Coded Caching in Presence of User Inactivity

Jialing Liao and Olav Tirkkonen
Aalto University
Department of Communications and Networking (Comnet)
FI-00076 Espoo, Finland
Email: jialing.liao@ieee.org, olav.tirkkonen@aalto.fi

***Abstract*—We consider a one-server cache-enabled network under homogeneous file and network settings in presence of user inactivity, which is inherent to wireless mobile networks. Coded caching has been well studied for wired networks with static network properties. However, the lack of user inactivity information in the placement phase in the considered scenario requires redesigning coded caching against uncertainty. Unlike random or probabilistic caching studied in the literature, deterministic coded caching is considered to minimize the worst-case backhaul load by optimizing the file subpacketization and the caching strategy. First, a coded caching method is used, where each file is split into the same type of fragments labeled using sets with fixed cardinality, and the optimality of the selected cardinality is proved. Optimal file subpacketization by splitting the file into multiple types of fragments labeled with multiple cardinalities is then discussed. We show that the closed-form optimum turns out to be given by the same fixed cardinality as the one obtained without user inactivity—optimizing for user inactivity only affects file delivery, cache placement is not affected.**

I. Introduction

Coded caching can considerably reduce the backhaul load, i.e. the number of coded messages transmitted in parallel, with a considerable global caching gain [1], by making use of the local cached content and transmissions at the backhaul to exploit simultaneous coded multicasting. Most of the works on the topic are targeted for wired networks with static network and content properties. However, in practice, there are many stochastic properties that impose the need to optimize coded caching against uncertainty. Coded caching should be studied considering the impacts of randomness [2]–[4], caused by wireless channel fading, multiple antennas, transmission interference, dynamic file popularity, user inactivity, and mobility.

An important source of uncertainty in content delivery, especially in a mobile network scenario, is *user inactivity*. The main reason for inactivity is that users are free to change location between cache placement and cache delivery. Edge optimized caching is local, e.g., performed on a per base station level. Due to mobility, a user present at the cache placement phase may not be within the range of the same cache during the delivery phase. When this happens, the content will be delivered only to active devices being served by the cache. As information about user activity is not available during cache placement, the design of optimal cache placement has to be addressed. The objective of this work is to find optimal cache placement and delivery strategies in the presence of user inactivity.

In [1], Maddah-Ali and Niesen (MAN) presented pioneering information-theoretic research on coded caching, where the network topology was deterministic without user inactivity, inspiring a substantial body of scientific work. Of relevance to this paper are [4]–[8] which considered coded caching either from the perspective of optimization or with user inactivity. Multi-server networks in a random topology were considered in [6], where the users were randomly connected to a fixed number of servers. Maximum distance separable (MDS) codes were utilized to construct file pieces and thus enabled the users to recover the required file with fewer fragments from a limited number of servers out of all. This is an opposite scenario of what is studied in this paper. Here we consider one server, and the randomness is in the set of users connected to the server.

The paper [4] characterized user inactivity for a cache-enabled D2D network with K users. Each user might be inactive independently at a given probability, thus the number of effective users can be predicted. With D2D, each user can both transmit and receive content from others, and hence there is a multiplier $K-1$ in file subpacketization generally. When part users became inactive, the multiplier might drop to some $K-1-\alpha$. One can choose a proper α with any given threshold of the outage probability for successful transmission which is a function of α. The D2D scenario in [4] is more similar to the case of [6] with fewer transmitters available than to our case. Instead of performance analysis, we pursue optimization for the best subpacketization and caching strategy.

Papers [7] and [8] provided insights on optimization based coded caching design for nonuniform file parameters. User inactivity was not considered. In multi-round delivery when multiple users share a cache [9]–[11], in some rounds not all users are present. Thus, from a cache delivery perspective, the results of [9]–[11] directly apply to an inactive user scenario. However, the *cache placement* setting is different. The user caching profile is assumed known during cache placement in [9], decentralized caching is applied in [10], while MAN cache placement with pre-selected cardinality is assumed upfront in [11]. In contrast, here we consider deterministic caching in a situation where the set of active users is not known during cache placement, and optimize cache placement.

