Centralized Coded Caching of Correlated Contents

Qianqian Yang and Deniz Gündüz
Information Processing and Communications Lab
Department of Electrical and Electronic Engineering
Imperial College London

***Abstract*—Coded caching and delivery is studied taking into account the correlations among the contents in the library. Correlations are modeled as common parts shared by multiple contents; that is, each file in the database is composed of a group of subfiles, where each subfile is shared by a different subset of files. The number of files that include a certain subfile is defined as the *level of commonness* of this subfile. First, a correlation-aware *uncoded* caching scheme is proposed, and it is shown that the optimal placement for this scheme gives priority to the subfiles with the highest levels of commonness. Then a correlation-aware *coded* caching scheme is presented, and the cache capacity allocated to subfiles with different levels of commonness is optimized in order to minimize the delivery rate. The proposed correlation-aware coded caching scheme is shown to remarkably outperform state-of-the-art correlation-ignorant solutions, indicating the benefits of exploiting content correlations in coded caching and delivery in networks.**

I. Introduction

In *proactive caching*, popular contents are stored in user devices during off-peak traffic periods even before they are requested by the users [1]–[3]. Proactive caching is considered as a promising solution for the recent explosive growth of wireless data traffic, and can alleviate both the network congestion and the latency during peak traffic periods (see [1]–[4], and references therein).

Proactive caching typically takes place in two phases: the first phase takes place during off-peak traffic periods, when users' caches are filled as a function of the whole library of files, referred to as the *placement phase*; while the second, *delivery phase*, takes place during the peak traffic period when the users' demands are revealed and satisfied simultaneously. In contrast to traditional uncoded caching schemes, which simply employ orthogonal unicast transmissions during the *delivery phase*, recently proposed *coded caching* [4] creates coded multicasting opportunities, significantly reducing the amount of data that needs to be delivered to the users to satisfy their demands, even when these demands are distinct. Coded caching benefits from the aggregate cache capacity across the network, rather than local cache capacities as in conventional uncoded caching [4]. This significant improvement has motivated intense research interest on coded caching in recent years [5]–[12]. Some works follow the simplified model proposed in [4], and aim to improve the fundamental limits of caching [6], [7], [13]; others consider more realistic settings, such as decentralized caching [5], nonuniform popularities across files [9], [10], audience retention rate aware caching [14], or heterogeneous quality of service requirements [12].

An important feature of video contents, which is the main source of the recent explosive traffic growth, is that, there may be significant overlaps among different files, e.g., the recordings of the same event from different angles and different cameras, or even the frames of the same scene in the same video. In [15], Hassanzadeh et al. propose a correlation-aware caching scheme, which groups the contents in the library into two sets according to their correlations as well as popularity, where each file in second set is compressed with respect to a file in the first set. This scheme is shown to outperform correlation-ignorant caching schemes. A more information theoretic formulation for caching of correlated sources is considered in [16], focusing on a special scenario with two receivers and one cache. A similar information-theoretic analysis is carried out in [17] for two files and two receivers, each with its own cache.

In this paper, we consider a server with a library of N correlated files, serving K users equipped with local caches. Different from [15], which only exploits a fixed level of common information among the files, we consider a more general model in order to fully exploit the potential correlations among different subsets of files. We model each file in the server to be composed of a group of subfiles, such that each subfile is shared by a different subset of files. The number of files to which each subfile belongs is defined as its *level of commonness*. Equivalently, in our model, any subset of files $\mathcal{S}$ share a common part that is independent of the rest of the library, and shared exclusively by the files in $\mathcal{S}$.

We first propose a correlation-aware *uncoded* caching scheme, and show that the optimal placement for this scheme is achieved by giving priority to the subfiles with the highest levels of commonness in the placement phase. We then propose a correlation-aware *coded* caching scheme, and derive a closed-form expression of the achievable delivery rate, based on which the cache capacity is optimally allocated to the subfiles according to their level of commonness. Then, we compare the performance of the proposed correlation-aware schemes with those that ignore the correlation, and the cut-set bound; and show that, exploiting file correlations in coded caching can significantly reduce the delivery rate.

Notations: The set of integers $\{i, ..., j\}$, where $i \leq j$, is denoted by $[i:j]$, particularly, $\{1, ..., j\}$ is denoted by $[j]$. For sets $\mathcal{A}$ and $\mathcal{B}$, we define $\mathcal{A} \backslash \mathcal{B} \triangleq \{x : x \in \mathcal{A}, x \notin \mathcal{B}\}$, and $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$. $\binom{j}{i}$ represents the binomial coefficient if $j \geq i$; otherwise, $\binom{j}{i} = 0$. For event E, $\mathbb{1}\{E\} = 1$ if E is true; and $\mathbb{1}\{E\} = 0$, otherwise.

II. System Model

We consider a server with a database of N correlated files, $W_1, ..., W_N$, where each file consists of 2^{N-1} independent subfiles, e.g., $W_i = \bigcup_{\substack{\mathcal{S}\subset[N]\\ i\in\mathcal{S}}} \overline{W}_{\mathcal{S}}$, $\forall i \in [N]$. Here, $\overline{W}_{\mathcal{S}}$ denotes the subfile shared exclusively by the subset of files $\{W_i : i \in \mathcal{S}\}$. For simplicity, we assume that for $\mathcal{S} \subset [N]$, $|\overline{W}_{\mathcal{S}}| = F_l$, if $|\mathcal{S}| = l$, i.e., the common subfiles shared exclusively by l files are of the same size of F_l bits. Let $\mathbf{F} \triangleq (F_1, \ldots, F_N)$. As a result, each file in the library is also of the same size of F bits, given by

$$F = \sum_{l=1}^{N} \binom{N-1}{l-1} F_l. \qquad (1)$$

For $\mathcal{S} \subset [N]$, $|\mathcal{S}| = l$, we say that the subfiles $\overline{W}_{\mathcal{S}}$ have a commonness level of l. For example, $\overline{W}_{\{1,2,3\}}$ and $\overline{W}_{\{3,4,5\}}$ both have level 3 commonness. For brevity, we refer to all the subfiles with level l commonness as l-subfiles, $l = 1, ..., N$. We consider K users connected to the server through a shared, error-free link, each equipped with a cache of size MF bits.

