Centralized Coded Caching for Heterogeneous Lossy Requests

Qianqian Yang and Deniz Gündüz
Dept. of Electrical and Electronic Eng., Imperial College London, UK
Email: {q.yang14, d.gunduz}@imperial.ac.uk

***Abstract*—*Centralized coded caching* of popular contents is studied for users with heterogeneous distortion requirements, corresponding to diverse processing and display capabilities of mobile devices. Users' distortion requirements are assumed to be fixed and known, while their particular demands are revealed only after the *placement phase*. Modeling each file in the database as an independent and identically distributed Gaussian vector, the minimum *delivery rate* that can satisfy any demand combination within the corresponding distortion target is studied. The optimal delivery rate is characterized for the special case of two users and two files for any pair of distortion requirements. For the general setting with multiple users and files, a layered caching and delivery scheme, which exploits the successive refinability of Gaussian sources, is proposed. This scheme caches each content in multiple layers, and it is optimized by solving two subproblems: lossless caching of each layer with heterogeneous cache capacities, and allocation of available caches among layers. The delivery rate minimization problem for each layer is solved numerically, while two schemes, called the *proportional cache allocation (PCA)* and *ordered cache allocation (OCA)*, are proposed for cache allocation. These schemes are compared with each other and the cut-set bound through numerical simulations.**

I. Introduction

Wireless data traffic is predicted to continue its exponential growth in the coming years, mainly driven by the proliferation of mobile devices with increased processing and display capabilities, and the explosion of available online contents. Current wireless architecture is widely acknowledged not to be sufficient to sustain this dramatic growth. A promising approach to alleviate the looming network congestion is to *proactively* place popular contents, fully or partially, at the network edge during off-peak traffic periods (see, for example, [1]–[3], and references therein).

Conventional caching schemes utilize orthogonal unicast transmissions, and benefit mainly from local duplication. On the other hand, by *coded caching*, a novel caching mechanism introduced in [3], further gains can be obtained by creating multicasting opportunities even across different requests. This is achieved by jointly optimizing the *placement* and *delivery* phases. Coded caching has recently been investigated under various settings, e.g., decentralized coded caching [5], online coded caching [6], distributed caching [7], etc.

Most of the existing literature follow the model in [3], in the sense that each file is assumed to have a fixed size, and users are interested in the whole file. However, in many practical applications, particularly involving multimedia contents, files can be downloaded at various quality levels depending on the channel and traffic conditions, or device capabilities. This calls for the design of *lossy* caching and delivery mechanisms.

We model the scenario in which each user has a preset distortion requirement known to the server. For example, a laptop may require high quality descriptions of requested files, whereas a mobile phone is satisfied with much lower resolution. Users may request any of the popular files, and the server is expected to satisfy all request combinations at their desired quality levels. We model the files in the server as independent sequences of Gaussian distributed random variables. Exploiting the successive refinability [11] of Gaussian sources, we derive the optimal caching scheme for the two-user, two-file scenario. For the general case, we propose an efficient coded caching scheme which considers multiple layers for each file, and first allocates the available cache capacity among these layers, and then solves the lossless caching problem with asymmetric cache capacities for each layer. We propose two algorithms for cache capacity allocation, namely *proportional cache allocation (PCA)* and *ordered cache allocation (OCA)*, and numerically compare the performance of the proposed layered caching scheme with the cut-set lower bound.

The most related work to this paper is [8], in which Hassanzadeh et al. solve the inverse of the problem studied here, and aim at minimizing the average distortion across users under constraints on the delivery rate as well as the cache capacities. In [9], authors also consider lossy caching taking into account the correlation among the available contents, based on which the tradeoff between the compression rate, reconstruction distortion and cache capacity is characterized for single, and some special two-user scenarios.

The rest of the paper is organized as follows. We present the system model in Section II. Section III presents results on the case with two files and two users. General case is investigated in Section IV, including a lower bound on the delivery rate. Numerical simulations are presented in Section V. Finally, we conclude the paper in Section VI.