In this paper, we focus on a scenario where there are several cache-enabled users connected to a single server via shared links, and where there is inactivity. First, a method with file fragments of one size is considered, and optimal fragment size is found. Second, a general scheme is considered where each

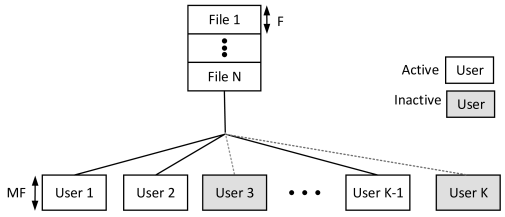

Fig. 1: A cache-aided network in presence of user inactivity.

file is divided into fragments of different sizes, and fragment sizes are optimized over. It is proved that the optimal cache placement is the same for the basic case without user inactivity and the case with user inactivity. Theoretical and simulation results are presented to illustrate the advantage of the caching scheme against user inactivity, and the equivalence among the subpacketization optimization in all cases.

Notation: $[a:b]$ denotes $\{a, a+1, \ldots, b-1, b\}$ consisting of integers from a to b. Specially, $[b]$ denotes the set $\{1, 2, \ldots, b-1, b\}$. The operator $\oplus$ denotes the bitwise "XOR" operation.

II. System Model

There is a base station connected to the core network with access to all the file library (N files $W_1, W_2, \ldots, W_N$ each with equal size F and popularity), and K users each with local storage of size MF. The users are connected to the server via error-free shared links. The probability for each user to be inactive is p. In the placement phase, the user inactivity is unknown while the base station has the information of the user inactivity in the delivery phase. Assuming that in a realization, there are I inactive users forming an inactive user set $\mathcal{I}$. To ensure the significance of the discussion, we assume at least one user is becoming inactive and at the same time, there is at least one active user to be served, i.e. $I \in [K-1]$. The number of active users is correspondingly defined as $J = K - I$. The probability for I of the K users being inactive is

$$P(I) = \binom{K}{I} p^I (1-p)^{K-I}, \ I \in [K-1]. \tag{1}$$

The cached content at the local cache of a user k is defined as Z_k. The content delivered through the backhaul via coded multicast is described as $X_{\boldsymbol{d}}$, where $\boldsymbol{d} = (d_1, d_2, \ldots, d_K)$ with $d_k \in [N], k \in [K]$ denoting the demand of user k.

As a common metric for measuring the performance of coded caching methods, the backhaul load is defined as the volume of content needed to be delivered via backhaul using coded multicasting. The backhaul load can be calculated both in the worst case when the active users each requesting a different file, and the average case with all types of possible demands considered. Here, the worst-case backhaul load is considered which implies that the number of files is higher than the number of users $N > K$. We aim to minimize the worst-case backhaul load by designing the caching strategy subject to file size and cache size constraints.

III. Coded Caching in Presence of Inactive Users

A. Content Placement and Delivery with User Inactivity

We begin with the effective file subpacketization in Maddah-Ali-Niesen's (MAN) method to make full usage of the multicast delivery opportunities. In the MAN method, all the users have the same cache content placement and all the files are equally cached in local storage because of the homogeneous settings. Define a variable $t \triangleq \frac{KM}{N}$, and then divide each file into $\binom{K}{t}$ fragments equally. t is referred to as the cache replication parameter in literature [12]. It is assumed t is an integer. If not, content sharing can be used to deal with this issue. The fragments are indexed by all subsets of users $\tau \subset [K]$ of fixed cardinality $|\tau| = t$. Accordingly the fragments of file n are $W_{n,\tau}$. For sake of simplicity, the set of all t-element subsets of $[K]$ is defined as $\zeta = \{\tau | \tau \subset [K], |\tau| = t\}$. It is assumed that user k stores the fragments $W_{n,\tau}$ of each file n when $k \in \tau, \tau \in \zeta$. Hence, the cache content placement at user k can be written as

$$Z_k = (W_{n,\tau} : \tau \in \zeta, k \in \tau, n \in [N]). \tag{2}$$

In each cache, there are $\binom{K-1}{t-1}$ fragments for each file, and each fragment has normalized size of $1/\binom{K}{t}$. Thus the cache capacity constraint holds as follows

$$N \binom{K-1}{t-1} \frac{1}{\binom{K}{t}} = M. \tag{3}$$

Without user inactivity ($I = 0$), the server can deliver several packets each of which comprises coded fragments to the users to help them reconstruct their requested files:

$$X_{\boldsymbol{d}} = (X_{\boldsymbol{d},\mathcal{S}} : \mathcal{S} \in \vartheta), \tag{4}$$

$$X_{\boldsymbol{d},\mathcal{S}} = \oplus_{k \in \mathcal{S}} \, W_{d_k, \mathcal{S}\setminus\{k\}}, \tag{5}$$