We consider centralized caching; that is, the server has the knowledge of the active users during the placement phase, though not the knowledge of their demands. Centralized caching allows the server to fill the user caches in a coordinated manner. After the placement phase, each user requests a single file from the library, where $d_k \in [N]$ denotes user k's request, $k \in [K]$. All the requests are satisfied simultaneously over the error-free shared link.

An $(\mathbf{F}, M, R)$ caching code for this system consists of:

- K **caching functions** f_k, $k \in [K]$,

$$f_k : \underbrace{[2^F] \times \cdots \times [2^F]}_{N \text{ files}} \to [2^{MF}], \qquad (2)$$

  such that the contents of user k's cache at the end of the placement phase, denoted by Z_k, is given by $Z_k = f_k(\{W_i\}_{i=1}^N)$;
- **a delivery function** g,

$$g : \underbrace{[2^F] \times \cdots \times [2^F]}_{N \text{ files}} \times \mathbf{D} \to [2^{RF}], \qquad (3)$$

  where $\mathbf{D} \triangleq (d_1, ..., d_K)$, such that a single message of RF bits, $X_{\mathbf{D}} = g((W_1, ..., W_N), \mathbf{D})$, is sent by the server over the shared link according to users' demands;
- K **decoding functions** h_k, $k \in [K]$,

$$h_k : \mathbf{D} \times [2^{MF}] \times [2^{RF}] \to [2^F], \qquad (4)$$

  where $\hat{W}_{d_k} = h_k(\mathbf{D}, Z_k, X_{\mathbf{D}})$, is the reconstruction of W_{d_k} at user k.

Definition 1. *A user cache capacity-delivery rate pair (M, R) is achievable for a system described above, if there exists a sequence of $(\mathbf{F}, M, R)$ codes such that for any demand realization $\mathbf{D} \subset [N]^K$,*

$$\lim_{F_1,...,F_N \to \infty} \Pr \left\{ \bigcup_{k\in[K]} \left\{ \hat{W}_{d_k} \neq W_{d_k} \right\} \right\} = 0. \qquad (5)$$

For a system with N files and K users, our goal is to characterize the minimum achievable rate R as a function of the user cache capacity M, i.e., $R^*(M) \triangleq \inf\{R : (M, R)$ is achievable$\}$.

III. Correlation-aware Uncoded Caching and Delivery (CAUC) Scheme

We first present an uncoded caching and delivery scheme exploiting the correlation among files, referred to as CAUC.

1) Placement phase: Each user caches the same $p_l F_l$ bits from each l-subfile, where $0 \le p_l \le 1$, $l \in [N]$, such that

$$MF = \sum_{l=1}^{N} \binom{N}{l} p_l F_l, \qquad (6)$$

which meets the limitation of the cache capacities. We refer to $\mathbf{P} \triangleq (p_1, ..., p_N)$ as the *cache allocation vector*, which will be specified in the sequel.

2) Delivery phase: The server delivers the remaining bits of each requested subfile that have not been cached by the users, i.e., $\overline{W}_{\mathcal{S}}$ for which $\sum_{k=1}^{K} \mathbb{1}\{d_k \in \mathcal{S}\} \ge 1$.

In the worst case, when the demand combination is the most distinct, i.e., users request distinct files for the case $N \ge K$, or each file is requested by at least one user for the case $N < K$, the delivery rate is given by

$$R_{CAUC}(\mathbf{P}) = \sum_{l=1}^{N} (1 - p_l) F_l \left(\binom{N}{l} - \binom{\min\{N-K, 0\}}{l} \right). \qquad (7)$$

The optimal $\mathbf{P}^*$ can be derived by solving the following optimization problem

$$\begin{aligned} \min \quad & R_{CAUC}(\mathbf{P}) \\ \text{such that} \quad & \sum_{l=1}^{N} \binom{N}{l} p_l F_l \le MF, \end{aligned} \qquad (8)$$

which, straightforwardly, leads to: $p_l^* = 1$, if $C(l) \le MF$; $p_l^* = \frac{MF - C(l+1)}{\binom{N}{l} F_l}$, if $C(l+1) < MF < C(l)$; and $p_l^* = 0$, otherwise; where we have defined $C(l) \triangleq \sum_{i=l}^{N} \binom{N}{i} F_i$, for $l \in [N]$. We remark that the optimal cache allocation gives priority to the subfiles with the highest level of commonness.

IV. Correlation-aware Coded Caching and Delivery (CACC) Scheme

In this section, we present a correlation-aware coded caching and delivery scheme, referred to as CACC. Similarly to the CAUC scheme, we allocate different cache capacities to subfiles of different levels of commonness, again specified by the cache allocation vector, $\mathbf{P} = (p_1, ..., p_N)$, which satisfies the constraint in (6), such that each user caches $p_l F_l$ bits from each l-subfile, $l \in [N]$. In the following, we first present how coded caching and delivery of the subfiles with the same level of commonness is carried out, and then specify the allocation of cache capacity.

A. Coded Caching and Delivery of l-subfiles

Here, for a given cache allocation vector $\mathbf{P}$, we present the coded caching and delivery of l-subfiles, $l \in [N]$. We define $t_l \triangleq Kp_l$, $0 \leq t_l \leq K$. If $t_l = 0$, users do not cache the l-subfiles at all, while if $t_l = K$, each user stores all the l-subfiles in its cache. In the following, we focus on the cases where $t_l \in [K-1]$.