II. System Model

We consider a server that is connected to K users through a shared, error-free link. The server has a database of N independent files, S_1, ..., S_N, where file S_i consists of n independent and identically distributed (i.i.d) samples $S_{i,1}$, ..., $S_{i,n}$ from a Gaussian distribution with zero-mean and variance σ^2, i.e., $S_i \sim \mathcal{N}(0, \sigma^2)$, for $i = 1, ..., N$.

The system operates in two phases. In the *placement phase*, users' caches are filled with the knowledge of the number of users and each user's quality requirement; but without the particular user demands. Each user has a cache of size $M_k n$ bits, whose content at the end of the *placement phase* is denoted by Z_k, $k = 1, ..., K$. Users' requests, $\mathbf{d} \triangleq (d_1, ..., d_K)$, $d_k \in \{1, ..., N\}$, are revealed after the *placement phase*. In the *delivery phase*, the server transmits a single message $X^n_{(d_1,...,d_K)}$ of size nR bits over the shared link according to all the users' requests and the cache contents. Using Z_k and $X^n_{(d_1,...,d_K)}$, each user k aims at reconstructing the file it requests within a certain distortion target D_k.

An $(n, M_1, ..., M_K, R)$ *lossy caching code* consists of K cache placement functions:

$$f_k^n : \underbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}_{N \ files} \to \{1, ..., 2^{nM_k}\} \quad \text{for} \quad k = 1, ..., K,$$

where $Z_k^n = f_k^n(S_1^n, ..., S_N^n)$; one delivery function:

$$g^n : \underbrace{\mathbb{R}^n \times ... \times \mathbb{R}^n}_{N \ files} \times \underbrace{d_1 \times ... \times d_K}_{K \ requests} \to \{1, ..., 2^{nR}\},$$

where $X^n_{(d_1,...,d_K)} = g^n(S_1^n, ..., S_N^n, d_1, ..., d_K)$; and K decoding functions:

$$h_k^n : \{1, ..., N\}^K \times \{1, ..., 2^{nM_k}\} \times \{1, ..., 2^{nR}\} \to R^n,$$

where $\hat{S}_k^n = h_k^n(\mathbf{d}, Z_k^n, X^n)$. Note that each user knows the requests of all other users in the delivery phase.

We consider quadratic (squared-error) distortion, and assume that each user has a fixed distortion requirement D_k, $k = 1, ..., K$. Without loss of generality, let $D_1 \geq D_2 \geq \cdots \geq D_K$. Accordingly, we say that a distortion tuple $\mathbf{D} \triangleq (D_1, ..., D_K)$ is *achievable* if there exists a sequence of caching codes $(n, M_1, ..., M_K, R)$, such that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} (S_{d_k,j} - \hat{S}_{k,j})^2 \leq D_k, \qquad k = 1, 2, ..., K,$$

holds for all possible request combinations $\mathbf{d}$. We reemphasize that $\mathbf{d}$ is not known during the *placement phase*, while $\mathbf{D}$ is known. For a given distortion tuple $\mathbf{D}$, we define the *cache capacity-delivery rate tradeoff* as follows:

$$R^\star(M_1, ..., M_K) \triangleq \inf\{R : \mathbf{D} \text{ is achievable.}\} \qquad (1)$$

Note that this problem is closely related to the classical rate-distortion problem. Let $R(D)$ denote the *rate-distortion function* of a Gaussian source $S \sim \mathcal{N}(0, \sigma^2)$. We have $R(D) \triangleq \frac{1}{2} \log_2 \frac{\sigma^2}{D}$ [10].