where the set $\mathcal{S}$ has one more element than set τ satisfying $\mathcal{S} \subset [K], |\mathcal{S}| = t+1$. The set of all $\mathcal{S}$ is ϑ. This coded multicast strategy works in the way that all the users are able to recover their request files using the same transmitted packets and the cached fragments in local cache. For any user i in a particular $\mathcal{S}$, the linear combination $X_{\boldsymbol{d},\mathcal{S}}$ can be rewritten as

$$X_{\boldsymbol{d},\mathcal{S}} = W_{d_i, \mathcal{S}\setminus\{i\}} \oplus (\oplus_{k \in \mathcal{S}, k \neq i} \, W_{d_k, \mathcal{S}\setminus\{k\}}), \tag{6}$$

where $W_{d_i,\mathcal{S}\setminus\{i\}}$ is one of the fragments that user i needs to recover the requested file d_i. The other fragments in linear combination $(\oplus_{k \in \mathcal{S}, k \neq i} \, W_{d_k, \mathcal{S}\setminus\{k\}})$ are all cached at user i according to the cache placement in (2) due to the fact that $i \in \mathcal{S}\setminus\{k\}$ for any $k \in \mathcal{S}$ but $k \neq i$. Therefore, $W_{d_i,\mathcal{S}\setminus\{i\}}$ can be decoded by user i. Taking all types of $\mathcal{S} \in \vartheta, i \in \mathcal{S}$ into account, user i is thus able to decode all the missing fragments of d_i, i.e. $(W_{d_i,\mathcal{S}\setminus\{i\}} : \mathcal{S} \in \vartheta, i \in \mathcal{S}) = (W_{d_i,\tau} : \tau \in \zeta, i \notin \tau)$.

As the linear combination of several fragments has the same size as a single fragment, i.e. $F/\binom{K}{t}$, the backhaul load can be written as the size of a fragment multiplied by the number of the different sets $\mathcal{S}$ as follows:

$$R = \frac{F}{\binom{K}{t}} \binom{K}{t+1} = F \, \frac{K-t}{t+1}. \tag{7}$$

Because file size F performs as a multiplier in backhaul loads, unit file size is assumed in the following for briefness.

In [9], multi-round delivery to users sharing caches is considered. In each delivery round, a subset of cache profiles may be present. Thus, for a given round, the cache delivery problem addressed in [9] is the same as the delivery problem in a situation with a set of inactive users. We shall thus use the delivery policy of [9], rephrased to an inactive user scenario.

Assuming there are I inactive users forming an inactive user set $\mathcal{I} \subset [K]$, we utilize a general cardinality $l \in [K]$ in file subpacketization instead of the cache replication parameter $t = KM/N$ that is used in the MAN method. The optimal value of l will be optimized in Subsection III-B. The transmitted packets would be

$$X_{\boldsymbol{d}} = \begin{cases} (X_{\boldsymbol{d},\mathcal{S}} : \mathcal{S} \in \vartheta), & \text{if } l+1 > I, \\ (X_{\boldsymbol{d},\mathcal{S}} : \mathcal{S} \in \vartheta, \mathcal{S} \not\subset \mathcal{I}), & \text{if } l+1 \leq I, \end{cases} \tag{8}$$

where the packet given any subset $\mathcal{S}$ is

$$X_{\boldsymbol{d},\mathcal{S}} = \oplus_{k \in \mathcal{S}, k \notin \mathcal{I}} W_{d_k,\mathcal{S}\setminus k}. \tag{9}$$

The worst case backhaul load then becomes [9]

$$R(l) = \begin{cases} \frac{1}{\binom{K}{l}} \binom{K}{l+1}, & \text{if } l+1 > I, \\ \frac{1}{\binom{K}{l}} \left[\binom{K}{l+1} - \binom{I}{l+1} \right], & \text{if } l+1 \leq I. \end{cases} \tag{10}$$

Comparing (7) and (10), it is obvious that the worst-case backhaul load in presence of user inactivity is either the same as the one derived without user inactivity when $t+1 > I$ or is smaller than the one without user inactivity when $t+1 \leq I$. For clarification, we summarize the procedure of the centralized coded caching scheme in presence of user inactivity in Alg. 1, based on (10). Note that in Alg. 1, l^* denotes the optimal cardinality based on file subpacketization optimization given by $l^* = KM/N$, which shall be carefully proved in Subsection III-B and Section IV.