1) Placement Phase: We employ the prefetching scheme proposed by [4] for the subfiles rather than the files themselves: each l-subfile is partitioned into $\binom{K}{t_l}$ disjoint parts, each with approximately the same size of $F_l / \binom{K}{t_l}$ bits. We label these $\binom{K}{t_l}$ disjoint parts of each l-subfile $\overline{W}_{\mathcal{S}}$ by $\overline{W}_{\mathcal{S}}^{\mathcal{A}}$, where $|\mathcal{A}| = t_l$, $\mathcal{A} \subset [K]$; that is, we have $\overline{W}_{\mathcal{S}} = \bigcup_{\mathcal{A}:|\mathcal{A}|=t_l,\ \mathcal{A}\subset[K]} \overline{W}_{\mathcal{S}}^{\mathcal{A}}$. Each of these parts, $\overline{W}_{\mathcal{S}}^{\mathcal{A}}$, is placed into the cache of user k if $k \in \mathcal{A}$. Thus, each user caches a total of $\binom{K-1}{t_l-1}$ disjoint parts of each l-subfile with a total size of $t_l F_l / K$ bits, which sums up to $p_l F_l$ bits.

2) Delivery Phase: We first focus on the case when $N \leq K$. There are a total of $\binom{N}{l}$ l-subfiles. We denote the set of these l-subfiles by $\mathcal{W}^l = \{\overline{W}_{\mathcal{S}} : \mathcal{S} \subset [N], |\mathcal{S}| = l\}$. Each user requires a total of $\binom{N-1}{l-1}$ l-subfiles, i.e., user k needs to recover subfiles in $\{\overline{W}_{\mathcal{S}} : \mathcal{S} \subset [N], |\mathcal{S}| = l, d_k \in \mathcal{S}\}$, $\forall k \in [K]$. For each user, we can regard these $\binom{N-1}{l-1}$ l-subfiles as $\binom{N-1}{l-1}$ distinct demands. Our delivery scheme for the l-subfiles operates in $\binom{N-1}{l-1}$ steps, and satisfies one demand of each user at each step.

We define $\mathbf{C}_j \triangleq (c_{1j}, ..., c_{Nj})$, where $c_{ij} \in \{\mathcal{S} : \mathcal{S} \subset [N], |\mathcal{S}| = l, i \in \mathcal{S}\}$, $\forall i \in [N]$, $j \in [\binom{N-1}{l-1}]$, which specifies which subfile should be delivered in the jth step of the delivery phase. $\mathbf{C}_j$ is generated by Algorithm 1 by setting $\mathcal{R} = [N]$ and $\overline{\mathcal{R}} = \emptyset$. Note that these vectors are generated independently of the number of users or their demands.

Algorithm 1 Generate $\mathbf{C}_j$, $j \in [\binom{|\mathcal{R}|-1}{l-\overline{\mathcal{R}}-1}]$

```
procedure ASSIGNMENT
    $\mathcal{W}_t \leftarrow \mathcal{W}^l$, $j \leftarrow 1$, $W_{last} \leftarrow \emptyset$, $c_{last} \leftarrow \emptyset$
    while $\mathcal{W}_t \neq \emptyset$ do
        $c_t \leftarrow \mathcal{R}$
        while $c_t \neq \emptyset$ do
            if $|c_t| \geq l - |\overline{\mathcal{R}}|$ then
                if $c_{last} = \emptyset$ then
                    Randomly select one $l$-subfile $\overline{W}_{\mathcal{S}}$ from $\mathcal{W}_t$ such that $\mathcal{S} \setminus \overline{\mathcal{R}} \subset c_t$, remove $\overline{W}_{\mathcal{S}}$ from $\mathcal{W}_t$, and $c_t \leftarrow c_t \setminus \mathcal{S}$
                    for $i \in \mathcal{S} \setminus \overline{\mathcal{R}}$ do
                        $c_{ij} \leftarrow W_{last}$
                    end for
                else
                    for $i \in c_{last} \setminus \overline{\mathcal{R}}$ do
                        $c_{ij} \leftarrow \mathcal{S}$
                    end for
                    $c_t \leftarrow c_t \setminus c_{last}$, $W_{last} \leftarrow -1$, and $c_{last} \leftarrow \emptyset$
                end if
            else
                Randomly select one $l$-subfile $\overline{W}_{\mathcal{S}}$ from $\mathcal{W}_t$ such that $c_t \subset \mathcal{S}$
                for $i \in c_t$ do
                    $c_{ij} \leftarrow \mathcal{S}$
                end for
                for $i \in \overline{\mathcal{R}}$ do
                    $c_{ij} \leftarrow \emptyset$
                end for
                $W_{last} \leftarrow \mathcal{S}$, $c_{last} \leftarrow \mathcal{S} \setminus c_t$, $c_t \leftarrow \emptyset$, and $j \leftarrow j + 1$
            end if
        end while
    end while
end procedure
```

Example 1. Consider $N = 5$ and $l = 2$. From Algorithm 1 we obtain:

$$\begin{aligned}
\mathbf{C_1} &= (\{1,2\},\{1,2\},\{3,4\},\{3,4\},\{1,5\});\\
\mathbf{C}_2 &= (\{1,5\},\{2,3\},\{2,3\},\{4,5\},\{4,5\});\\
\mathbf{C}_3 &= (\{1,3\},\{2,5\},\{1,3\},\{2,4\},\{2,5\});\\
\mathbf{C}_4 &= (\{1,4\},\{2,4\},\{3,5\},\{1,4\},\{3,5\}).
\end{aligned}$$

This means, for example, that, in the first step, subfiles W_{12}, W_{34}, and W_{15} will be delivered (if there is a user requesting them).

