. Sub-problems for each group g can then be solved individually. Now considering such a sub-problem, for sBSs which can be grouped together into a single cache, each file f requested by a user in the group will be placed in *only one* of the caches. Thus assuming a group g of $\tilde{n}$ ($\tilde{n} \leq N$) sBSs, the following problem needs to be solved:

$$\textbf{UNC-2:} \quad \max_{\mathcal{C}^{\pi}} \sum_{n=1}^{\tilde{n}} |\mathcal{U}(\mathcal{S}_n)| \sum_{f=1}^{F} \overline{\theta}_{f,n} S_f c^{\pi}_{f,n}$$

$$\text{s.t.} \quad \sum_{f=1}^{F} c^{\pi}_{f,n} \cdot S_f \leq M, \ \forall n \in g; \ \sum_{n=1}^{\tilde{n}} c^{\pi}_{f,n} = 1, \quad \forall f. \tag{21}$$

The additional constraint $\sum_{n=1}^{\tilde{n}} c^{\pi}_{f,n} = 1, \quad \forall f$, ensures the placement of each file in only one sBS within the group. The optimization in (21) is a special case of the SAP presented in [37]. We next present the graph coloring based sBS grouping algorithm and then provide an greedy local search based algorithm for caching within each group.

1) Graph Coloring based sBS Grouping: In order to facilitate the grouping of sBSs, first, a *weighted connectivity graph* $\mathcal{G}_{\text{con}} = (\text{sBS}, E)$ is constructed, where edges between any two sBSs, $(\mathcal{S}_i, \mathcal{S}_j)$ exist if they have common users i.e., $\{\mathcal{U}(\mathcal{S}_i) \bigcap \mathcal{U}(\mathcal{S}_j) \neq \phi\}$. The weight of each edge w_{ij} is given by the number of common users between $\mathcal{S}_i$ and $\mathcal{S}_j$. Since $\mathcal{G}_{\text{con}}$ is an undirected graph, $w_{ij} = w_{ji}$. Next we construct the *complementary connectivity graph* $\overline{\mathcal{G}_{\text{con}}} = (\text{sBS}, E^{'})$ where edge $(\mathcal{S}_i, \mathcal{S}_j) \in E^{'}$ iff the edge between $(\mathcal{S}_i, \mathcal{S}_j) \notin E$. Note that for any node $\mathcal{S}_i$ in $\overline{\mathcal{G}_{\text{con}}}$, the set of non-neighbors forms a set of weighted neighbors in $\mathcal{G}_{\text{con}}$. Fig. 4 shows the construction of the weighted connectivity and complementary connectivity graph for the example network shown in Fig. 1.

Next, a $k-$coloring for the graph $\overline{\mathcal{G}_{\text{con}}}$ is obtained [38]. A $k-$coloring is the assignment of k colors to the vertices of a graph such that no two adjacent vertices have the same color. Note that $k = N$ signifies that $\overline{\mathcal{G}_{\text{con}}}$ is fully connected i.e., none of the sBSs have common users as in Network Example 1 in Fig. 3. Again for Network Example 2 in Fig. 3, $\overline{\mathcal{G}_{\text{con}}}$ will have no edges and $k = 1$. Since these two are the extreme examples

Algorithm 2 W-DSATUR Coloring Algorithm

1: Arrange nodes in $\overline{\mathcal{G}_{\text{con}}}$ by decreasing order of degrees.
2: Color a node of maximal degree with color 1.
3: Choose a node $\mathcal{S}_i$ with a maximal degree of saturation. If there is an equality, choose any node of maximal degree in the uncolored sub-graph.
4: **if** All non-neighbors of $\mathcal{S}_i$ are uncolored **then**
5: Color $\mathcal{S}_i$ with the least possible color.
6: **else**
7: Order all the non-neighbors, $\mathcal{S}_j$, of $\mathcal{S}_i$ in $\overline{\mathcal{G}_{\text{con}}}$ in decreasing order of w_{ij} in $\mathcal{G}_{\text{con}}$.
8: Select the lowest ranked node in this list which is colored with a feasible color. Assign the color of this node to current node.
9: **end if**
10: If all the vertices are colored, stop. Else goto 3.

of topology, a real network is expected to have $1 \leq k \leq N$ colors. For coloring the weighted $\overline{\mathcal{G}_{\text{con}}}$, we propose a modified version of the greedy degree-of-saturation (DSATUR) algorithm [38] which accounts for weighted edges. The *degree-of-saturation* of a node is the maximum number of unique colors that can be found in the neighborhood of the node. In this work, we propose the **Weighted-DSATUR** (W-DSATUR) algorithm, presented in Algorithm 2, by incorporating the edge weights w_{ij} of the original connectivity graph to color the weighted complementary connectivity graph.

The W-DSATUR algorithm assigns integers $1, 2, \ldots, k$ to k colors and initializes by coloring the sBS with highest degree in $\overline{\mathcal{G}_{\text{con}}}$ with the color numbered 1. In any given iteration, W-DSATUR assigns to an uncolored node, the lowest numbered color which is available i.e., which is not present in its set of neighbors. For any uncolored node $\mathcal{S}_i$, we rank all its non-neighbors, which are colored with an available color, in decreasing order of the weights from $\mathcal{G}_{\text{con}}$ and then choose to assign the color of the node with highest weight ensuring pairing of neighbors with highest common users. The algorithm gives as an output, the set of colors for every node in $\overline{\mathcal{G}_{\text{con}}}$. Nodes with same color e.g., $\mathcal{S}_1, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5$ in Fig. 4 have a set of common users and for the purpose of collaborative caching, their caches can be grouped together and file placement without repetition can be performed. The total number of colors required to color $\overline{\mathcal{G}_{\text{con}}}$ is given by $\chi(\overline{\mathcal{G}_{\text{con}}})$ i.e., the sBSs can be grouped into $\chi(\overline{\mathcal{G}_{\text{con}}})$ groups.

2) Collaborative Caching within sBS Groups: After grouping, we consider $\chi(\mathcal{G}_{\text{con}})$ different cache placement problems of the form **UNC-2** in (21) within each group. The per-group sub-problems form a class of SAP and are NP-hard. In [37], Fleischer et. al provide $\left(1-\frac{1}{e}\right)-$approximation algorithms for this class of problems which run in exponential time. However, in this work, we use such a local search based polynomial time approach [37].

The placement problem for each sBS can be presented as a knapsack problem (refer to proof of Lemma 2). Knapsacks are a well-known class of NP-hard problems which have finite polynomial time approximation algorithms [5], [35], [39] with approximation ratio $\alpha_{\text{KP}} \in (0,1]$. For the caching problem, $\alpha_{\text{KP}}-$approximate placement yields a reward which is at least α_{KP} fraction of the optimal reward for given CMAB indices $\overline{\Theta}_n$. Based on the existence of an $\alpha_{\text{KP}}-$approximation for the single sBS knapsack problem, a local search based $\epsilon-$greedy cache placement algorithm [37] is presented in Algorithm 3 lines $5-18$ for collaborative caching within each group. For a given color $\mathsf{C} \in \{1,2,\ldots,\chi(\mathcal{G}_{\text{con}})\}$, with N_c sBSs, the local search algorithm decouples the N_c placement problems. Based on the value of ϵ, the algorithm runs $\frac{1}{\alpha_{\text{KP}}} N_c \ln\left(\frac{1}{\epsilon}\right)$ iterations. The algorithm initializes with an empty cache assignment. Within each iteration, for each sBS, the algorithm calculates a $\mathsf{value}_{f,n}$ for each file by assigning to it, the CMAB index if the file is cached at any other sBS within the group and 0 if it is cached at $\mathcal{S}_n$ or not cached at all. Based on this, a marginal value for each file $m_{f,n}$ is calculated as the difference of the CMAB index $\overline{\theta}_{f,n}$ and $\mathsf{value}_{f,n}$. Using this marginal value, the following single sBS knapsack problem is solved at $\mathcal{S}_n$:

$$\textbf{SKP:}\ \max_{\mathbf{c}_n^{\pi}} \sum_{f=1}^{F} |\mathcal{U}(\mathcal{S}_n)| m_{f,n} S_f c_{f,n}^{\pi} \quad \text{s.t.} \sum_{f=1}^{F} c_{f,n}^{\pi} S_f \leq M.$$

An $\alpha_{\text{KP}}-$approximate cache placement $\mathbf{c}_n^{\pi,\mathsf{new}}$ for single sBS **SKP** sub-problem is discussed in the next section. Based on this new cache placement, the marginal value for this solution $\mathcal{M}_n$ is calculated for $\mathcal{S}_n$. Next, difference D_n between the value of current cache placement and the marginal value of new placement is calculated. For the sBS $\mathcal{S}_{n^*}$ with the maximum differential D_{n^*}, the placement is changed to the new one if its $D_{n^*} > 0$ i.e., if the new placement improves the reward. The approximation ratio of the greedy cache placement algorithm is given by the following lemma.