While in (10) and Alg. 1 a framework of coded caching with user inactivity is presented, there is an essential problem remaining: *What is the optimal cache placement policy if it is known at the cache placement that a random set of I out of K users will be inactive at the time of cache delivery?*

B. Subpacketization Optimization with Fixed Cardinality

The subpacketization method and cache content placement used here is based on a group of subsets $\tau \subset [K]$ with $|\tau| = l$ to create possible multicast opportunities in the content delivery phase. We strive to find the optimal cardinality of τ.

First, we consider the normal case without user inactivity, and define an multiple cardinalities $l = |\tau|, l \in [K]$, to replace the fixed $t = KM/N$. The optimal l should give the lowest backhaul load $R(l) = \binom{K}{l+1}/\binom{K}{l} = \frac{K-l}{l+1}$ while satisfying cache capacity constraint $\binom{K-1}{l-1}/\binom{K}{l} = l/K \leq M/N$. The optimization problem can be rewritten into

$$\min_{l} \quad \frac{K+1}{l+1} - 1 \tag{11a}$$

$$\text{s.t.} \quad l \leq \frac{KM}{N}, l \in [K]. \tag{11b}$$

Algorithm 1 Coded Caching in Presence of User Inactivity

procedure PLACEMENT
  $l \leftarrow l^*$ (the optimal solution in file subpacketization optimization: $l^* = KM/N$)
  $\zeta \leftarrow \{\tau | \tau \subset [K], |\tau| = l\}$
  for $n \in [N]$ do
    split W_n into $(W_{n,\tau} | \tau \in \zeta)$ with identical size
  end for
  for $k \in [K]$ do
    user k caches $Z_k \leftarrow (W_{n,\tau} | \tau \in \zeta, k \in \tau, n \in [N])$
  end for
end procedure
Users make requests $\boldsymbol{d}$ given the number and identity of the inactive users $(I, \mathcal{I})$
procedure DELIVERY
  $l \leftarrow KM/N$
  $\vartheta \leftarrow \{\mathcal{S} | \mathcal{S} \subset [K], |\mathcal{S}| = l+1\}$
  if $l+1 > I$ do
    $X_{\boldsymbol{d}} \leftarrow (\oplus_{k \in \mathcal{S}, k \notin \mathcal{I}} W_{d_k,\mathcal{S}\setminus\{k\}} : \mathcal{S} \in \vartheta)$
  else if $l+1 \leq I$ do
    $X_{\boldsymbol{d}} \leftarrow (\oplus_{k \in \mathcal{S}, k \notin \mathcal{I}} W_{d_k,\mathcal{S}\setminus\{k\}} : \mathcal{S} \in \vartheta, \mathcal{S} \not\subset \mathcal{I})$
  end if
end procedure

Since the objective (11a) decreases with respect to l, the optimal solution is the largest l with cache capacity constraint satisfied with equality, i.e. $l = \frac{KM}{N}$, which agrees with the cache replication parameter $t = (KM)/N$ used in the MAN method. In particular, when $(KM)/N$ is not an integer, the optimal cardinality becomes $l = \lfloor KM/N \rfloor$, which works for the scenario with user inactivity as well.

Similarly, we substitute the worst case backhaul load with user inactivity (10) into the objective function and again replace the fixed $t = KM/N$ with a variable l to be optimized. The coded caching optimization with user inactivity after simplification can be written as

$$\min_{l} \quad R(l) \tag{12a}$$

$$\text{s.t.} \quad l \leq \frac{KM}{N}, \quad l \in [K-1]. \tag{12b}$$

where the objective function is given by (10).

To find the optimal l, the analysis of (12) can be divided into two parts. We have

Lemma 1: The backhaul load $R(l)$ of (10) is a decreasing function of l in the interval $l \in [I-1]$.

Proof: See Appendix A. ∎

We can now show

Theorem 1: If MAN cache placement with cardinality l of all caching subsets τ is used in the presence of user inactivity, minimum worst-case backhaul load is achieved with cardinality $l = \frac{KM}{N}$.

Proof: We first treat separately the minimization in the regions $l > I-1$ and $l \leq I-1$ separately. In the region $l > I-1$, the backhaul load (10) is the same as the one without

user inactivity (7), for which the optimal cardinality has been proved to be $l = MK/N$. Thus if $I \leq KM/N$, this yields the minimum backhaul in this region, while if $I > KM/N$, all of this region is infeasible.

According to Lemma 1, the minimum backhaul in the second region $l \leq I-1$ is achieved at the maximal feasible point $l = \min(KM/N, I-1)$.