We denote by $d_k^j \triangleq c_{d_k j}$ the demand of user k to be satisfied in the jth step, i.e., user k recovers $\overline{W}_{d_k^j}$ after the jth step. We emphasize that $\bigcup_{j=1}^{\binom{N-1}{l-1}} \overline{W}_{c_{ij}} = \{\overline{W}_{\mathcal{S}} : \mathcal{S} \in [N], |\mathcal{S}| = l, i \in \mathcal{S}\}$, $\forall i \in [N]$; that is all the required l-subfiles will be recovered by each user after step $\binom{N-1}{l-1}$ for any demand combination.

Example 2. Consider $N = K = 5$ and $l = 2$ as in Example 1. Consider distinct demands, i.e., $\mathbf{D} = \{1,2,3,4,5\}$. Thus, based on $\mathbf{C}_1$, we have $d_1^1 = d_2^1 = \{1,2\}$, $d_3^1 = d_4^1 = \{3,4\}$, and $d_5^1 = \{1,5\}$; that is, at the end of the first step, users 1 and 2 should recover W_{12}, users 3 and 4 should recover W_{34}, while user 5 should recover W_{15}.

Example 3. With the same setting as in Example 2, consider now a non-distinct demand combination $\mathbf{D} = \{1,1,1,3,4\}$. Based on $\mathbf{C}_1$, we have $d_1^1 = d_2^1 = d_3^1 = \{1,2\}$, and $d_4^1 = d_5^1 = \{3,4\}$; that is, at the end of the first step users 1, 2 and 3 should recover W_{12}, while users 4 and 5 should recover W_{34}.

Based on the delivery scheme proposed in [13], we present our coded transmission scheme in Algorithm 2 according to $\mathbf{C}_j$, $j \in [\binom{N-1}{l-1}]$, where $\overline{\mathcal{R}} = \emptyset$, $\mathcal{R} = [N]$. In Algorithm 2, we define A_j as the number of distinct d_k^j for each $j \in [\binom{N-1}{l-1}]$. We note that, among the CODED DELIVERY and RANDOM

Algorithm 2 Coded transmission based on $\mathbf{C}_j$

```
procedure CODED DELIVERY
    for $k = 1, ..., K$ do
        $d_k^j \leftarrow c_{d_k j}$
    end for
    $\mathcal{U}_j \leftarrow$ Any subset of $A_j$ users with distinct $d_k^j$
    for $\mathcal{V} \subset [K] : |\mathcal{V}| = t_l + 1, \sum_{k \in \mathcal{U}_j} \mathbb{1}\{k \in \mathcal{V}\} \geq 1$ do
        Send $\bigoplus_{k \in \mathcal{V}} \overline{W}_{d_k^j}^{\mathcal{V} \setminus \{k\}}$.
    end for
end procedure

procedure RANDOM DELIVERY
    for $\mathcal{S} \subset \mathcal{R} : |\mathcal{S}| = l - |\overline{\mathcal{R}}|$ do
        Server sends enough random linear combinations of the bits of $l$-subfile $\overline{W}_{\mathcal{S} \cup \overline{\mathcal{R}}}$ to enable the users demanding it to decode it.
    end for
end procedure
```

DELIVERY procedures of Algorithm. 2, the one that requires a smaller delivery rate is performed.

Example 2 - continued. In Example 2, assume that $t_l = 1$, i.e., each l-subfile is divided into K disjoint parts of equal size, and each disjoint part is cached exactly by one user. Based on $\mathbf{D}$, we have $A_1 = 3$. Assume that $\mathcal{U}_1 = \{1, 3, 5\}$. Then, the server sends $\overline{W}^{\{1\}}_{\{1,2\}} \bigoplus \overline{W}^{\{2\}}_{\{1,2\}}$, $\overline{W}^{\{1\}}_{\{3,4\}} \bigoplus \overline{W}^{\{3\}}_{\{1,2\}}$, $\overline{W}^{\{1\}}_{\{3,4\}} \bigoplus \overline{W}^{\{4\}}_{\{1,2\}}$, $\overline{W}^{\{1\}}_{\{1,5\}} \bigoplus \overline{W}^{\{5\}}_{\{1,2\}}$, $\overline{W}^{\{2\}}_{\{3,4\}} \bigoplus \overline{W}^{\{3\}}_{\{1,2\}}$, $\overline{W}^{\{3\}}_{\{3,4\}} \bigoplus \overline{W}^{\{4\}}_{\{3,4\}}$, $\overline{W}^{\{5\}}_{\{3,4\}} \bigoplus \overline{W}^{\{3\}}_{\{1,5\}}$, $\overline{W}^{\{2\}}_{\{1,5\}} \bigoplus \overline{W}^{\{5\}}_{\{1,2\}}$, $\overline{W}^{\{4\}}_{\{1,5\}} \bigoplus \overline{W}^{\{5\}}_{\{3,4\}}$. By receiving these coded bits, users 1 and 2 can recover $\overline{W}_{\{1,2\}}$ together with the contents of their own caches. Similarly, users 3 and 4 can recover $\overline{W}_{\{3,4\}}$, while user 5 recovers $\overline{W}_{\{1,5\}}$. In the same manner, by coded transmission based on $\mathbf{C}_2$, user 1 can recover $\overline{W}_{\{1,5\}}$, users 2 and 3 recover $\overline{W}_{\{2,3\}}$, and users 4 and 5 recover $\overline{W}_{\{4,5\}}$ in the second step. After four delivery steps based on $\mathbf{C}_1, \ldots, \mathbf{C}_4$ each user decodes all the 2-subfiles of their requests. The total number of bits delivered in these four steps is $36F_2/5$.