In the sequel we heavily exploit the *successive refinability* of a Gaussian source under squared-error distortion measure [11]. Successive refinement refers to compressing a sequence of source samples in multiple stages, such that the quality of reconstruction improves, i.e., distortion reduces, at every stage. A given source is said to be successively refinable under a given distortion measure if the single resolution distortion-rate function can be achieved at every stage. Successive refinement has been extensively studied in the source coding literature; please see [8] for its use in the caching context.

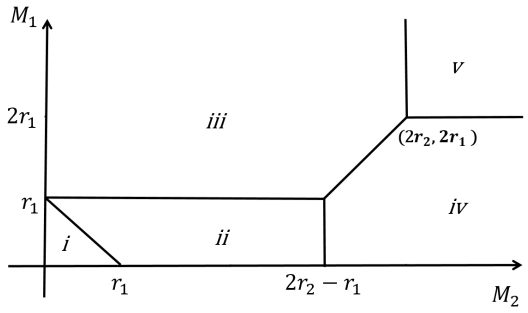

Fig. 1. Illustration of the five distinct cases of the cache capacities, M_1 and M_2, depending on the distortion requirements of the users, r_1 and r_2.

III. Optimal Lossy Caching: Two Users and Two Files ($N = K = 2$)

In this section, we characterize the optimal cache capacity-delivery rate tradeoff for the lossy caching problem with two users ($K = 2$) and two files ($N = 2$). The target average distortion values for user 1 and user 2 are D_1 and D_2, respectively, with $D_1 \geq D_2$. Let r_1 and r_2 be the minimum compression rates that achieve D_1 and D_2, respectively; that is $r_i \triangleq R(D_i) = \frac{1}{2} \log_2 \frac{\sigma^2}{D_i}$, $i = 1, 2$. This means that, to achieve the target distortion of D_i, the user has to receive a minimum of nr_i bits corresponding to its desired file. We first present Lemma 1 specifying the lower bound on the delivery rate for given M_1 and M_2 in this particular scenario, followed by the coded caching scheme achieving this lower bound. The proof of the lemma is skipped due to space limitations.

Lemma 1. *For the lossy caching problem with $N = K = 2$, a lower bound on the cache capacity-delivery rate tradeoff is given by*

$$\begin{aligned} R^\star(M_1, M_2) \geq & R_c(M_1, M_2) = \max\{r_1 - M_1/2, \\ & r_2 - M_2/2, r_1 + r_2 - (M_1 + M_2), \\ & r_1/2 + r_2 - (M_1 + M_2)/2, 0\} \text{ bpss.} \end{aligned} \qquad (2)$$

The first three terms in (2) are derived from the cut-set lower bound, which will be presented for the general scenario in Theorem 1.

Based on (2), we consider five cases depending on the cache capacities of the users, illustrated in Fig. 1:

Case i: $M_1 + M_2 \leq r_1$. In this case, $R_c(M_1, M_2) = r_1 + r_2 - (M_1 + M_2)$ bpss.

Case ii: $M_1 + M_2 > r_1$, $M_1 \leq r_1$, $M_2 \leq 2r_2 - r_1$. We have $R_c(M_1, M_2) = \frac{r_1}{2} + r_2 - \frac{M_1+M_2}{2}$ bpss.

Case iii: $M_1 > r_1$, $M_2 \leq 2r_2$, $M_2 - M_1 \leq 2r_2 - 2r_1$. Then $R_c(M_1, M_2) = r_2 - \frac{M_2}{2}$ bpss.

Case iv: $M_1 \leq 2r_1$, $M_2 > 2r_2 - r_1$, $M_2 - M_1 > 2r_2 - 2r_1$. It yields $R_c(M_1, M_2) = r_1 - \frac{M_1}{2}$ bpss.

Case v: $M_1 > 2r_1$, $M_2 > 2r_2$. Then $R_c(M_1, M_2) = 0$.