It remains to find the smaller value of the solutions in the two regions, in the case $KM/N \geq I$. For this, we compute

$$R(I-1) - R\left(\frac{KM}{N}\right) = \frac{(I-1)!}{I(KM/N+1)K!} \times B\,\Gamma, \quad (13)$$

where $B = I(KM/N+1)(K-I+1)!$ and

$$\begin{aligned} \Gamma + 1 &= \frac{(K+1)(KM/N+1-I)}{I(KM/N+1)(K-I+1)!\binom{K}{I-1}} \\ &\geq \frac{K+1}{KM/N+1} > 1. \end{aligned} \quad (14)$$

This completes the proof. ∎

Theorem 1 thus states that the optimal cardinality in file subpacketization for coded caching in presence of user inactivity in the whole interval $I \in [K-1]$ is always $l = KM/N$, which is the same as t used in the MAN method without user inactivity.

IV. Subpacketization with Multiple Cardinalities

The analysis in previous section contains redundancy in the content placement caused by caching the fragments related to the inactive users. For instance, the fragments $(W_{n,\tau} : k \in \tau, \tau \cap \mathcal{I} \neq \emptyset, n \in [N])$ stored in an active user k seem to take up storage space without contributing in reducing backhaul load. The optimal file subpacketization and cache placement is to cache only the fragments corresponding to the active users, e.g. $(W_{n,\tau} : k \in \tau, \tau \cap \mathcal{I} = \emptyset, n \in [N])$. However, the information about user inactivity is unknown in the placement phase, which means that the set $\mathcal{I}$ can not be specified.

Given an inactivity probability, the probability of a user caching a file fragment in vain grows with fragment label cardinality $|\tau|$. Above, we found that the optimal cardinality is given by $l = KM/N$ if all labels have the same cardinality. As the number of active users decreases, there is a possibility that having labels of multiple cardinalities might lead to more efficient use of the caches. In [8], the multiple cardinalities based file subpacketization is utilized to deal with the heterogeneous of file popularity which imposes multilevel file subpacketization in terms of popularity.

For this, we split each file based on subsets with a series of different cardinalities $l \in [0:K]$ instead of a fixed number t. That is to say, each file is split into 2^K fragments labeled with $W_{n,\mathcal{A}^l} : \mathcal{A}^l \subset [K], |\mathcal{A}^l| = l, l \in [0:K]$. Similarly, we assume that in the cache placement phase, a fragment is cached by user k if its fragment label $\mathcal{A}^l, l \in [0:K]$ includes k:

$$\begin{aligned} Z_k = (W_{n,\mathcal{A}^l} : k \in \mathcal{A}^l, \mathcal{A}^l \subset [K], |\mathcal{A}^l| = l, \\ l \in [0:K], n \in [N]). \end{aligned} \quad (15)$$

According to the cardinality l, the fragments for each file n can be divided into $K+1$ groups as $W_n^l = (W_{n,\mathcal{A}^l} : \mathcal{A}^l \subset [K], |\mathcal{A}^l| = l), l = 0, 1, \ldots, K$. There are $\binom{K}{l}$ types of fragments in fragment group W_n^l. In total, there are $\sum_l \binom{K}{l} = 2^K$ different fragments for each file. By adjusting the weights of the fragment groups $W_n^l, l \in [0:K]$ for each file, the space that each fragment group takes from the cache is decided accordingly. The number of effective users involved in the caching design can then be controlled to some degree.

It is assumed that the fragments in the same group l have the same size. Define a weight vector as $\boldsymbol{\alpha} \triangleq [\alpha^0, \alpha^1, \ldots, \alpha^K]$ with α^l denoting the size of a fragment in fragment group l normalized by file size F, i.e. $\alpha^l = |W_{n,\mathcal{A}^l}|, n \in [N]$. Hence, the size of fragment group l of file n, W_n^l, is $\binom{K}{l}\alpha^l F$.

Now the file size and cache capacity constraints are:

$$\sum_{l=0}^{K} \binom{K}{l} \alpha^l = 1, \quad \sum_{l=1}^{K} \alpha^l \binom{K-1}{l-1} \leq \frac{M}{N}, \quad (16a)$$

$$\alpha^l \geq 0, \ l = 0, \ldots, K. \quad (16b)$$

The content to be delivered to the users via backhaul then becomes:

$$X_{\boldsymbol{d}} = \begin{cases} (X_{\boldsymbol{d},\mathcal{A}^{l+1}} : \mathcal{A}^{l+1} \subset [K], |\mathcal{A}^{l+1}| = l+1), & \text{if } l+1 > I, \\ (X_{\boldsymbol{d},\mathcal{A}^{l+1}} : \mathcal{A}^{l+1} \not\subset \mathcal{I}, \mathcal{A}^{l+1} \subset [K], |\mathcal{A}^{l+1}| = l+1), & \\ & \text{if } 1 < l+1 \leq I, \\ X_{\boldsymbol{d},\mathcal{A}^l}, & \text{if } l = 0, \end{cases}$$

where the packet corresponding to a given fragment label set $\mathcal{A}^{l+1}$ is given by

$$X_{\boldsymbol{d},\mathcal{A}^{l+1}} = \begin{cases} \oplus_{k \in \mathcal{A}^{l+1}, k \notin \mathcal{I}} \ W_{d_k,\mathcal{A}^{l+1}\setminus\{k\}}, & \text{if } l+1 > I, \\ \oplus_{k \in \mathcal{A}^{l+1}, k \notin \mathcal{I}} \ W_{d_k,\mathcal{A}^{l+1}\setminus\{k\}}, & \text{if } 1 < l+1 \leq I, \\ W_{d_k,\mathcal{A}^l}, & \text{if } l = 0. \end{cases}$$

In particular, there is an exceptional case for $l = 0$ when $\mathcal{A}^0$ equals to $\emptyset$ and thus none of the users has stored the subfiles $W_{n,\mathcal{A}^0}, n \in [N]$. Accordingly, the backhaul load normalized by file size F is written as

$$R(\boldsymbol{\alpha}) = \sum_{l=0}^{K-1} \binom{K}{l+1} \alpha^l - \sum_{l=0}^{I-1} \binom{I}{l+1} \alpha^l. \quad (17)$$

The caching design turns to solving a linear programming of minimizing the backhaul load subject to the file size and cache capacity constraints (16):

$$\min_{\boldsymbol{\alpha}} \quad R(\boldsymbol{\alpha}) \quad \text{s.t.} \ \ (16a) - (16b). \quad (18)$$

Existing solvers, e.g. CVX [13], can be used to solve problem (18) with $K+1$ variables and $K+3$ constraints [14]. It is important to figure out the structure of the optimal solution which is discussed below.

To simplify the problem, we replace the original variables with the variables satisfying $\beta^l = \binom{K}{l}\alpha^l, l \in [0:K]$. Thus we derive a new weight vector as $\boldsymbol{\beta} \triangleq [\beta^0, \beta^1, \ldots, \beta^K]$. In

this case, problem (18) can then be rewritten into

$$\min_{\{\beta^l\}} \quad \sum_{l=0}^{K-1} \frac{K-l}{l+1}\beta^l - \sum_{l=0}^{I-1} \binom{I}{l+1} / \binom{K}{l} \beta^l \tag{19a}$$

$$\text{s.t.} \quad \sum_{l=1}^{K} l\beta^l \leq t, \tag{19b}$$

$$\sum_{l=0}^{K} \beta^l = 1, \quad \beta^l \geq 0, \ l \in [0:K]. \tag{19c}$$

It can be observed that the terms related to the inactive users in the objective destroy the similarity among the combination terms in the objective, and thus it becomes challenging to find a closed-form solution to problem (19).

To proceed, we analyze the properties of the objective function and the linear constraints. The discussion is generally accomplished in *four* steps each of which is formulated as a Lemma given below, discussing *the objective, the constraints, the structure of the optimal solution, and an exceptional case.* Due to space limitation, the proofs of the lemmas are omitted here, with details available in the longer version from Arxiv: 2109.14680.

Lemma 2: The first derivative of coefficients in the objective function (19a) is negative while the second derivative is positive when $I \in [K-2]$ and equal to 0 when $I = K-1$.

Lemma 3: The optimal solution to problem (19) must have a tight cache capacity constraint (19b).

Lemma 4: The optimal solution has at most one non-zero variables of $\boldsymbol{\beta}$ (two for non-integer t).

Lemma 5: There is an exceptional case $I = K-1$ when the second derivative equals 0. It will destroy the proof of Lemma 4 which strictly requires a positive second derivative. The optimal solution, in this case, is no longer unique, but the solution in the general case still works.