Example 3 - continued. Assume again that $t = 1$. Based on $\mathbf{D}$, we have $A_1 = 2$, and let $\mathcal{U}_1 = \{1, 4\}$. Then, the server sends $\overline{W}^{\{1\}}_{\{1,2\}} \bigoplus \overline{W}^{\{2\}}_{\{1,2\}}$, $\overline{W}^{\{1\}}_{\{1,2\}} \bigoplus \overline{W}^{\{3\}}_{\{1,2\}}$, $\overline{W}^{\{1\}}_{\{3,4\}} \bigoplus \overline{W}^{\{4\}}_{\{1,2\}}$, $\overline{W}^{\{1\}}_{\{3,4\}} \bigoplus \overline{W}^{\{5\}}_{\{1,2\}}$, $\overline{W}^{\{2\}}_{\{3,4\}} \bigoplus \overline{W}^{\{4\}}_{\{1,2\}}$, $\overline{W}^{\{3\}}_{\{3,4\}} \bigoplus \overline{W}^{\{4\}}_{\{1,2\}}$, $\overline{W}^{\{5\}}_{\{3,4\}} \bigoplus \overline{W}^{\{4\}}_{\{3,4\}}$, such that users $1, 2$ and 3 can recover $\overline{W}_{\{1,2\}}$, while users 4 and 5 can recover $\overline{W}_{\{3,4\}}$. Based on $\mathcal{C}_2$, the server sends $\overline{W}^{\{1\}}_{\{1,5\}} \bigoplus \overline{W}^{\{2\}}_{\{1,5\}}$, $\overline{W}^{\{1\}}_{\{1,5\}} \bigoplus \overline{W}^{\{3\}}_{\{1,5\}}$, $\overline{W}^{\{1\}}_{\{2,3\}} \bigoplus \overline{W}^{\{4\}}_{\{1,5\}}$, $\overline{W}^{\{1\}}_{\{4,5\}} \bigoplus \overline{W}^{\{5\}}_{\{1,5\}}$, $\overline{W}^{\{2\}}_{\{2,3\}} \bigoplus \overline{W}^{\{4\}}_{\{1,5\}}$, $\overline{W}^{\{3\}}_{\{2,5\}} \bigoplus \overline{W}^{\{4\}}_{\{1,5\}}$, $\overline{W}^{\{5\}}_{\{2,3\}} \bigoplus \overline{W}^{\{4\}}_{\{4,5\}}$, $\overline{W}^{\{2\}}_{\{4,5\}} \bigoplus \overline{W}^{\{5\}}_{\{1,5\}}$, $\overline{W}^{\{3\}}_{\{4,5\}} \bigoplus \overline{W}^{\{5\}}_{\{1,5\}}$, such that users $1, 2$ and 3 can recover $\overline{W}_{\{1,5\}}$, while user 4 and user 5 can recover $\overline{W}_{\{2,3\}}$ and $\overline{W}_{\{4,5\}}$, respectively. Based on $\mathbf{C}_3$, the server sends $\overline{W}^{\{1\}}_{\{1,3\}} \bigoplus \overline{W}^{\{2\}}_{\{1,3\}}$, $\overline{W}^{\{1\}}_{\{1,3\}} \bigoplus \overline{W}^{\{3\}}_{\{1,3\}}$, $\overline{W}^{\{1\}}_{\{1,3\}} \bigoplus \overline{W}^{\{4\}}_{\{1,3\}}$, $\overline{W}^{\{1\}}_{\{2,4\}} \bigoplus \overline{W}^{\{5\}}_{\{1,3\}}$, $\overline{W}^{\{2\}}_{\{2,4\}} \bigoplus \overline{W}^{\{5\}}_{\{1,3\}}$, $\overline{W}^{\{3\}}_{\{2,4\}} \bigoplus \overline{W}^{\{5\}}_{\{1,3\}}$, $\overline{W}^{\{4\}}_{\{2,4\}} \bigoplus \overline{W}^{\{5\}}_{\{1,3\}}$, based on which users 1,2,3 and 4 can recover $\overline{W}_{\{1,3\}}$, while user 5 recovers $\overline{W}_{\{2,4\}}$. Finally, based on $\mathbf{C}_4$, the server sends $\overline{W}^{\{1\}}_{\{1,4\}} \bigoplus \overline{W}^{\{2\}}_{\{1,4\}}$, $\overline{W}^{\{1\}}_{\{1,4\}} \bigoplus \overline{W}^{\{3\}}_{\{1,4\}}$, $\overline{W}^{\{1\}}_{\{1,4\}} \bigoplus \overline{W}^{\{5\}}_{\{1,4\}}$, $\overline{W}^{\{1\}}_{\{3,5\}} \bigoplus \overline{W}^{\{4\}}_{\{1,4\}}$, $\overline{W}^{\{2\}}_{\{3,5\}} \bigoplus \overline{W}^{\{4\}}_{\{1,4\}}$, $\overline{W}^{\{3\}}_{\{3,5\}} \bigoplus \overline{W}^{\{4\}}_{\{1,4\}}$, $\overline{W}^{\{5\}}_{\{3,5\}} \bigoplus \overline{W}^{\{4\}}_{\{1,4\}}$, such that users $1, 2, 3$ and 4 are able to recover $\overline{W}_{\{1,4\}}$, while user 4 recovers $\overline{W}_{\{3,5\}}$. Thus, all the users are able to decode the 2-subfiles they requested. The total number of bits delivered in this case is $30F_2/5$.

Next, we consider the case $N > K$. We first select a subset of K files $\mathcal{R}$, $\mathcal{R} \subset [N]$, such that $|\mathcal{R}| = K$, and $d_k \in \mathcal{R}$ for $k = 1, ..., K$. For any subset $\overline{\mathcal{R}} \in [N] \setminus \mathcal{R}$, $|\overline{\mathcal{R}}| = s$, $s \in [\max\{l - K, 0\} : \min\{l - 1, N - K\}\}]$, Algorithm 1 is applied to a subset of l-subfiles $\mathcal{W}^l = \{\overline{W}_{\mathcal{S} \cup \overline{\mathcal{R}}} : \mathcal{S} \subset \mathcal{R}, |\mathcal{S}| = l - |\overline{\mathcal{R}}|\}$ to derive $\mathbf{C}_j$, $j \in [\binom{|\mathcal{R}|-1}{l-\overline{\mathcal{R}}-1}]$, based on which Algorithm 2 is applied to enable each user to decode its demanded l-subfiles in $\mathcal{W}^l$. Therefore, each user can decode all the l-subfiles it is demanding.