Lemma 3. *Given an $\alpha_{KP}-$approximation for the single sBS cache placement problem **SKP**, the local search based $\epsilon-$greedy cache placement in lines $5-18$ of Algorithm 3 has an approximation ratio of $\left(\frac{\alpha_{KP}}{\alpha_{KP}+1}-\epsilon\right)$ for some $\epsilon > 0$.*

The Lemma directly follows from [37, Theorem 3.1]. Once the greedy search algorithm finishes, a residual cache placement is performed to maximize cache utilization. In this step, for each sBS $\mathcal{S}_n$, if additional cache space is left after initial placement, the most popular files not already in the cache of $\mathcal{S}_n$ are placed there subject to the storage constraint. The reward obtained after this placement can only be greater than or equal to the reward without it. Therefore, the residual cache placement stage potentially improves the approximation ratio of the caching strategy. Next we discuss an $\alpha_{\text{KP}} = 0.5$ approximation of the single sBS knapsack problem **SKP**.

Algorithm 3 UNCODED COLLABORATIVE CACHING

GRAPH COLORING BASED SBS GROUPING:
1: Construct graphs $\mathcal{G}_{\text{con}}$ and $\overline{\mathcal{G}_{\text{con}}}$
2: Color $\overline{\mathcal{G}_{\text{con}}}$ using Algorithm 2 and find $\chi(\overline{\mathcal{G}_{\text{con}}})$.
LOCAL SEARCH BASED APPROXIMATE CACHING:
3: **for** Each color $\mathsf{C} \in \{1,2,\ldots,\chi(\overline{\mathcal{G}_{\text{con}}})\}$ **do**
4: Initialize Cache placement matrix $\mathcal{C}^{\pi} = [\mathbf{0}]_{F \times N_c}$ where N_c is the number of sBSs with color C.
5: **while** $\texttt{loop\_count} \leq \frac{1}{\alpha_{\text{KP}}} N_c \ln\left(\frac{1}{\epsilon}\right)$ **do**
6: Let current cache placement be $\mathcal{C}_{\mathsf{curr}}^{\pi} = [\mathbf{c}_1^{\pi,\mathsf{curr}}, \mathbf{c}_2^{\pi,\mathsf{curr}}, \ldots, \mathbf{c}_{N_c}^{\pi,\mathsf{curr}}]$
7: **for** Each sBS $\mathcal{S}_n$ with color C **do**
8: For each file $f \in \mathbb{F}$, let
$$\mathsf{value}_{f,n} = \begin{cases} \overline{\theta}_{f,n'}, & \text{if } c_{f,n'}^{\pi,\mathsf{curr}} = 1 \ \ \exists\ n' \neq n \\ 0, & \text{if } c_{f,n}^{\pi,\mathsf{curr}} = 1 \ \text{ or } \ f \notin C_{\mathsf{curr}}^{\pi} \end{cases}$$
9: For each file f, let marginal value $m_{f,n} = \overline{\theta}_{f,n} - \mathsf{value}_{f,n}$
10: Solve the **SKP** problem at $\mathcal{S}_n$ to cache files, using value $|\mathcal{U}(\mathcal{S}_n)| m_{f,n} S_f$ and weights S_f.
11: Let $\mathbf{c}_n^{\pi,\mathsf{new}}$ be the new cache placement and $\mathcal{M}_n = \sum_{f \in \mathbb{F}} m_{f,n}\ S_f\ c_{f,n}^{\pi,\mathsf{new}}$ be the marginal value for this solution.
12: **end for**
13: For each sBS $\mathcal{S}_n$ with color C, let $D_n = \mathcal{M}_n - \sum_{f \in \mathbb{F}} \overline{\theta}_{f,n}\ S_f\ c_{f,n}^{\pi,\mathsf{curr}}$
14: Let $\mathcal{S}_{n^*}$ be the sBS s.t. $n^* = \arg\max_n D_n$.
15: **if** $D_{n^*} > 0$ **then**
16: Change the cache placement $\mathbf{c}_{n^*}^{\pi,\mathsf{curr}} \leftarrow \mathbf{c}_{n^*}^{\pi,\mathsf{new}}$
17: **end if**
18: **end while**
19: **end for**
RESIDUAL CACHE PLACEMENT:
20: **for** Each sBS $\mathcal{S}_n \in \mathbb{S}$ **do**
21: **for** All files $\mathbb{F}_n \notin$ cache of $\mathcal{S}_n$ **do**
22: Rank files $f \in \mathbb{F}_n$ in descending order of $\overline{\theta}_{f,n}$.
23: **end for**
24: Place files from $\mathbb{F}_n$ with highest $\overline{\theta}_{f,n}$ in the cache subject to storage constraints
25: **end for**

3) Single sBS Greedy Knapsack: A greedy approximation for the single-sBS knapsack problem is based on a relaxation of the cache placement variable such that $0 \leq c_{f,n}^{\pi} \leq 1$ which reduces the placement problem to a linear program (LP). The solution to the LP is then rounded to give an $\alpha_{\text{KP}} = 0.5$ approximate solution [39] for **SKP**. The solution is as follows:

1. Reorder $m_{f,n}$ such that $m_{i,n} \geq m_{j,n}$ if $i < j, \forall i,j \in \mathbb{F}$ i.e., relabel files in decreasing order of their marginal values. For **SKP** with values $v_i = m_{i,n} S_i$ and weights $w_i = S_i$, $\forall i \in \mathbb{F}$, the re-ordering satisfies the *regularity condition* [39]:
$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_F}{w_F} \triangleq m_{1,n} \geq m_{2,n} \geq \ldots \geq m_{F,n}.$$
As a result, the following assignment assures $\alpha_{\text{KP}} = 0.5$.
2. Place files with highest marginal value $m_{f,n}$ sequentially into the cache of $\mathcal{S}_n$ until capacity of the cache is reached.

The above approximation algorithm is used to solve each **SKP** in Algorithm 3. Furthermore, if $S_f = 1,\ \forall f \in \mathbb{F}$, the above placement is optimal with $\alpha_{\text{KP}} = 1$. Combining the graph coloring based sBS grouping and the per-group greedy caching yields the uncoded collaborative caching algorithm presented in Algorithm 3.

4) Approximation Ratio: Let R^{opt}_{unc} be the reward of the optimal uncoded caching strategy given by the solution to **UNC-1**. Again, consider the reduction into $\chi(\overline{\mathcal{G}_{\text{con}}})$ number of sub-problems of the form **UNC-2** in (21). Let R^{opt}_{C} be the optimal reward for the $\mathsf{C}-$th sub-problem for $\mathsf{C} \in \{1, 2, \ldots, \chi(\overline{\mathcal{G}_{\text{con}}})\}$. Thus we have:

$$R^{opt}_{\mathsf{unc}} \leq \sum_{\mathsf{C}=1}^{\chi(\overline{\mathcal{G}_{\text{con}}})} R^{opt}_{\mathsf{C}}. \tag{22}$$

Let $R^{\mathsf{greedy}}_{\mathsf{C}}$ be the reward of the greedy approximate cache placement in Algorithm 3. Then, we have

$$R^{\mathsf{greedy}}_{\mathsf{C}} \geq \left(\frac{\alpha_{\text{KP}}}{\alpha_{\text{KP}}+1} - \epsilon\right) R^{opt}_{\mathsf{C}}. \tag{23}$$

Using (22) and (23), we have:

$$\sum_{\mathsf{C}=1}^{\chi(\overline{\mathcal{G}_{\text{con}}})} R^{\mathsf{greedy}}_{\mathsf{C}} \geq \left(\frac{\alpha_{\text{KP}}}{\alpha_{\text{KP}}+1} - \epsilon\right) R^{opt}_{\mathsf{unc}}, \tag{24}$$

which gives an approximation guarantee on the greedy algorithm. Adding the final residual placement stage, the uncoded collaborative cache placement can be represented as an $(\alpha, \beta)-$approximation algorithm with $\alpha = \left(\frac{\alpha_{\text{KP}}}{\alpha_{\text{KP}}+1} - \epsilon\right)$ and $\beta = 1$. Since we have $\alpha_{\text{KP}} = 0.5$, we have an approximation ratio of $\alpha = \left(\frac{1}{3} - \epsilon\right)$. In the next section, we discuss coded collaborative cache placement which is an optimal strategy, given the file popularity profile, with $\alpha = \beta = 1$.