Nest, for each of these cases, we explain the coded caching scheme that achieves the corresponding $R_c(M_1, M_2)$. We assume that the server employs an optimal successive refinement source code, denoted by $A(B)$ the source codeword of length

TABLE I
ILLUSTRATION OF CACHE PLACEMENT

	First Layer						Second Layer	
S_1	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
S_2	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
User 1	$A_1 \oplus B_1$		A_3, B_3		A_5, B_5			
User 2		$A_2 \oplus B_2$		A_4, B_4	A_5, B_5		A_7, B_7	
Case i	M_1	M_2	0	0	0	$r_1 - M_1 - M_2$	0	$r_2 - r_1$
Case ii	M_1	$r_1 - M_1$	0	0	0	0	$\frac{M_1+M_2-r_1}{2}$	$r_2 - \frac{M_1+M_2-r_1}{2}$
Case iii	$r_1 - l_1 - 2l_2$	0	l_2	l_2	l_1	0	$\min\{r_2 - r_1, M_2/2\}$	$\max\{0, r_2 - r_1 - M_2/2\}$
Case iv	0	$r_1 - M_1$	$M_1/2$	$M_1/2$	0	0	$r_2 - r_1$	0
Case v	0	0	0	0	r_1	0	$r_2 - r_1$	0

nr_2 bits that can achieve a distortion of D_2 for file $S_1(S_2)$. Thanks to the successive refinability of Gaussian sources, a receiver having received only the first nr_1 of these bits can achieve a distortion of D_1. We refer to the first nr_1 bits as the first layer, and the $n(r_2-r_1)$ remaining bits as the second layer.

In each case, we divide the first layers of codewords A and B into six disjoint parts denoted by A_1, ..., A_6 and B_1, ..., B_6, and the second layers into two disjoint parts denoted by A_7, A_8 and B_7, B_8, such that $|A_i| = |B_i|$ for $i = 1, ..., 8$, where $|X|$ denotes the length of the binary sequence X (normalized by n).

Table I illustrates the placement of contents in users' caches for each case.. The second and third rows illustrate how the first and second layers are partitioned for each file. The fourth and fifth rows indicate the cache contents of each user at the end of the *placement phase*. In all the cases, user 1 caches $Z_1 = \{A_1 \oplus B_1, A_3, B_3, A_5, B_5\}$ and user 2 caches $Z_2 = \{A_2 \oplus B_2, A_4, B_4, A_5, B_5, A_7, B_7\}$. The entries from the 6th row to the 10th specify the size of each portion in each case. For example, the 6th row implies that in *Case i*, $|A_1| = |B_1| = M_1$, $|A_2| = |B_2| = M_2$, $|A_6| = |B_6| = r_1 - M_1 - M_2$, $|A_8| = |B_8| = r_2 - r_1$, and the sizes of all other portions are equal to 0, which is equivalent to dividing $A(B)$ into four portions $A_1(B_1)$, $A_2(B_2)$, $A_6(B_6)$ and $A_8(B_8)$. Thus, in the placement phase, user 1 caches $Z_1 = \{A_1 \oplus B_1\}$, and user 2 caches $Z_2 = \{A_2 \oplus B_2\}$ so that $|Z_1| = M_1$ and $|Z_2| = M_2$, which meets the cache capacity constraints. The cache placements of the other 4 cases are presented in a similar manner in Table I.

Next, we focus on the delivery phase. We will explain the delivered message in each case to satisfy demands $\mathbf{d} = (S_1, S_2)$. All other requests can be satisfied similarly, without requiring higher delivery rates.