B. Achievable rate

The following theorem presents the delivery rate achieved by the proposed coded caching and delivery scheme for any demand combination, for a given cache allocation vector $\mathbf{P}$.

Theorem 1. *For the caching system described in Section II, given a cache allocation vector* $\mathbf{P}$*, the following delivery rate is achievable*

$$R_{CACC}(\mathbf{P}) = \sum_{l=1}^{N} R_l(t_l), \tag{9}$$

where $t_l = p_l K$*, and for* $t_l \in [0 : K]$,

$$R_l(t_l) = \min\{\alpha_l(t_l), m_l(t_l)\}, \tag{10}$$

and

$$\alpha_l(t_l) \triangleq \sum_{s=\max\{l-K,0\}}^{\max\{\min\{l-1,N-K\},0\}} \binom{N-K}{s}\binom{\min\{N,K\}-1}{l-s-1} \cdot \left[\binom{K}{t_l+1} - \binom{\max\{K - \lceil \frac{\min\{N,K\}}{l-s} \rceil - 1, 0\}}{t_l+1}\right] \frac{F_l}{F\binom{K}{t_l}}, \tag{11}$$

$$m_l(t_l) \triangleq \left(\binom{N}{l} - \binom{\min\{N-K,0\}}{l}\right)(F_l - t_l F_l/K)/F. \tag{12}$$

For $t_l \notin [0 : K]$*,* $R_l(t_l)$ *is given by the lower convex envelop of the above achievable points.*

Proof. We show that $R_l(t_l)$ given above is achievable $t_l \in [0 : K]$. The lower convex envelop of these integer points can then be achieved by memory sharing. For the case $N \leq K$, recall that the requested l-subfiles are delivered in $\binom{N-1}{l-1}$ steps based on $\mathbf{C}_j$, $j = [\binom{N-1}{l-1}]$, derived by Algorithm 1. In each step, the server sends at most $\lceil N/l \rceil + 1$ l-subfiles. Therefore, for any demand combination, the number of distinct d_k^j based on $\mathbf{C}_j$, i.e., $A_j \leq \lceil N/l \rceil + 1$, $n = 1, ..., \binom{N-1}{l-1}$. Similar to the delivery scheme proposed in [13], by the CODED DELIVERY procedure of Algorithm 2, the server broadcasts binary sums that help at least one user in $\mathcal{U}_j$ based on $\mathbf{C}_j$. The total number of such subsets of $t_1 + 1$ users that contain at least one user in $\mathcal{U}_j$ is given by $\binom{K}{t_l+1} - \binom{\max\{K-\lceil N/l \rceil-1,0\}}{t_l+1}$. Hence, given $t_l \in \{1, ..., K-1\}$, the total number of bits sent by CODED DELIVERY procedure of Algorithm 2 for the delivery of the l-subfiles is bounded by (normalized by F):

$$R_l(t_l) \leq \binom{N-1}{l-1} \left[\binom{K}{t_l+1} - \binom{\max\{K - \lceil N/l \rceil - 1, 0\}}{t_l+1} \right] \frac{F_l}{F\binom{K}{t_l}} \tag{13}$$

The right hand side (RHS) of 13 is equal to (11) for $N \leq K$. Since each user caches $t_l F_l / K$ bits of each l-subfile, according to [5, Appendix A], the number of bits sent by the RANDOM DELIVERY procedure is bounded by

$$R_l(t_l) \leq \binom{N}{l} (F_l - t_l F_l / K)/F. \tag{14}$$

The RHS equals to (12) for $N \leq K$. Hence, for $t_l \in \{1, ..., K-1\}$, we have proven $R_l(t_l)$ given in (10) is achievable for $N \leq K$.

We then focus on the case where $N > K$. For any $\max\{l-K, 0\} \leq s \leq \min\{l-1, N-K\}$, there are a total of $\binom{N-K}{s}$ subsets $\overline{\mathcal{R}}$ such that $\overline{\mathcal{R}} \in [N] \setminus \mathcal{R}$, $|\overline{\mathcal{R}}| = s$. Following the similar analysis for the case where $N \leq K$, given $\mathcal{R}$ containing all the demanded files such that $|\mathcal{R}| = K$, and any $\overline{\mathcal{R}}$ such that $\overline{\mathcal{R}} \in [N] \setminus \mathcal{R}$, $|\overline{\mathcal{R}}| = s$, requested l-subfiles in $\mathcal{W}^l = \{\overline{W}_{\mathcal{S} \cup \overline{\mathcal{R}}} : \mathcal{S} \subset \mathcal{R}, |\mathcal{S}| = l - |\overline{\mathcal{R}}|\}$ are sent in $\binom{K-1}{l-s-1}$ step. At each step, there are at most $\lceil \frac{N}{l-s} \rceil + 1$ l-subfiles to be sent. Therefore, with similar arguments, the total number of bits sent by CODED DELIVERY procedure of Algorithm 2 in each step is bounded by $\binom{K-1}{l-s-1} \left(\binom{K}{t_l+1} - \binom{\max\{K-\lceil \frac{N}{l-s} \rceil-1,0\}}{t_l+1} \right) \frac{F_l}{F\binom{K}{t_l}}$, while the number of bits send by the RANDOM DELIVERY procedure is bounded by $\binom{K}{l-s}(F_l - t_l F_l/K)/F$. By summing over all $\binom{N-K}{s}$ subsets $\overline{\mathcal{R}}$ for each $s \in [\max\{l-K, 0\} : \min\{l-1, N-K\}]$, we have