Theorem 2: The optimal solution to problem (19) is the same as the optimal solution to the file subpacketization optimization with fixed cardinality, which is given by

$$\beta^l = \begin{cases} 1, & \text{if } \ l = t, \\ 0, & \text{else}, \end{cases} \tag{20}$$

assuming $t = KM/N$ is integer. When $t = KM/N$ is non-integer, there are two adjacent nonzero elements in $\{\beta^l\}$ around t. Letting $\eta = \lceil t \rceil - t$, the solution becomes

$$\beta^l = \begin{cases} \eta, & \text{if } \ l = \lfloor t \rfloor, \\ 1-\eta, & \text{if } \ l = \lceil t \rceil, \\ 0, & \text{else}. \end{cases} \tag{21}$$

Proof: This follows directly from following the four steps given in Lemma 2-Lemma 5. ∎

Corollary 1: Based on Theorem 1 and Theorem 2, Alg. 1 is optimal for integer KM/N.

Note that the optimal solutions to problems (12) and (18) are proven to be independent on I. The considered schemes can thus be used in a user inactivity case knowing neither $\mathcal{I}$ nor the number I of inactive users and content placement phase.

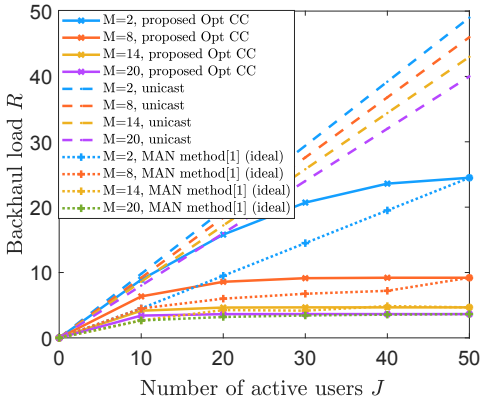

Fig. 2: Backhaul load of coded caching with user inactivity.

V. Simulations

A. Coded Caching with Fixed Cardinality

A one-server cache-enable network is considered. There are $K = 50$ users and $N = 100$ files with equal popularity and size. Fig. 2 presents the impacts of cache size M and the number of active users J on worst case backhaul load. The performance of the proposed optimization-based coded caching with fixed cardinality is compared to the unicast caching method, and the ideal MAN method [1] where perfect user inactivity information is assumed in the placement phase.

As can be seen Fig. 2, the proposed scheme outperforms the unicast caching scheme while providing a compatible performance to the idea MAN method against user inactivity. In general, the backhaul load decreases with the increase of cache size M and rises with the increase of the number of active users J. Increasing M, the backhaul load decreases dramatically. The decrease of backhaul load is more obvious with more active users, i.e. larger J. The gap between them decreases when the cache size M rises. When $M = 20$, the proposed method is approximately the same as the MAN method in the ideal case. Increasing J causes some increase in backhaul load, but the gap is limited for the proposed method, e.g. less than 10. The gap is getting narrower when increasing cache size M. That is to say, the proposed method can provide an acceptable solution to user inactivity. Particularly when $J = K = 50$, the proposed method is the same as the MAN method as all the users are active. Moreover, the backhaul load tends to be stable when J tends to K, which agrees with (10) since the backhaul load is independent of the number of inactive users I when $I < t+1$, i.e. $J > K-t+1$. For instance, when $M = 8$, the solid curve in orange becomes stable after reaching $J = 43$.

B. Optimal Coded Caching with Multiple Cardinalities

To investigate the optimal coded caching scheme against user inactivity, the performance of coded caching using fixed cardinality and multiple cardinalities in file subpacketization is compared. Results obtained using optimization solver CVX are compared to the closed-form optimal solution (20)-(21).

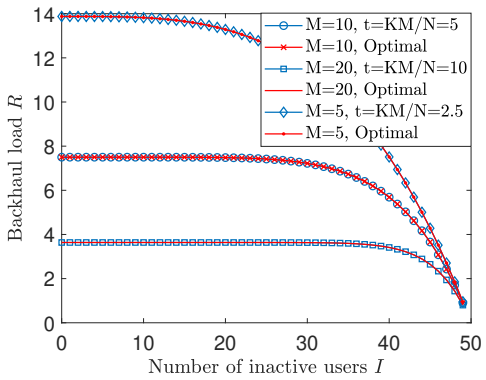

Fig. 3: Performance comparison between coded caching with fixed l and optimal scheme with multiple cardinalities against user inactivity.

In Fig. 3 we consider a cache enabled network where we let $K = 50, N = 100, M = 5/10/20$, and the number of inactive users varies within $[0 : K)$, to compare the performance of the optimal solution for coded caching with multiple $l \in [0 : K]$ and the one with fixed $l = t$. The simulation confirms the closed-form solution. The backhaul load for the proposed caching scheme with fixed cardinality is always the same as the optimal solution with multiple l for all the different values of inactive users and cache size.