Case i ($M_1 + M_2 \leq r_1$): The server sends B_1, A_2, A_6, B_6 and B_8. Thus, the delivery rate is

$$R(M_1, M_2) = r_1 + r_2 - (M_1 + M_2) \text{ bpss.}$$

Case ii ($M_1 + M_2 > r_1$, $M_1 \leq r_1$, $M_2 \leq 2r_2 - r_1$): Server delivers B_1, A_2 and B_8. We have

$$R(M_1, M_2) = \frac{r_1}{2} + r_2 - \frac{M_1 + M_2}{2} \text{ bpss.}$$

Case iii ($M_1 > r_1$, $M_2 \leq 2r_2$, $M_2 - M_1 \leq 2r_2 - 2r_1$): The values of l_1 and l_2 in Table I are given as: $l_1 = \max\{0, \min\{M_1 - r_1, M_2/2 - (r_2 - r_1)\}\}$ and $l_2 = \max\{0, M_2/2 - (r_2 - r_1) - l_1\}$. The server sends B_1, $B_3 \oplus A_4$ and B_8 in the delivery phase, which results in

$$R(M_1, M_2) = r_2 - \frac{M_2}{2} \text{ bpss.}$$

Case iv ($M_1 \leq 2r_1$, $M_2 > 2r_2 - r_1$, $M_2 - M_1 > 2r_2 - 2r_1$): The server sends B_2, $B_3 \oplus A_4$ and we have

$$R(M_1, M_2) = r_1 - \frac{M_1}{2} \text{ bpss.}$$

Case v ($M_1 > 2r_1$, $M_2 > 2r_2$): The cache capacities of both users are sufficient to cache the required descriptions for both files. Thus, any request can be satisfied from local caches at desired distortion levels, and we have $R(M_1, M_2) = 0$.

Corollary 1. *For $N = K = 2$, the proposed caching scheme meets the lower bound in Lemma 1; and hence, it is optimal, i.e., we have $R^*(M_1, M_2) = R_c(M_1, M_2)$.*

IV. LOSSY CODED CACHING: GENERAL CASE

In this section, we tackle the lossy content caching problem in the general setting with N files and K users. Recall that the distortion requirements are assumed to be ordered as $D_1 \geq D_2 \geq \cdots \geq D_K$. Let $r_k = R(D_k)$, $k = 1, ..., K$. Exploiting the successive refinability of Gaussian sequences, we consider a layered structure of descriptions for each file, where the first layer, called the r_1-description, consists of nr_1 bits, and achieves distortion D_1 when decoded. The kth layer, called the $(r_k - r_{k-1})$-refinement, $k = 2, ..., K$, consists of $n(r_k - r_{k-1})$ bits, and having received the first k layers, a user achieves a distortion of D_k.

The example in Section III illustrates the complexity of the problem; we had five different cases even for two users and two files. The problem becomes intractable quickly with the increasing number of files and users. However, note that only users $k, k+1, ..., K$, whose distortion requirements are lower than D_k, need to decode the kth layer for the file they request, for $k = 1, ..., K$. Therefore, once all the contents are compressed into K layers based on the distortion requirements of the users employing an optimal successive refinement source code, we have, for each layer, a lossless caching problem. However, each user also has to decide how much of its cache capacity to allocate for each layer. Hence, the lossy caching problem is divided into two subproblems:

the lossless caching problem of each source coding layer, and the cache allocation problem among different layers.

A. Coded Lossless Caching of Each Layer

Here we focus on the first subproblem, and investigate centralized lossless caching with heterogeneous cache sizes, which is unsolved in the literature, regarding each layer separately. Consider, for example, the kth refinement layers of all the files. There are only $L_k \triangleq K-k+1$ users (users $k, k+1, ..., K$) who may request these layers. Let user j, $j \in \{k, ..., K\}$, allocate $M_{j,k}$ (normalized by n) of its cache capacity for this layer. Without loss of generality, we order users $k, ..., K$ according to the cache capacity they allocate, and re-index them, such that $M_{k,k} \leq M_{k+1,k} \leq \cdots \leq M_{K,k}$.