C. Coded Collaborative Caching Strategy

In this section we present the coded collaborative caching strategy wherein file segments (fractions of files) can be stored in the caches instead of the entire files. This can be achieved in practice by encoding files using a rateless code like Raptor Codes [2], [34]. The output codewords of the rateless code are *coded packets* of the original file which can be stored by the sBSs in their caches. These coded packets enable the recovery of the original file when a user gathers enough number of such packets. The storing of coded packets is modeled as each sBS storing a fraction of a file and the file is assumed to be recovered when fractions summing to 1 are recovered. The coded cache placement matrix C^{π} for a policy π is an $F \times N$ matrix defined in (4) with elements $c^{\pi}_{f,n} \in [0, 1]$ for $f \in \mathbb{F}$ and $n = 1, \ldots, N$ as in (5). The fraction $c^{\pi}_{f,n}$ is the fraction of coded packets of file f stored at $\mathcal{S}_n$. Note that for the coded caching strategy, $T_{f,n}$ in Algorithm 1 denotes the number of times a *fraction* of file $f \in \mathbb{F}$ has been cached in $\mathcal{S}_n$ upto time t. The CMAB indices, $\overline{\theta}_n, \forall n \in \{1, 2, \ldots, N\}$ in Algorithm 1, are evaluated using this definition of $T_{f,n}$.

The optimal cache placement is again modeled as a reward maximization problem. Based on the per-user reward model from (17), we formulate a related per-user reward for the coded caching strategy. The intuition behind the formulation is as follows: For a user u and file f at time t, we first order the sBSs, $\mathcal{S}_n \in \mathcal{N}(u)$, in descending order of CMAB indices upto time $t-1$, with the first sBS being the one with the highest $\overline{\theta}_{f,n},\ \mathcal{S}_n \in \mathcal{N}(u)$. The amount of data for file f that the user u is able to download from $\mathcal{S}_n$ is denoted by $c^{\pi}_{f,n} S_f$. An expected accumulated reward for this download is given by $c^{\pi}_{f,n} \overline{\theta}_{f,n} S_f$. The expected accumulated reward for a user u that can download file f from the first k sBS's in the ordered list is given by,

$$\begin{aligned}\overline{R}^{f,k}_u &= \sum_{i=1}^{k-1} c^{\pi}_{f,i}\ \overline{\theta}_{f,i} S_f + \left(1 - \sum_{i=1}^{k-1} c^{\pi}_{f,i}\right) \overline{\theta}_{f,k} S_f \\ &= \left[\overline{\theta}_{f,k} - \sum_{i=1}^{k-1} c^{\pi}_{f,i} (\overline{\theta}_{f,k} - \overline{\theta}_{f,i})\right] S_f, \end{aligned} \tag{25}$$

where $k \in \{1, 2, \ldots, |\mathcal{N}(u)|\}$. The per-sBS reward function is similar to (17) in that it only assures the maximum reward for the case when the user downloads an entire file from the sBS where the file has the highest popularity. Since the sBSs are ordered and fractional storage is allowed, download of an entire file from the k most popular sBSs in the user's neighborhood yields the maximum rewards for the smallest k. Thus implicitly, the reward function also discourages replication of file fragments. The reward function is linear (affine) in terms of the placement variables $c^{\pi}_{f,n}$. The file f can be fully downloaded from the k best caches in the list if and only if $\sum_{i=1}^{k} c^{\pi}_{f,i} \geq 1$. The reward for a user u for downloading the file f is thus a piecewise-defined affine function of the placement variables $c^{\pi}_{f,n}$:

$$\overline{R}^{f}_u = \begin{cases} \overline{R}^{f,1}_u & \text{if } c^{\pi}_{f,1} \geq 1 \\ \vdots & \\ \overline{R}^{f,j}_u & \text{if } \begin{array}{l}\sum_{i=1}^{j-1} c^{\pi}_{f,i} < 1 \\ \sum_{i=1}^{j} c^{\pi}_{f,i} \geq 1\end{array} \\ \vdots & \\ \overline{R}^{f,|\mathcal{N}(u)|}_u & \text{if } \sum_{i=1}^{|\mathcal{N}(u)|-1} c^{\pi}_{f,i} < 1 \end{cases} \tag{26}$$

We aim to maximize the sum of minimum expected accumulated rewards for all users and files to determine the optimal cache placement.

Lemma 4. *The per-user reward, $\bar{R}^{f}_u$, is a concave function of the placement matrix C^{π}.*

Proof. The point-wise minimum of a piece-wise affine function is concave [40]. Thus it suffices to show that the per-user reward in (26) can be represented as:

$$\bar{R}^{f}_u = \min_{k \in \{1,2,\ldots,|\mathcal{N}(u)|\}} \bar{R}^{f,k}_u. \tag{27}$$

Suppose that user u downloads the entire file from the first j caches in the list. Then the conditions $\sum_{i=1}^{j-1} c^{\pi}_{f,i} < 1$ and $\sum_{i=1}^{j} c^{\pi}_{f,i} \geq 1$ hold and we have $\bar{R}^{f}_u = \bar{R}^{f,j}_u$. We have to show that $\bar{R}^{f,j}_u \leq \bar{R}^{f,j'}_u\ \ \forall j \neq j'$. Using (25), the condition is given by:

$$\overline{\theta}_{f,j} - \sum_{i=1}^{j-1} c^{\pi}_{f,i} (\overline{\theta}_{f,j} - \overline{\theta}_{f,i}) \leq \overline{\theta}_{f,j'} - \sum_{i=1}^{j'-1} c^{\pi}_{f,i} (\overline{\theta}_{f,j'} - \overline{\theta}_{f,i}) \tag{28}$$

Considering the case for $j' > j$, we can re-write (28) as:

$$\left[\sum_{i=1}^{j} c^{\pi}_{f,i} - 1\right] \left(\overline{\theta}_{f,j} - \overline{\theta}_{f,j'}\right) + \sum_{i=j+1}^{j'-1} c^{\pi}_{f,i} \left(\overline{\theta}_{f,i} - \overline{\theta}_{f,j'}\right) \geq 0 \tag{29}$$

Now we have, $\sum_{i=1}^{j} c_{f,i}^{\pi} \geq 1$ and for all $j^{'} > j$, we have $(\overline{\theta}_{f,j} - \overline{\theta}_{f,j^{'}}) \geq 0$. Also since $i < j^{'}$, the factor $(\overline{\theta}_{f,i} - \overline{\theta}_{f,j^{'}}) \geq 0$. Therefore (29) is satisfied. Next, considering the case of $j^{'} < j$, we can rewrite (28) as:

$$\left[\sum_{i=1}^{j-1} c_{f,i}^{\pi} - 1\right]\left(\overline{\theta}_{f,j^{'}} - \overline{\theta}_{f,j}\right) + \sum_{i=j^{'}+1}^{j-1} c_{f,i}^{\pi}\left(\overline{\theta}_{f,i} - \overline{\theta}_{f,j^{'}}\right) \geq 0. \tag{30}$$

We have $\sum_{i=1}^{j-1} c_{f,i}^{\pi} - 1 \leq 0$ and since $i > j^{'}$, the factor $\left(\overline{\theta}_{f,i} - \overline{\theta}_{f,j^{'}}\right) \leq 0$ which ensures that (30) is satisfied. This completes the proof. ■

The optimal coded cache placement problem can then be formulated as the following convex optimization problem:

$$\max_{\mathcal{C}^{\pi}} \quad \sum_{u=1}^{U}\sum_{f=1}^{F} \min_{k\in\{1,2,\ldots,|\mathcal{N}(u)|\}} \overline{R}_u^{f,k}$$

$$\text{s.t.} \quad \sum_{f=1}^{F} c_{f,n}^{\pi} \cdot S_f \leq M, \quad \forall n \quad \text{and} \quad c_{f,n}^{\pi} \in [0,1]. \tag{31}$$

The optimization in (31) can be reduced to the following LP by introducing the auxiliary variable y_u^f [2]. The resulting optimization problem is given by,

$$\max_{\mathcal{C}^{\pi}} \sum_{u=1}^{U}\sum_{f=1}^{F} \mathrm{y}_u^f \quad \text{s.t.} \quad \mathrm{y}_u^f \leq \overline{R}_u^{f,k}, \ \forall k \in \{1,2,\ldots,|\mathcal{N}(u)|\}$$

$$\sum_{f=1}^{F} c_{f,n}^{\pi} \cdot S_f \leq M, \quad \forall n, \text{ and } c_{f,n}^{\pi} \in [0,1]. \tag{32}$$

Since the LP in (32) can be solved optimally, the coded cache placement problem gives an $(\alpha, \beta)-$approximate placement for the CCP policy π with $\alpha = \beta = 1$ i.e., *given the CMAB indices* the solution is optimal. The formulation in (32) can be considered as a convex relaxation of the uncoded caching problem **UNC-1** in the sense that the $\{0,1\}$ binary assignment of the uncoded scheme is also a feasible solution to the coded caching problem. As a result, the average accumulated reward obtained with the coded cache placement for a given popularity index $\overline{\theta}_{f,n}$, encompasses the uncoded case. The caveat however, is that the convergence to the optimal reward value is also dependent on the accuracy of the CMAB based distributed learning and the network connectivity. In the next section, numerical results are presented for the proposed collaborative caching framework with discussions on impact of coded and uncoded caching on the learning accuracy.