$$R_l(t_l) \leq \sum_{s=\max\{l-K,0\}}^{\min\{l-1,N-K\}} \binom{N-K}{s} \binom{K-1}{l-s-1} \cdot \left(\binom{K}{t_l+1} - \binom{\max\{K - \lceil \frac{N}{l-s} \rceil - 1, 0\}}{t_l+1} \right) \frac{F_l}{F\binom{K}{t_l}}, \tag{15}$$

and,

$$R_l(t_l) \leq \sum_{s=\max\{l-K,0\}}^{\min\{l-1,N-K\}} \binom{N-K}{s} \binom{K}{l-s} (F_l - t_l F_l/K)/F, \tag{16}$$

by which, we have proven the correctness of (10) for the case $N > K$. □

Remark 1. *At each step of sending l-subfiles in $\mathcal{W}^l = \{\overline{W}_{\mathcal{S} \cup \overline{\mathcal{R}}} : \mathcal{S} \subset \mathcal{R}, |\mathcal{S}| = l - |\overline{\mathcal{R}}|\}$, there are sometimes $\lceil \frac{N}{l-s} \rceil + 1$ and sometimes $\lceil \frac{N}{l-s} \rceil$ distinct demands, while when N is a multiple of $l-s$, there are always $\frac{N}{l-s}$ distinct demands ($\mathcal{R} = [N]$, $\overline{\mathcal{R}} = \emptyset$, $s = 0$, for the case $N \leq K$). To obtain a closed-form expression for the achievable delivery rate, we simply assume $\lceil \frac{N}{l-s} \rceil + 1$ distinct demands at each step. Note that, the more the number of distinct demands at each step, the larger the delivery rate. Therefore $R_{CACC}(\mathbf{P})$ in (9) is an upper bound on the actual achievable delivery rate of CACC.*

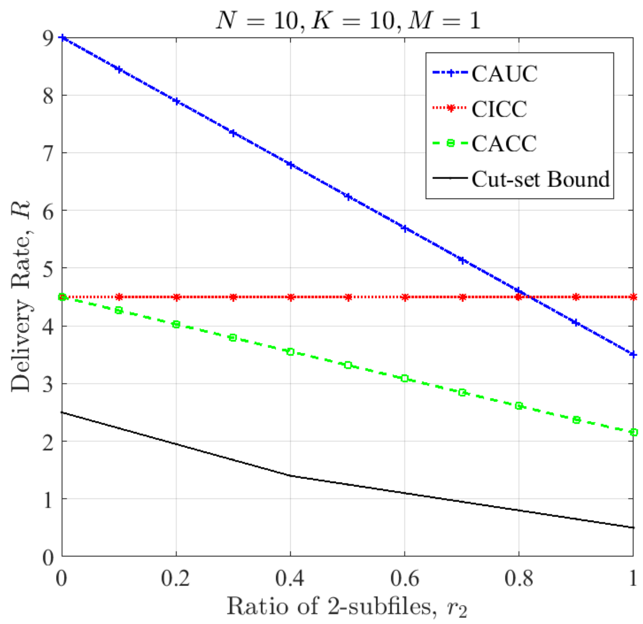

Fig. 1. Delivery rate (R) vs. ratio of 2-subfiles (r_2),$r_3 = \cdots = r_{10} = 0$.

C. Allocation of Cache Capacity

We can further optimize the cache content distribution $\mathbf{P}$ by solving:

$$\min \quad R_{CACC}(\mathbf{P}) \tag{17a}$$

$$\text{such that} \quad \sum_{l=1}^{N} \binom{N}{l} p_l F_l \leq MF, \tag{17b}$$

where the objective is to minimize the achievable delivery rate under the cache capacity constraint. The problem in (17) can be solved numerically.

V. Lower Bound

In this section, we present a lower bound derived using cut-set arguments.

Theorem 2. *(Cut-set Bound) For the caching problem described in Section II, the optimal achievable delivery rate is lower bounded by*

$$R^*(M) \geq \max_{p \in [1:\min\{N,K\}]} \sum_{s=0}^{N-p\lfloor N/p \rfloor} \sum_{l=1}^{p\lfloor N/p \rfloor} \binom{N - p\lfloor N/p \rfloor}{s} \cdot \binom{p\lfloor N/p \rfloor}{l} \frac{F_{l+s}}{\lfloor N/p \rfloor} - \frac{pM}{\lfloor N/p \rfloor}. \tag{18a}$$

Proof. The proof will be provided in a longer version of the paper. □

VI. Numerical results

In this section, we numerically compare the delivery rates of the proposed correlation-aware caching schemes CAUC and CACC with the lower bound and the state-of-the-art coded caching scheme from [13], which does not take the content

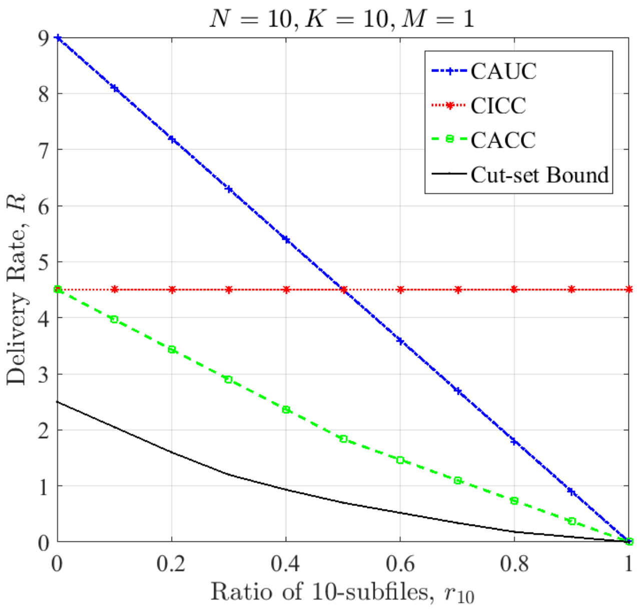

Fig. 2. Delivery rate (R) vs. ratio of 10-subfiles (r_{10}), $r_2 = \cdots = r_9 = 0$.

correlations into account. We refer to the later scheme as the correlation-ignorant coded caching scheme (CICC).