We would like to have symmetry among allocated cache capacities to enable multicasting to a group of users. Based on this intuition, we further divide layer k into L_k sub-layers, and let each user in $\{k, ..., K\}$ allocate $M_k^1 = M_{k,k}$ of its cache for the first sub-layer, and each user in $\{k+i-1, ..., K\}$ allocate $M_k^i = M_{k+i-1,k} - M_{k+i-2,k}$ of its cache for the ith sublayer, for $i = 2, \ldots, L_k$. Overall, we have L_k sub-layers, and users $k+i-1, k+i, ..., K$ allocate M_k^i of their caches for sub-layer i, whereas no cache is allocated by users $k, k+1, ..., k+i-2$.

We denote by r_k^i the size of the ith sub-layer of the kth refinement layer, and by $R(L_k, i, M_k^i, r_k^i, N)$ the minimum required delivery rate for this sub-layer. The rates, r_k^i, $i = 1, ..., L_k$, should be optimized jointly in order to minimize the total delivery rate for the kth layer. The optimization problem can be formulated as follows:

$$\min_{r_k^1, ..., r_k^{L_k}} \sum_{i=1}^{L_k} R(L_k, i, M_k^i, r_k^i, N) \tag{3a}$$

$$\text{s.t.} \sum_{i=1}^{L_k} r_k^i = r_k - r_{k-1}. \tag{3b}$$

We explore the achievable $R(L_k, i, M_k^i, r_k^i, N)$ based on the existing caching schemes in in [3] and [4], which are referred to as *coded delivery* and *coded placement*, respectively. We consider two cases:

Case 1) $L_k < N$. In this case, coded placement scheme of [4] provides no global caching gain. Thus, we employ only coded delivery, and illustrate this scheme in our setup by focusing on the ith sub-layer: users $k+i-1$ to K each allocate M_k^i of cache capacity, while users k to $k+i-2$ allocate no cache for this sublayer. If $r_k^i \in \{0, M_k^i/N, M_k^i L_k^i/((L_k^i - 1)N), M_k^i L_k^i/((L_k^i - 2)N), ..., M_k^i L_k^i/N\}$, where $L_k^i = L_k + 1 - i$, we have

$$R(L_k, i, M_k^i, r_k^i, N) = (i-1) \cdot r_k^i + r_k^i L_k^i \cdot (1 - M_k^i/r_k^i N) \cdot \frac{1}{1 + M_k^i L_k^i / r_k^i N}. \tag{4}$$

The first term on the right hand side is due to unicasting to users k to $k+i-2$, while the second term is the *coded delivery* rate to users $k+i-1$ to K given in [3]. Based on the memory sharing argument, any point on the line connecting two points, $(r_1', R(L_k, i, M_k^i, r_1', N))$ and $(r_2', R(L_k, i, M_k^i, r_2', N))$, is also achievable, i.e., if $r_k^i \in [r_1', r_2']$, then we have

$$R(L_k, i, M_k^i, r_k^i, N) = \frac{r_k^i - r_1'}{r_2' - r_1'} R(K_k, i, M_k^i, r_1', N) + \frac{r_2' - r_k^i}{r_2' - r_1'} R(K_k, i, M_k^i, r_2', N), \tag{5}$$

where $r_1', r_2' \in \{0, M_k^i/N, M_k^i L_k^i/(L_k^i - 1)N, M_k^i L_k^i/(L_k^i - 2)N, ..., M_k^i L_k^i/N\}$; and if $r_k^i > M_k^i L_k^i/N$, we have

$$R(L_k, i, M_k^i, r_k^i, N) = (i-1) \cdot r_k^i + \frac{M_k^i L_k (L_k - 1)}{2N} + (r_k^i M_k^i L_k/N) \times (L_k - i + 1).$$