VI. Numerical Results

In this section we present simulation results under different network configurations to show the performance of the proposed collaborative caching strategies. In order to compare results, and present insight on multi-sBS edge caching networks, we first outline two baseline caching strategies:

- *Uncoded Local Caching Strategy:* In this strategy, uncoded cache placement is performed locally at each sBS. In this procedure, every sBS learns the file popularity profile from its own set of connected users without taking into account the overall network connectivity i.e., topology-aware $T_{f,n}$ update, as discussed in Remark 1, is not performed. The exploration-exploitation in this baseline scheme is based only on the local cache placement. This scheme entails the solving following knapsack problem at each sBS $\mathcal{S}_n$:

$$\max_{\mathcal{C}^{\pi}\in\{0,1\}} \quad \sum_{f=1}^{F} \overline{\theta}_{f,n} S_f c_f^{\pi} \quad \text{s.t.} \quad \sum_{f=1}^{F} c_f^{\pi} S_f \leq M, \quad \forall n. \tag{33}$$

  This scheme was proposed and analysed in [5] for a single sBS caching system.
- *Coded Local Caching Strategy:* Coded local caching strategy is a relaxation of uncoded local caching and the learning process is again topology agnostic. This scheme entails the solution of the following linear program at each sBS $\mathcal{S}_n$:

$$\max_{\mathcal{C}^{\pi}\in[0,1]} \quad \sum_{f=1}^{F} \overline{\theta}_{f,n} S_f c_f^{\pi} \quad \text{s.t.} \quad \sum_{f=1}^{F} c_f^{\pi} S_f \leq M, \quad \forall n. \tag{34}$$

We compare the performance of the proposed schemes to the baselines to highlight the advantage of collaborative learning over local learning in a multi-sBS setting. In order to obtain results in a reasonable time-frame, we consider a $N = 5$ sBS setting with $K = 30$ users in the system. There is a file library of $F = 30$ files $\mathbb{F}$, with file sizes $S_f \in \{1, 3, 5, 7, 9\}$ units. The entire library has a size of 150 units. The cache size of each sBS is $M = 15$ units, which is 10% of the entire library. We assume that the users randomly request files at each time step t. The requests are generated i.i.d from a Zipf distribution (as in (1)) with $\gamma \in \{0.56, 2\}$. Furthermore, we consider two collaborative caching strategies for comparison -

- *Absolute Upper Bound:* This strategy assumes that the instantaneous user demands are known apriori at the central BS and represents the optimal strategy in terms of reward maximization. It gives an upper bound on our learning based caching strategy.
- *Informed Collaborative Caching:* In this strategy, we consider that the popularity distribution i.e., the Zipf γ value at each sBS is known apriori. Based on this the use of the proposed caching strategies in the previous section with the known popularity distribution, instead of the CMAB indices, as input leads to a distribution optimal caching strategy. The reward of the learning-aided strategy should converge to the reward of this stragety over time.

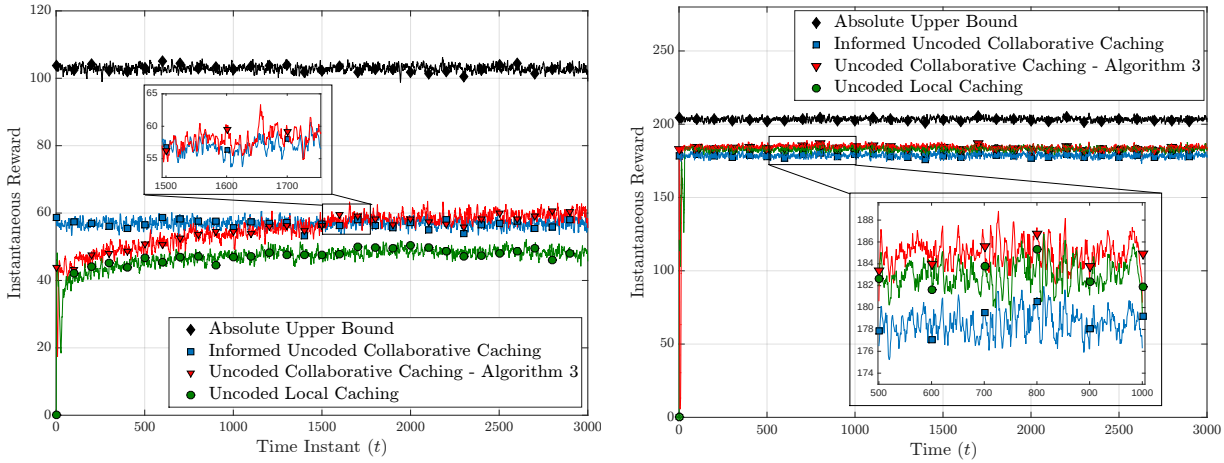
For comparison of different schemes, we consider the metric of *instantaneous reward* at time t. The instantaneous reward is the total amount of data downloaded from the sBS caches to serve the requests of the users and is a measure of the cache hits at each instant. For the uncoded caching schemes, it is defined as

$$R_{\text{unc}}^{t} = \sum_{u\in\mathbb{U}}\sum_{f\in\mathbb{F}} \mathbb{1}_{\{f\in\mathcal{N}(u)\}} \cdot d_f^{t,u} \cdot S_f, \tag{35}$$

where, $d_f^{t,u} = 1$ for the file f requested by user u and $\mathbb{1}_{\{f\in\mathcal{N}(u)\}}$ is the indicator function and is equal to 1 when file f is cached in a sBS in the neighborhood of user u. For the coded caching strategy, the instantaneous reward is defined as:

$$R_{\text{cod}}^{t} = \sum_{u\in\mathbb{U}}\sum_{f\in\mathbb{F}} d_f^{t,u} \cdot \max\left\{1, \sum_{n\in\mathcal{N}(u)} c_{f,n}^{\pi}\right\} \cdot S_f, \tag{36}$$

which accounts for the fact that a user u either downloads the entire file or the sum of fractions (≤ 1) placed in the



(a) (b)

Fig. 5. Performance of Uncoded Caching for (a) $\gamma = 0.56$ and (b) $\gamma = 2$.

caches of sBSs in its neighborhood $\mathcal{N}(u)$. Using this metric, we study the multi-sBS network under different connectivity profiles and caching strategies. To this end, we define three network topologies namely (i) *sparse connectivity* - every user is connected to at least 1 and at most 2 sBSs; (ii) *moderate connectivity* - every user is connected to at least 2 and at most 3 sBSs and (iii) *dense connectivity* - every user connects to at least 3 and at most 5 sBSs.

A. Uncoded Caching Strategies

We first consider a network scenario with moderate connectivity. Fig. 5(a) shows the simulation results for uncoded caching for a file popularity distribution with $\gamma = 0.56$ i.e., the distribution is not skewed and there are many popular files in the system. Algorithm 3 is used for the caching strategy at each time step for collaborative caching with an epsilon value of $\epsilon = 0.01$. We also assume a worst-case $\alpha_{\text{KP}} = 0.5$ for the iterations in the greedy placement. It can be seen from the results, that the collaborative learning based caching converges to the informed collaborative caching strategy i.e., the learning effectively converges to the true file popularity distribution. As expected, the absolute upper bound outperforms all other schemes. Finally, the collaborative caching strategy clearly out-performs the local caching strategy.