We consider $N = 10$ files and $k = 10$ users. Each user is equipped with a cache of size F bits, i.e., $M = 1$. We denote by r_l the ratio of l-subfiles among each file, i.e., $r_l \triangleq \binom{N-1}{l-1} F_l/F$. Note that, we have $\sum_{l=1}^{K} r_l = 1$. In Fig. 1, we assume that the files have only pairwise correlations, that is, $r_3 = \cdots = r_{10} = 0$, and we plot the delivery rate as a function of r_2. Meanwhile, in Fig. 2, we assume that each file consists of a private part, i.e., 1-subfile, and a common subfile that is shared by all the files in the library, i.e., 10-subfile, i.e., $r_2 = \cdots = r_9 = 0$. We plot the delivery rate as a function of r_{10}.

We observe in both figures that the delivery rate achieved by the correlation-ignorant scheme, CICC, remains the same no matter how high the ratio of common subfiles, while the delivery rates of the correlation-aware schemes, CAUC and CACC, decrease as the ratio of the common subfiles increases. Obviously, CACC achieves a lower delivery rate than both CAUC and CICC, since it benefits both from incorporating the correlations among the files as well as coded multicasting. When the ratio of the common subfiles is sufficiently large, even without coded multicasting CAUC achieves a lower delivery rate than CICC. It can also be observed that the delivery rates of correlation-aware schemes decrease faster with the percentage of common subfiles in Fig. 2 than in Fig. 1. That is because the gain from exploiting correlation is more pronounced as the common parts are shared among more files. While there is a gap between the cut-set lower bound and the achievable delivery rate, we note that the gap is smaller in Fig. 2, where the level of commonness is higher.

VII. Conclusions

We have studied coded caching taking into account the available correlations among the files in the library. To capture arbitrary correlations, we assume that each file consists of a number of subfiles, each of which is shared by a different subset of files in the library, and the number of files that share a certain subfile is defined as its level of commonness. We proposed both a correlation-aware uncoded caching scheme, the optimal placement of which is proven to be caching the subfiles with the highest levels of commonness, and a correlation-aware coded caching scheme (CACC), the placement of which is optimized in terms of the achievable delivery rate. The proposed CACC scheme, or even the uncoded caching scheme when the correlation among files is strong enough, is shown to significantly outperform the best known achievable delivery rate by correlation-unaware solution in the literature.

References

[1] M. Gregori, J. Gomez-Vilardebo, J. Matamoros, and D. Gündüz, “Wireless content caching for small cell and D2D networks,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222–1234, Mar 2016.

[2] E. Bastug, M. Bennis, and M. Debbah, “Living on the edge: The role of proactive caching in 5g wireless networks,” *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, Aug 2014.

[3] S. O. Somuyiwa, A. György, and D. Gündüz, “Improved policy representation and policy search for proactive content caching in wireless networks,” in *Proc. IEEE Int'l Symp. on Modeling and Opt. in Mobile, Ad Hoc, and Wireless Netw. (WiOpt)*, Paris, France, May 2017.

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

[5] ——, “Decentralized caching attains order optimal memory-rate tradeoff,” *IEEE/ACM Trans. Netw*, vol. 23, no. 4, pp. 1029–1040, Apr. 2014.

[6] M. Mohammadi Amiri, Q. Yang, and D. Gündüz, “Coded caching for a large number of users,” in *Proc. IEEE Inform. Theory Workshop (ITW)*, Cambridge, UK, Sep. 2016, pp. 171–175.

[7] M. Mohammadi Amiri and D. Gündüz, “Fundamental limits of coded caching: Improved delivery rate-cache capacity trade-off,” *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 806–815, Feb. 2017.

[8] J. Gomez-Vilardebo, “Fundamental limits of caching: Improved bounds with coded prefetching,” *arXiv:1612.09071v2 [cs.IT]*, Jan. 2017.

[9] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, “Order-optimal rate of caching and coded multicasting with random demands,” *arXiv: 1502.03124v1 [cs.IT]*, Feb. 2015.

[10] U. Niesen and M. A. Maddah-Ali, “Coded caching with nonuniform demands,” *IEEE Trans. Inform. Theory*, vol. 63, no. 2, pp. 1146–1158, Feb. 2017.

[11] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, “Online coded caching,” in *Proc. IEEE Int'l Conf. Commun. (ICC)*, Sydney, Australia, Jun. 2014, pp. 1878–1883.

[12] Q. Yang and D. Gündüz, “Coded caching and content delivery with heterogeneous distortion requirements,” *arXiv:1608.05660v1 [cs.IT]*, Aug. 2016.

[13] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory tradeoff for caching with uncoded prefetching,” in *Proc. IEEE Int'l Symp. on Inform. Theory (ISIT)*, Aachen, Germany, Jun 2017, pp. 1613–1617.

[14] Q. Yang, M. Mohammadi Amiri, and D. Gündüz, “Audience retention rate aware coded video caching,” in *Proc. IEEE Int'l Conf. on Commun. Workshop (ICC Workshop)*, Paris, France, Jun. 2017, pp. 1189–1194.

[15] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, “Correlation-aware distributed caching and coded delivery,” in *Proc. IEEE Inform. Theory Workshop (ITW)*, Cambridge, UK, Sep. 2016, pp. 166–170.

[16] R. Timo, S. S. Bidokhti, M. Wigger, and B. C. Geiger, “A rate-distortion approach to caching,” *arXiv:1610.07304v1 [cs.IT]*, Oct. 2016.

[17] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, “Rate-memory tradeoff for the two-user broadcast caching network with correlated sources,” *arXiv:1705.04616v1 [cs.IT]*, May 2017.