Case 2) $L_k \geq N$. In this case, *coded placement* outperforms *coded delivery* if the allocated cache capacity satisfies $M_k^i \leq \frac{r_k^i}{L_k^i}$ [4]. Note that for the ith sub-layer, there are $i-1$ users with no cache allocation. If $i - 1 \geq N$, there will be no gain with either schemes. When $i - 1 < N$ and $r_k^i \geq M_k^i L_k^i$, the delivery rate of *coded placement* is

$$R(L_k, i, M_k^i, r_k^i, N) = N r_k^i - M_k^i r_k^i (N - i + 1). \tag{6}$$

When $0 \leq r_k^i \leq M_k^i L_k^i$, the delivery rate is given by the lower convex envelope of points $(M_k^i L_k^i, R(L_k, i, M_k^i, M_k^i L_k^i, N))$ given by (6) and $(r_k^i, R(L_k, i, M_k^i, r_k^i, N))$, and for $r_k^i \in \{0, M_k^i/N, M_k^i L_k^i/((L_k^i - 1)N), M_k^i L_k^i/((L_k^i - 2)N), ..., M_k^i L_k^i/N\}$, given by (4).

B. Allocation of Cache Capacity

We propose two algorithms for cache allocation among layers: *proportional cache allocation* (PCA) and *ordered cache allocation* (OCA), which are elaborated in Algorithms 1 and 2, respectively, where r_k is as defined earlier, and we let $r_0 = 0$.

.

Algorithm 1 Proportional Cache Allocation (PCA)

Require: $\mathbf{r} = r_1, ..., r_K$

1) for all $k \in 1, ..., K$
2) for all $i \in 1, ..., k$
3) user k allocates $\frac{r_i - r_{i-1}}{r_k} M_k$ to layer i
4) end for
5) end for

Algorithm 2 Ordered Cache Allocation (OCA)

Require: $\mathbf{r} = r_1, ..., r_K$

1) for all $k \in 1, ..., K$
2) user k allocates all of its cache to the first i layers, where $r_{i-1} < \frac{M_k}{N} \leq r_i$
3) end for

PCA allocates each user's cache among the layers it may request proportionally to the sizes of the layers, while OCA gives priority to lower layers. The server can choose the one resulting in a lower delivery rate. Numerical comparison of these two allocation schemes will be presented in Section V.

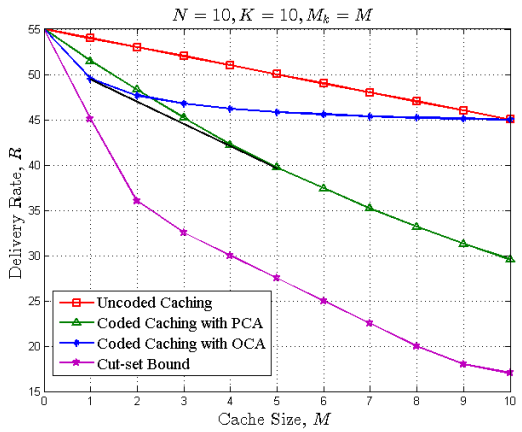

Fig. 2. Delivery rate vs. cache capacity with identical cache sizes.

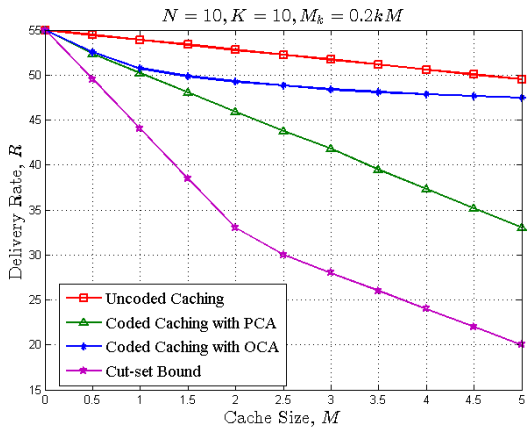

Fig. 3. Delivery rate vs. cache capacity with heterogeneous cache sizes.

C. Lower Bound

The following lower bound is obtained using cut-set arguments.