VI. Conclusions

In this paper, we studied the coded caching strategy for one-server cache-enabled networks with inactive users. Based on the classic file subpacketization and coded multicast strategy, we considered a coded caching method with fixed cardinality of the fragment label set, and proved the optimality cardinality $t = KM/N$. This is known as the cache replication parameter used in the MAN method without user inactivity. The scheme was extended to a scenario where multiple cardinalities can be used instead of a fixed one. The weights for different types of fragments labeled with different cardinalities were optimized to minimize the worst-case backhaul load. The optimal solution turns out to be the same as the one derived from the caching scheme with fixed cardinality. Numerical results show that the considered coded caching scheme approximates the ideal MAN method.

Acknowledgment

This work was funded in part by the Academy of Finland (grant 319058).

Appendix A

To proceed, we compute the ratio $R(l+1)/R(l)$, with $1 \leq l \leq I-2$. By expanding the binomial coefficients we get

$$\begin{aligned}\frac{R(l+1)}{R(l)} &= \frac{\left[\binom{K}{l+2} - \binom{I}{l+2}\right]}{\left[\binom{K}{l+1} - \binom{I}{l+1}\right]} \frac{\binom{K}{l}}{\binom{K}{l+1}} \\ &= \frac{A(l)}{A(l) + B(l) - C(l)}\end{aligned}$$

where

$$\begin{aligned}A &= (l+1)\left(\frac{K!}{(K-l-2)!} - \frac{I!}{(I-l-2)!}\right) \\ B &= \frac{(K+1)!}{(K-l-1)!} \\ C &= (K+1+(K-I)(l+1))\frac{I!}{(I-l-1)!}.\end{aligned}$$

Because $K \geq I$, we have $A \geq 0$, as well as $B > 0$ and $C > 0$ in the domain of interest. It is thus sufficient to prove that $B > C$ for all l in the domain. For this, we first use the fact that $(K-m)/(I-m) \geq K/I$ for non-negative m and $K \geq I$ to lower bound the $l+1$ smallest terms in the $l+2$-fold product in B to get

$$B \geq (K+1)\left(\frac{K}{I}\right)^{l+1} \frac{I!}{(I-l-1)!} \tag{22}$$

Writing $K/I = 1 + (K-I)/I$ we can expand $(K/I)^{l+1}$ using the binomial expansion. When $K > I$, all terms in the expansion are positive. Comparing the two first terms in this expansion to C directly shows that $B > C$ holds. This completes the proof. ■

References

[1] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 6, no. 5, pp. 2856–2867, May 2014.
[2] M. Bayat, K. Wan, M. Ji, and G. Caire, "Cache-Aided modulation for heterogeneous coded caching over a Gaussian broadcast channel," arXiv preprint arXiv:2001.05784.
[3] A. Tölli, S. P. Shariatpanahi, J. Kaleva, and B. H. Khalaj, "Multi-Antenna interference management for coded caching," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2091–2106, 2020.
[4] C. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of D2D coded caching with uncoded cache placement and one-shot delivery," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8179–8192, 2019.
[5] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 6, no. 99, pp. 281–288, 2014.
[6] N. Mital, D. Gündüz, and C. Ling, "Coded caching in a multi-server system with random topology," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4620–4631, 2020.
[7] Y. Deng and M. Dong, "Optimal uncoded placement and file grouping structure for improved coded caching under nonuniform popularity," in *Proc Int. Sym. Modeling and Opt. in Mobile, Ad Hoc, and Wireless Net. (WiOPT)*, 2020, pp. 1–8.
[8] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1893–1919, Mar. 2020.
[9] E. Parrinello, A. unsal, and P. Elia, "Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2252–2268, Apr. 2020.
[10] S. Jin, Y. Cui, H. Liu, and G. Caire, "Order-Optimal decentralized coded caching schemes with good performance in finite file size regimeorming for MIMO-OFDM with partial feedback," in *Proc. IEEE GLOBECOM*, Washington, DC, USA, Dec. 2016, pp. 1–7.
[11] M. Bayat, K. Wan, and G. Caire, "Coded caching over multicast routing networks," arXiv preprint arXiv:2008.08900.
[12] S. Wang, W. Li, X. Tian, and H. Liu, "Fundamental limits of heterogenous cache," arXiv preprint arXiv:1504.01123v1.
[13] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep. 2013.
[14] M. Tao, D. Gündüz, F. Xu, and J. P. Roig, "Content caching and delivery in wireless radio access networks," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4724–4749, Jul. 2019.