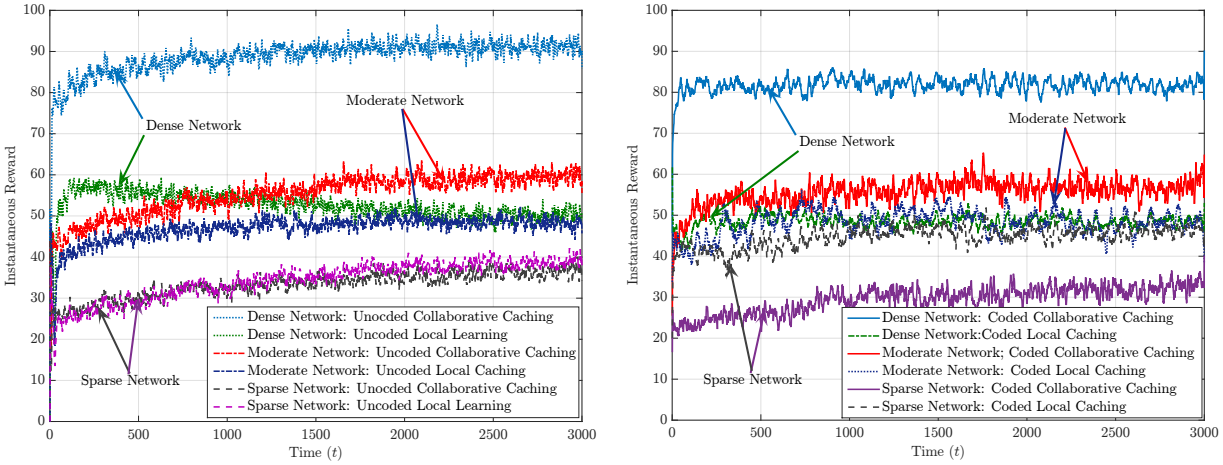
Fig. 5(b) shows the simulation results for uncoded caching for a popularity distribution with $\gamma = 2$ i.e., when there are very few popular files in the system. In this case, the learning converges faster the cumulative reward is higher in comparison with the previous case of $\gamma = 0.56$. The reward is closer to the absolute upper bound i.e., the sub-optimality gap or regret is small as discussed in Example 1 for skewed popularity profiles. Furthermore, in this case, both the local and collaborative caching strategies perform better than the informed upper bound. This is due to the exploration of new files in the CMAB based learning which allows the placement of new files in the caches, in addition to the most popular ones, which can serve some non-popular demands as well. We observe that, for very few popular files in the system, the performance of the local scheme is almost as good as the collaborative caching scheme. In general, $\gamma = 0.56$ i.e., having a large number of popular files is a more realistic network parameter for cache aided systems [5], [5] and we present further results for this network parameter.

Next, we compare the performance of the uncoded collaborative caching strategies under the different network topologies e.g., sparse, moderate and dense. Fig. 6(a) shows the comparison of the collaborative and local learning strategies under the three network settings. It can be seen that the collaborative caching scheme improves upon the local learning based scheme when network connectivity is dense. Under a sparse setting, users are mostly served by single sBSs and the network reduces to a one similar to Network Example 1 in Fig. 3. Under such a setting, the collaboration among sBSs to jointly cache content does not offer any added advantage.

B. Coded Caching Strategies

Next, we study the performance of the coded caching strategies. Fig. 6(b) illustrates the rewards obtained by using coded strategies over the same three network configurations as before. In this case, it can be seen that similar performance trends exist i.e., the collaborative caching schemes outperform the local learning schemes under denser network settings. A comparison of the rewards in the coded and uncoded cases shows that the greedy algorithm for the uncoded caching performs similar to the coded caching case.

Remark 2 (*Learning for Coded Caching*)**.** Coded cache placement is a relaxation of the uncoded placement in the sense that uncoded caching is also a viable solution for the coded formulation. Thus, given the same CMAB indices $\overline{\theta}_{f,n}$ at any time instant t, coded caching should outperform the uncoded scheme in terms of sum reward. However, the reward value, to which a strategy converges over time, is not only dependent on the optimality of the cache placement at every instant. It is also dependent on the effectiveness of the CMAB based learning (Algorithm 1) in conjunction with the caching strategy. From our simulations, we observe that both the coded and uncoded caching schemes converge to very similar instantaneous rewards. It is interesting to note that a simpler local search with no sub-packetization (no rateless coding) performs close to a provably optimal coded caching strategy for most network settings of interest. This stems from the fact that learning the popularity profile over time through



(a) (b)

Fig. 6. Performance under different network configurations (a) Uncoded Caching and (b) Coded Caching.

fractional placement (i.e., sampling fractions of files to learn their popularity distributions) is more difficult than learning based on caching entire files. This in turn offsets the reward gains offered by coded caching over the uncoded scheme. ■

For coded caching, collaborative learning and placement is particularly detrimental in the sparse setting. In this case, the exploration based on collective topology leads to consistently lower rewards. Fig. 7(a) and 7(b) show the performance of the coded and uncoded caching schemes for local and collaborative learning under the sparse network setting. In this case, the collaborative learning leads to very similar performance for both coded and uncoded schemes. However, for the sparse network, local learning suffices. In fact, it can be seen from Fig. 7(b), that the coded local caching, which is a direct linear relaxation of the uncoded local learning, outperforms all schemes in this setting. Note that the uncoded local caching is a naive multi-sBS extension of the scheme presented in [5]. The proposed collaborative caching framework lends itself to easy adaptation, based on network configuration, by enabling the sBSs to change the updating of the $T_{f,n}$ parameter in the CMAB index. Local learning based updates can be used when connectivity is sparse while collaborative learning can be used for moderate to dense connectivity.

Finally we study the performance of all the schemes under the more realistic moderately connected network setting when the cache size at the sBS changes. We plot the instantaneous reward values (averaged over 500 time steps) to which the algorithms converge after 3000 initial learning time steps[3]. From Fig. 7(c), it can be seen that the collaborative caching schemes outperform the local learning based schemes in terms of reward. However, when the cache size is particularly small ($\leq 5\%$ of the library size), the coded caching schemes outperform both the uncoded schemes. This is due to the fact that when cache size is very small, the coded schemes offer flexibility by allowing fractional storage of very popular files. Interestingly, at $M = 5$, the coded local scheme outperforms the coded collaborative scheme. This stems from the fact that at such small cache sizes, accounting for topology-aware placement leads to fractional storage of files which might not directly contribute to the sBS's reward. Thus, when cache size is less than 5%, coded local learning offers the highest rewards.

[3]The convergence time in a real network application would be of the order of a few hours to a day. Once converged, caching can be performed till popularity changes again when a new learning phase should begin.

VII. Conclusions

In this paper, we presented a novel topology-aware collaborative cache placement framework for a multi-sBS small cell network using a reinforcement learning perspective. The multi-armed bandit based learning was shown to have a regret scaling logarithmically with time. We proposed two cache placement strategies: uncoded and coded collaborative caching for the multi-sBS setting. The uncoded cache placement strategy was shown to be NP-hard and a novel graph coloring based polynomial time approximation algorithm was proposed. A linear relaxation to the uncoded problem, namely, the collaborative coded cache placement problem was formulated and was shown to be optimal for a given popularity distribution. Through numerical simulations, collaborative caching was shown to outperform the naive local learning based schemes for network topologies of interest. It was also demonstrated that the sub-optimal greedy uncoded caching performed close to the optimal coded cache placement due to the fact that learning file popularity through fractional placement was harder as compared to learning by caching entire files.

References

[1] F. Boccardi, R. Heath, A. Lozano, T. Marzetta, and P. Popovski, "Five Disruptive Technology Directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, February 2014.

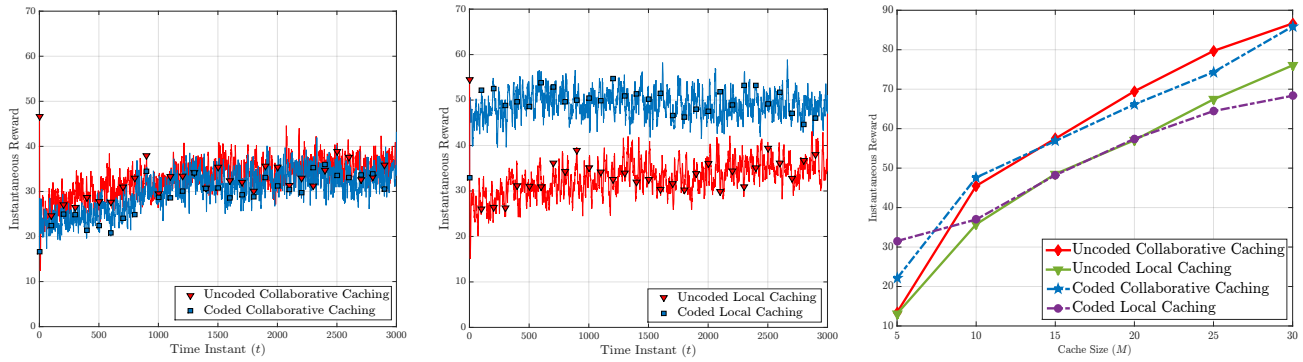
[2] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers," *IEEE Transactions on Information Theory*, vol. 59(12), pp. 8402–8413, Dec. 2013.

[3] M. Ji, G. Caire, and A. F. Molisch, "Optimal Throughput-Outage Trade-off in Wireless One-Hop Caching Networks," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, July 2013, pp. 1461–1465.

[4] ——, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 1, pp. 176–189, Jan 2016.

[5] P. Blasco and D. Gündüz, "Learning-Based Optimization of Cache Content in a Small Cell Base Station," in *Proc. IEEE International Conference on Communications (ICC)*, June 2014, pp. 1897–1903.

[6] ——, "Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds," *arXiv:1407.6154*, 2014.



(a) (b) (c)

Fig. 7. Performance under sparse network: (a) Collaborative Caching and (b) Local learning based Caching. (c) Performance under varying cache sizes.