Theorem 1. *(Cut-set Bound) For the lossy caching problem described in Section II, the optimal achievable delivery rate is lower bounded by*

$$\max_{s\in\{1,\dots,K\}} \max_{\substack{\mathcal{U}\subset\{1,\dots,K\}\\ |\mathcal{U}|=s}} \left(\sum_{k\in\mathcal{U}} r_k - \frac{\sum_{k\in\mathcal{U}} M_k}{\lfloor N/s \rfloor} \right).$$

V. Simulations

In this section, we numerically compare the achievable delivery rates for uncoded caching, the proposed caching schemes, and the lower bound. In Fig. 2, we consider $K = 10$ users and $N = 10$ files in the server. Cache sizes of the users are identical, i.e., $M_1 = M_2 = \cdots = M_{10} = M$. The distortion levels $(D_1, D_2, ..., D_{10})$ are such that $(r_1, r_2, ..., r_{10}) = (1, 2, ..., 10)$. While we observe that the proposed coded caching scheme greatly reduces the delivery rate, OCA performs better for small cache sizes, while PCA dominates as M increases. Using memory sharing, we can argue that the dotted curve in Fig. 2, which is obtained through the convex combination of the delivery rates achieved by the two proposed schemes, is also achievable.

In Fig. 3, we consider the same setting but with heterogeneous cache sizes, where $M_k = 0.2kM$, for $k = 1, ..., 10$. In this setting, PCA allocates the same amount of cache to each layer at different users, which creates symmetry among the caches. The achievable delivery rates in Fig. 3 illustrate significant improvements in coded caching with PCA over both uncoded and OCA schemes in terms of the achievable delivery rates. We observe that the gains become more significant as the cache capacity, M, increases. While the lower bound is not tight in general, we see in both figures that the PCA performance follows the lower bound with an approximately constant gap over the range of M values considered.

VI. Conclusion

We investigated the lossy caching problem where users have different distortion requirements for the reconstruction of contents they request. We proposed a coded caching scheme that achieves the information-theoretic lower bound for the special case with two users and two files. Then, we tackled the general case with K users and N files in two steps: delivery rate minimization, which finds the minimum delivery rate for each layer separately, and cache allocation among layers. We proposed two different algorithms for the latter, namely, PCA and OCA. Our simulation results have shown that the proposed PCA scheme improves the required delivery rate significantly for a wide range of cache capacities; and particularly when the users' cache capacities are heterogenous.

References

[1] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch and G. Caire, "Femtocaching: wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, Orlando, FL, Mar. 2012, pp.1107–1115.

[2] M. Gregori, J. Gomez-Vilardebo, J. Matamoros and D. Gündüz, "Wireless content caching for small cell and D2D networks," *to appear, IEEE J. Sel. Areas Commun.*, 2016.

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

[4] Z. Chen, P. Fan and K. B. Letaief, "Fundamental limits of caching: Improved bounds for small buffer users," *ArXiv:1407.1935v2 cs.IT*, Nov. 2015.

[5] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw*, vol. 23, no. 4, pp. 1029-1040 Apr. 2014.

[6] R. Pedarsani, M. Maddah-Ali and U. Niesen, "Online coded caching," *ArXiv:1311.3646 cs.IT*, Nov. 2013.

[7] M. Ji, G. Caire and A. F. Molisch, "Fundamental limits of distributed caching in D2D wireless networks," in *Proc. IEEE Inform. Theory Workshop (ITW)*, Jeju Island, Korea, Oct. 2015, pp. 1–5.

[8] P. Hassanzadeh, E. Erkip, J. Llorca and A. Tulino, "Distortion-memory tradeoffs in cache-aided wireless video delivery," *ArXiv:1511.03932 cs.IT*, Nov. 2015.

[9] R. Timo, S. S. Bidokthi, M. Wigger and B. Geiger, "A rate-distortion approach to caching," in *Proc. Int'l. Zurich Seminar (IZS)*, Zurich, Switzerland, Mar. 2016.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 2012.

[11] T. M. Cover and W. H. Equitz, "Successive refinement of information," *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 269–275 , Mar. 1991.