[7] S. Mller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *Proc. IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–7.

[8] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. PP, no. 99, pp. 1–1, July 2016.

[9] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning Distributed Caching Strategies in Small Cell Networks," *The Eleventh International Symposium on Wireless Communication Systems (ISWCS)*, pp. 917–921, Aug 2014.

[10] E. Bastug, M. Bennis, and M. Debbah, "Think Before Reacting: Proactive Caching in 5G Small Cell Networks," *Towards 5G: Applications, Requirements and Candidate Technologies, Wiley, 2015 (Submitted)*, September 2014.

[11] E. Batug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Proc. International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2015, pp. 161–166.

[12] B. B. Nagaraja and K. G. Nagananda, "Caching with unknown popularity profiles in small cell networks," in *Proc. IEEE Global Communications Conference*, Dec 2015, pp. 1–6.

[13] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A Learning-Based Approach to Caching in Heterogenous Small Cell Networks," *arXiv: 1508.03517*, 2015. [Online]. Available: http://arxiv.org/abs/1508.03517

[14] M. A. Maddah-Ali and U. Niesen, "Fundamental Limits of Caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[15] A. Sengupta, R. Tandon, and T. C. Clancy, "Fundamental Limits of Caching with Secure Delivery," *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 355–370, Feb 2015.

[16] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for heterogeneous wireless networks with multi-level access," *CoRR*, vol. abs/1404.6560, 2014. [Online]. Available: http://arxiv.org/abs/1404.6560

[17] M. A. Maddah-Ali and U. Niesen, "Cache aided interference channels," in *Proc. IEEE International Symposium on Information Theory*, June 2015, pp. 809–813.

[18] A. Sengupta, R. Tandon, and O. Simeone, "Cloud and cache-aided wireless networks: Fundamental latency trade-offs," *arXiv: 1605.01690*, May 2016. [Online]. Available: http://arxiv.org/abs/1605.01690

[19] N. Naderializadeh, M. A. Maddah-Ali, and A. Salman Avestimehr, "Fundamental limits of cache-aided interference management," *arXiv:1602.04207*, Feb. 2016. [Online]. Available: http://arxiv.org/pdf/1602.04207v1

[20] U. Niesen and M. A. Maddah-Ali, "Coded Caching with Nonuniform Demands," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2014, pp. 221–226.

[21] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," in *Proc. Information Theory and Applications Workshop (ITA)*, Feb 2015, pp. 98–107.

[22] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *arxiv:1502.03124*, 2015. [Online]. Available: http://arxiv.org/abs/1502.03124

[23] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless Video Content Delivery through Coded Dstributed Caching," in *Proc. IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 2467–2472.

[24] S. Bubeck and N. Cesa-Bianchi, "Regret Analysis of Stochastic and Non-Stochastic Multi-armed Bandit Problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–22, 2012.

[25] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, May 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1013689704352

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

[27] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial multi-armed bandit and its extension to probabilistically triggered arms," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1746–1778, Jan. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=2946645.2946695

[28] Y. Zhou and X. Li, "Multi-armed bandits with combinatorial strategies under stochastic bandits," *arXiv:1307.5438*, 2013. [Online]. Available: http://arxiv.org/abs/1307.5438

[29] M. Hefeeda and O. Saleh, "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447–1460, Dec. 2008.

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

[31] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501 – 514, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128608003423

[32] C. Szepesvári, "The asymptotic convergence-rate of q-learning," in *Proc. Conference on Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1998, pp. 1064–1070. [Online]. Available: http://dl.acm.org/citation.cfm?id=302528.302898

[33] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning-aided collaborative caching in small cell networks," *(Extended Version)*. [Online]. Available: https://filebox.ece.vt.edu/~aviksg/lcc.pdf

[34] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[35] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, University of Copenhagen, 1995.

[36] D. B. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 62, no. 1-3, pp. 461–474, 1993. [Online]. Available: http://dx.doi.org/10.1007/BF01585178

[37] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *Proc. ACM-SIAM Symposium on Discrete Algorithm*. Society for Industrial and Applied Mathematics, 2006, pp. 611–620.

[38] D. Brélaz, "New Methods to Color the Vertices of a Graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, Apr. 1979. [Online]. Available: http://doi.acm.org/10.1145/359094.359101

[39] A. Korbut and I. Sigal, "Exact and greedy solutions of the knapsack problem: the ratio of values of objective functions," *International Journal of Computer and Systems Sciences*, vol. 49, no. 5, pp. 757–764, 2010. [Online]. Available: http://dx.doi.org/10.1134/S1064230710050102

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

APPENDIX A
PROOF OF LEMMA 1

The proof is based on the proof of the upper bound on CMAB regret presented in [26]. First, we present the Chernoff-Hoeffding Lemma which is integral to the proof.

Lemma 5 (Chernoff-Hoeffding Bound)**.** *Let $X_1, X_2, \ldots, X_n$ be random variables with common support $[0,1]$ and $\mathbb{E}[X_i] = \theta$. Let $S_n = \sum_{i=1}^{n} X_i$. Then for all $t \geq 0$, we have*
$\Pr[S_n \geq n\theta + t] \leq e^{-2t^2/n}$ and $\Pr[S_n \leq n\theta + t] \leq e^{-2t^2/n}$.

A. Popularity Estimation Process

Let $T_{f,n}^t$ be the value of the variable $T_{f,n}$ after t rounds i.e., the number of times file f is cached at $\mathcal{S}_n$ after t rounds is $T_{f,n}^t$. Also let $\widehat{\theta}_{f,n,s}$ be the value of the variable $\widehat{\theta}_{f,n}$ after the file $f \in \mathbb{F}$ has been cached s times. The estimation of popularity profile is assumed to be *well behaved* if, at time t, the empirical popularity estimate $\widehat{\theta}_{f,n}$ is close to the actual expectation $\theta_{f,n}$ i.e., if

$$|\widehat{\theta}_{f,n,T_{f,n}^{t-1}} - \theta_{f,n}| < \sqrt{\frac{\Psi_{f,n} \log U_n t}{2T_{f,n}^{t-1}}}, \quad \forall f \in \mathbb{F} \tag{37}$$

where $U_n = |\mathcal{U}(\mathcal{S}_n)|$. Let $\mathcal{P}_t$ be the event such that it is true when the estimation process satisfies (37). Using Chernoff-Hoeffding Bound, for any $f \in \mathbb{F}$, we have:

$$\begin{aligned}
&\Pr\left[|\widehat{\theta}_{f,n,T_{f,n}^{t-1}} - \theta_{f,n}| \geq \sqrt{\frac{\Psi_{f,n} \log U_n t}{2T_{f,n}^{t-1}}}\right] \\
&= \sum_{s=1}^{t-1} \Pr\left[\left\{|\widehat{\theta}_{f,n,s} - \theta_{f,n}| \geq \sqrt{\frac{\Psi_{f,n} \log U_n t}{2s}}, T_{f,n}^{t-1} = s\right\}\right] \\
&\leq \sum_{s=1}^{t-1} \Pr\left[\left\{|\widehat{\theta}_{f,n,s} - \theta_{f,n}| \geq \sqrt{\frac{\Psi_{f,n} \log U_n t}{2s}}\right\}\right] \\
&\leq 2te^{-\Psi_{f,n} \log U_n t} = 2U_n^{-\Psi_{f,n}} t^{1-\Psi_{f,n}}.
\end{aligned} \tag{38}$$

Taking an union bound on f, we have

$$\begin{aligned}
\Pr[\mathcal{P}_t] &= \Pr\left[\forall f \in \mathbb{F}, |\widehat{\theta}_{f,n,T_{f,n}^{t-1}} - \theta_{f,n}| < \sqrt{\frac{\Psi_{f,n} \log U_n t}{2T_{f,n}^{t-1}}}\right] \\
&\leq 1 - 2FU_n^{-\Psi_{f,n}} t^{1-\Psi_{f,n}}.
\end{aligned} \tag{39}$$

The value of $\widehat{\theta}_{f,n}$ at the end of time step t is $\widehat{\theta}_{f,n,T_{f,n}^t}$ since by definition file f is cached $T_{f,n}^t$ times after t time steps. Also, for $\overline{\theta}_{f,n}$, let $\overline{\theta}_{f,n,t}$ be its value after t rounds and $\overline{\Theta}_{n,t} = [\overline{\theta}_{1,n,t}, \overline{\theta}_{2,n,t}, \ldots, \overline{\theta}_{F,n,t}]$ be the input to the CCP solver at round t. Then, from Algorithm 1, we have:

$$\overline{\theta}_{f,n,t} = \widehat{\theta}_{f,n,T_{f,n}^{t-1}} + \sqrt{\frac{\Psi_{f,n} \log (U_n t)}{2T_{f,n}^{t-1}}}. \tag{40}$$

B. α-sub-optimal Caching Strategy:

A caching strategy $\mathbf{c}_n^\pi(t)$ is defined to be α-*sub-optimal* if the reward obtained by the strategy is less than α fraction of the optimal reward:

$$R_{\Theta_n}(\mathbf{c}_n^\pi(t)) < \alpha \cdot R_{\Theta_n}^{\text{opt}}.$$

The set of all α-sub-optimal caching strategies for $\mathcal{S}_n$ is denoted by:

$$\mathbb{C}_n^B = \{\mathbf{c}_n^\pi(t) | R_{\Theta_n}(\mathbf{c}_n^\pi(t)) < \alpha \cdot R_{\Theta_n}^{\text{opt}}\}.$$

Let the set of all K_f α-sub-optimal caching strategies at $\mathcal{S}_n$ in which the file $f \in \mathbb{F}$ is cached be denoted by

$$\mathbb{C}_{f,n}^B = \{C_{f,n}^{B,i}, \forall i = 1, \ldots, K_f\}$$

Without loss of generality, the strategies $C_{f,n}^{B,i}$ are reordered in increasing order of expected rewards $C_{f,n}^{B,1}, C_{f,n}^{B,2}, \ldots, C_{f,n}^{B,K_f}$, such that $C_{f,n}^{B,K_f}$ is strategy which yields the best expected reward. We define:

$$\Delta_n^{f,j} = \alpha \cdot R_{\Theta_n}^{\text{opt}} - R_{\Theta_n}(C_{f,n}^{B,j}) \tag{41}$$

Based on this, we further define:

$$\Delta_{n,\max}^f = \Delta_n^{f,1} \text{ and } \Delta_{n,\min}^f = \Delta_n^{f,K_f}.$$

For any underlying arm (file) $f \in \mathbb{F}$, similar to [26], we define:

$$\Delta_{\max} = \max_{f \in \mathbb{F}} \left[\alpha \cdot \mathsf{R}_{\Theta_n}^{\text{opt}} - \min\{\mathsf{R}_{\Theta_n}(\mathbf{c}_n) | \mathbf{c}_n \in \mathbb{C}_B, f \in \mathbf{c}_n\}\right]$$

$$\Delta_{\min} = \max_{f \in \mathbb{F}} \left[\alpha \cdot \mathsf{R}_{\Theta_n}^{\text{opt}} - \max\{\mathsf{R}_{\Theta_n}(\mathbf{c}_n) | \mathbf{c}_n \in \mathbb{C}_B, f \in \mathbf{c}_n\}\right].$$

Intuitively, $\Delta_{\max}$ is the difference in expected reward between the optimal and the reward for playing the worst α-*sub-optimal* super-arm, while $\Delta_{\min}$ is difference with the optimal for the case of playing the best possible α-*sub-optimal* super-arm. If all arms (files) are sampled sufficiently w.r.t $\Delta_{\min}$, then the sample means $\widehat{\theta}_{f,n}$ are close to their true means and the probability of the algorithm playing a α-*sub-optimal* super-arm $\mathbf{c}_n \in \mathbb{C}_B$ is low. However, if the sampling is insufficient, then we incur a regret proportional to $\Delta_{\max}$.

C. Sampling of Cached Files

For the proof, at each sBS $\mathcal{S}_n$, a counter N_f is maintained for each file $f \in \mathbb{F}$ after the F initial rounds (when each file is sequentially placed in the cache to initialize the algorithm). Let $N_{f,t}$ be the value of the counter after t time instants. Thus $N_{f,F} = 1$ and $\sum_{f \in \mathbb{F}} N_{f,F} = F$. The counters are updated as follows: For any instant $t > F$, if $\mathbf{c}_n^\pi(t) \in \mathbb{C}_n^B$, then $f^* = \arg\min_{j \in \mathbf{c}_n^\pi(t)} N_{j,t-1}$ and we increment the counter N_{f^*} by 1 i.e., $N_{f^*,t} = N_{f^*,t-1} + 1$. If f^* is not unique, any random file f with the smallest counter in $\mathbf{c}_n^\pi(t)$ is picked and its counter is incremented. In every α-sub-optimal caching round, exactly one counter is incremented. The counter N_f is further sub-divided into counters $\{N_f^l\}_{l=1}^{K_f}$, whose value at a round $t = T$ is defined as

$$N_{f,T}^l = \sum_{t=F+1}^{T} \mathbb{I}\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}\}, \ \forall l \in [K_f], \tag{42}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function such that $\mathbb{I}\{x\} = 1$ iff x is true. We define

$$\ell_T(\Delta) = \frac{2\Psi_{f,n} \log U_n T}{(g^{-1}(\Delta))^2}, \tag{43}$$

which is the number of samplings that is considered *sufficient* for a caching strategy which yields a reward Δ away from the $\alpha-$approximation w.r.t time horizon T. If $N_{f,t-1} > \ell_T(\Delta^{f,l})$, the α-sub-optimal strategy $C_{f,n}^{B,l}$ is *sufficiently sampled*. Otherwise it is *under-sampled*. Thus, we have the number of sufficiently sampled and under-sampled periods given in (44) and (45). Thus we have,

$$N_{f,T} = 1 + \sum_{l \in [K_f]} (N_{f,T}^{l,\text{suf}} + N_{f,T}^{l,\text{und}}). \tag{46}$$

$$N_{f,T}^{l,\text{suf}} = \sum_{t=F+1}^{T} \mathbb{I}\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1} > \ell_T(\Delta_n^{f,l})\} \tag{44}$$

$$N_{f,T}^{l,\text{und}} = \sum_{t=F+1}^{T} \mathbb{I}\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, N_{f,t-1} < \ell_T(\Delta_n^{f,l})\} \tag{45}$$

$$\sum_{l\in[K_f]} N_{f,T}^{l,\text{und}} \cdot \Delta_n^{f,l} = \sum_{t=F+1}^{T} \sum_{l\in[K_f]} \mathbb{I}\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, N_{f,t-1} < \ell_T(\Delta_n^{f,l})\} \cdot \Delta_n^{f,l} \tag{58}$$

$$= \sum_{t=F+1}^{T} \sum_{l\in[K_f]} \sum_{j=1}^{l} \mathbb{I}\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, N_{f,t-1} \in (\ell_T(\Delta_n^{f,j-1}), \ell_T(\Delta_n^{f,j})]\} \cdot \Delta_n^{f,l} \tag{59}$$

$$\leq \sum_{t=F+1}^{T} \sum_{l\in[K_f]} \sum_{j\in[K_f]} \mathbb{I}\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, N_{f,t-1} \in (\ell_T(\Delta_n^{f,j-1}), \ell_T(\Delta_n^{f,j})]\} \cdot \Delta_n^{f,j} \tag{60}$$

$$\leq \sum_{j\in[K_f]} \sum_{t=F+1}^{T} \mathbb{I}\{\mathbf{c}_n^\pi(t) \in \mathbb{C}_{f,n}^{B}, N_{f,t} > N_{f,t-1}, N_{f,t-1} \in (\ell_T(\Delta_n^{f,j-1}), \ell_T(\Delta_n^{f,j})]\} \cdot \Delta_n^{f,j} \tag{61}$$

$$\leq \sum_{j\in[K_f]} \left[\ell_T(\Delta_n^{f,j}) - \ell_T(\Delta_n^{f,j-1})\right] \cdot \Delta_n^{f,j} = \ell_T(\Delta_n^{f,K_f})\Delta_n^{f,K_f} + \sum_{j=1}^{K_f-1} \ell_T\left[\Delta_n^{f,j})(\Delta_n^{f,j} - \Delta_n^{f,j+1}\right] \tag{62}$$

$$\leq \ell_T(\Delta_{n,\min}^{f}) \cdot \Delta_{n,\min}^{f} + \ell_T(\Delta_{n,\min}^{f}) \left[\Delta_{n,\max}^{f,K_f} - \Delta_{n,\min}^{f,K_f}\right] = \ell_T(\Delta_{n,\min}^{f}) \cdot \Delta_{n,\max}^{f}. \tag{63}$$

The total reward at round T is at least

$$T\alpha R_{\Theta_n}^{\text{opt}} - \mathbb{E}\left[\sum_{f\in\mathbb{F}} \left\{\Delta_n^{f,1} + \sum_{l\in[K_f]} (N_{f,T}^{l,\text{suf}} + N_{f,T}^{l,\text{und}}) \cdot \Delta_n^{f,l}\right\}\right] \tag{47}$$

D. Upper Bound on $N_{f,T}^{l,suf}$

Define $\Lambda_{f,t} = \sqrt{\frac{\Psi_{f,n}\log(U_n t)}{2T_{f,n}^{t-1}}}$ which is a random variable since $T_{f,n}^{t-1}$ is a random variable. We have $\Lambda_t = \max\{\Lambda_{f,t} | f \in \mathbf{c}_n^\pi(t)\}$. Further, we define $\Lambda^{f,l} = \sqrt{\frac{\Psi_{f,n}\log(U_n t)}{2\ell_T(\Delta_n^{f,l})}}$. Let $\mathcal{B}_t$ be the event that the $(\alpha,\beta)-$approximation solver running in the CCP fails to produce an $\alpha-$approximate solution w.r.t $\overline{\Theta}_{n,t}$ in round t. Also, let $\neg\mathcal{B}_t$ be the event when an the solver does produce an $\alpha-$approximate solution. Thus, by the definition of the $(\alpha,\beta)-$approximation solver, we have:

$$\Pr(\mathcal{B}_t) = \mathbb{E}[\mathbb{I}\{\mathcal{B}_t\}] \leq 1 - \beta.$$

Based on (40), the following holds true:

$$\mathcal{P}_t \Rightarrow \overline{\theta}_{f,n,t} - \theta_{f,n} > 0, \ \forall f \in \mathbb{F} \tag{48}$$

$$\mathcal{P}_t \Rightarrow \overline{\theta}_{f,n,t} - \theta_{f,n} < 2\Lambda_{f,t}, \ \forall f \in \mathbf{c}_n^\pi(t) \tag{49}$$

Again, we have $\forall f \in \mathbb{F}$, $\forall l \in [K_f]$, and $\forall f^{'} \in \mathbf{c}_n^\pi(t)$:

$$\left\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, N_{f^{'},t-1} > \ell_T(\Delta_n^{f,l})\right\} \Rightarrow \Lambda^{f,l} > \Lambda_{f,t}. \tag{50}$$

For any $f \in \mathbb{F}, l \in [K_f]$, if $\left\{\mathcal{P}_t, \neg\mathcal{B}_t, \mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, N_{f^{'},t-1} > \ell_T(\Delta_n^{f,l})\right\}$ holds at time t, the following holds true:

$$R_{\Theta_n}(\mathbf{c}_n^\pi(t)) + g(2\Lambda^{f,l}) > R_{\Theta_n}(\mathbf{c}_n^\pi(t)) + g(2\Lambda_{f,t}) \tag{51}$$

$$\geq R_{\overline{\Theta}_{n,t}}(\mathbf{c}_n^\pi(t)) \geq \alpha R_{\overline{\Theta}_{n,t}}^{\text{opt}} \geq \alpha R_{\Theta_n}^{\text{opt}}, \tag{52}$$

where the first inequality follows from the strict monotonicity of $g(\cdot)$ and (50), the second follows from the bounded smoothness property and (49). The third inequality is true since $\neg\mathcal{B}_t \Rightarrow \mathbf{c}_n^\pi(t)$ is an $\alpha-$approximation w.r.t $\overline{\Theta}_{n,t}$. Thus we have

$$R_{\Theta_n}(C_{f,n}^{B,l}) + g(2\Lambda^{f,l}) > \alpha R_{\Theta_n}^{\text{opt}} \tag{53}$$

Now,

$$\ell_T(\Delta_n^{f,l}) = \frac{2\Psi_{f,n}\log U_n T}{(g^{-1}(\Delta_n^{f,l}))^2}$$

and we have

$$2\Lambda^{f,l} = g^{-1}(\Delta_n^{f,l})\sqrt{\frac{\log t}{\log T}} \Rightarrow g(2\Lambda^{f,l}) \leq \Delta_n^{f,l}.$$

Thus, (52) contradicts (41) i.e.,

$$\Pr\Big\{\mathcal{P}_t, \neg\mathcal{B}_t, \mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, \forall f^{'} \in \mathbf{c}_n^\pi(t), N_{f^{'},t-1} > \ell_T(\Delta_n^{f,l})\Big\} = 0, \quad \forall f \in \mathbb{F}, l \in [K_f] \tag{54}$$

$$\Rightarrow \Pr\Big\{\mathcal{P}_t, \neg\mathcal{B}_t, \exists f \in \mathbb{F}, \exists l \in [K_f], \mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, \forall f^{'} \in \mathbf{c}_n^\pi(t), N_{f^{'},t-1} > \ell_T(\Delta_n^{f,l})\Big\} = 0 \tag{55}$$

$$\Rightarrow \sum_{f\in\mathbb{F}, l\in[K_f]} \Pr\Big\{\mathbf{c}_n^\pi(t) = C_{f,n}^{B,l}, N_{f,t} > N_{f,t-1}, \forall f^{'} \in \mathbf{c}_n^\pi(t), N_{f^{'},t-1} > \ell_T(\Delta_n^{f,l})\Big\}$$

$$\leq \Pr\{\neg\mathcal{P}_t \vee \mathcal{B}_t\} \leq (1-\beta) + 2FU_n^{-\Psi_{f,n}} t^{1-\Psi_{f,n}} \tag{56}$$

Thus by the definition of $N_{f,T}^{l,\text{suf}}$, we have

$$\mathbb{E}\left[\sum_{f\in\mathbb{F}, l\in[K_f]} N_{f,T}^{l,\text{suf}}\right] \leq \sum_{t=1}^{T} \left[(1-\beta) + 2FU_n^{-\Psi_{f,n}} t^{1-\Psi_{f,n}}\right]$$

$$= (1-\beta)T + 2F\frac{\zeta(\Psi_{f,n}-1)}{U_n^{\Psi_{f,n}}}. \tag{57}$$

$$\begin{aligned}
\mathrm{Reg}^{\pi}_{\Theta_n,\alpha,\beta}(T) &= T\alpha\beta R^{\text{opt}}_{\Theta_n} - \left(T\alpha R^{\text{opt}}_{\Theta_n} - \mathbb{E}\left[\sum_{f\in\mathbb{F}} \left(\Delta_n^{f,l} + \sum_{l\in K_f} \left(N_{f,T}^{l,\text{suf}} + N_{f,T}^{l,\text{und}} \right) \cdot \Delta_n^{f,l} \right) \right] \right) \\
&\leq \left(F + \mathbb{E}\left[\sum_{f\in\mathbb{F}, l\in[K_f]} N_{f,T}^{l,\text{suf}} \right] \right) \cdot \Delta_{n,\max} + \sum_{f\in\mathbb{F},\Delta^f_{n,\min}>0} \left(\ell_T(\Delta^f_{n,\min}) \cdot \Delta^f_{n,\max} \right) - (1-\beta)\cdot T\alpha R^{\text{opt}}_{\Theta_n} \\
&\leq \left(\frac{2\zeta(\Psi_{f,n}-1)}{U_n^{\Psi_{f,n}}} + 1 \right) \cdot F \cdot \Delta_{n,\max} + \sum_{f\in\mathbb{F},\Delta^f_{n,\min}>0} \frac{2\Psi_{f,n}\log U_n t}{(g^{-1}(\Delta^f_{n,\min}))^2} \cdot \Delta^f_{n,\max} \qquad (64)
\end{aligned}$$

E. Upper Bound on $N_{f,T}^{l,und}$

Consider α-sub-optimal caching strategies $\mathbf{c}_n^{\pi}(t) \in \mathbb{C}_n^B$ which are under-sampled when played (i.e., $f \in \mathbf{c}_n^{\pi}(t)$ are cached). The counter N_f for a file f increases from 1 to $\ell_T(\Delta_n^{f,K_f})$. The range of counters N_f can be broken into segments $(\ell_T(\Delta_n^{f,j-1}), \ell_T(\Delta_n^{f,j})]$ for $j \in [K_f]$. Assume that for a file f, $N_{f,t-1} \in (\ell_T(\Delta_n^{f,j-1}), \ell_T(\Delta_n^{f,j})]$ for some j. In a α-*sub-optimal* round t, for $\mathbf{c}_n^{\pi}(t) = C_{f,n}^{B,l}$ for some $l > j$, the regret suffered $\Delta_n^{f,l} < \Delta_n^{f,j}$. Thus for the counter $N_{f,t}$ in the range $(\ell_T(\Delta_n^{f,j-1}), \ell_T(\Delta_n^{f,j})]$, the total regret for the under-sampled files is at most $(\ell_T(\Delta_n^{f,j}) - \ell_T(\Delta_n^{f,j-1}))\cdot\Delta_n^{f,j})$ in the rounds the counter $N_{f,t}$ is incremented. These are used in the following. For any file $\{f \in \mathbb{F} | \Delta^f_{n,\min} > 0\}$, we have (58)-(63). The last inequality (63) follows from the definitions of $\Delta^f_{n,\min}$ and $\Delta^f_{n,\max}$ and the fact that $\ell_T(\Delta)$ is a decreasing function of Δ.

Combining (47), (57) and (63), and substituting in (10), we have (64) which completes the proof of the Lemma. ■