Decentralized Coded Caching Without File Splitting

Seyed Ali Saberali, Member, IEEE, Lutz Lampe, Senior Member, IEEE,
and Ian Blake, Fellow, IEEE

Abstract

Coded caching is an effective technique to reduce the redundant traffic in wireless networks. The existing coded caching schemes require the splitting of files into a possibly large number of subfiles, i.e., they perform coded subfile caching. Keeping the files intact during the caching process would actually be appealing, broadly speaking because of its simpler implementation. However, little is known about the effectiveness of this coded file caching in reducing the data delivery rate. In this paper, we propose such a file caching scheme which uses a decentralized algorithm for content placement and either an online clique cover or matching algorithm for the delivery of missing data. We derive approximations for the expected delivery rate of both schemes using the differential equations method, and show them to be tight through concentration analysis and computer simulations. Our numerical results demonstrate that the proposed coded file caching is significantly more effective than uncoded caching in reducing the delivery rate. We furthermore show the additional improvement in the performance of the proposed scheme when its application is extended to subfile caching with a small number of subfiles.\footnote{This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.}

Index Terms

5G communications, clique cover algorithm, coded file caching, index coding, traffic offloading.
I. INTRODUCTION

Caching of popular content at the wireless edge is a promising technique to offload redundant traffic from the backhaul communication links of the next generation wireless networks [1]–[5]. An integral part of the 5G cellular systems is the dense deployment of small-cells within macrocells to increase the spectral efficiency [5]–[7]. By having each small base-station equipped with a large memory storage, multiple caching nodes co-exist within each macrocell. This provides the opportunity to use network coding to further decrease the backhaul traffic of the macrocell over the conventional caching systems [8]. In particular, the cached content can be used as side information to decode multicast messages that simultaneously deliver the missing content to multiple caches. The design of content placement in the caches and construction of the corresponding coded multicast messages are two elements that constitute the coded caching problem [8]–[10].

A. Preliminaries

Coded caching is closely related to the index coding with side information problem [11], [12]. In both cases, there is a server that transmits data to a set of $K$ caching clients over a broadcast channel. The server is aware of the clients’ cached content. Each client wants certain blocks of data, yet some of these blocks might be missing from its cache. The objective is to transmit the minimum amount of supplemental data over the broadcast channel such that all clients can derive the data blocks that they requested [9], [12]. In the coded caching literature, the amount of supplemental data transmitted is referred to as delivery rate [9]. The main factor that differentiates the two problems is that in index coding, the cached content is usually given and the focus is on the design of the server messages. However, in coded caching, the placement of the content in the caches can also be designed.

An information theoretic formulation of coded caching was developed in [9]. The authors proposed a coded caching scheme which uses a centralized content placement algorithm and a set of coded delivery messages. The worst-case delivery rate of this scheme was shown to be $R = \frac{K(1-M/N)}{1+KM/N}F$ packets, where $M$ and $N$ are the number of files that each cache can store and the total number of files that are predicted to be popular, respectively. Parameter $F$ is the
number of packets per file. A packet can be a single bit or a chunk of bits of a file, but they are all of the same length and cannot be broken into smaller parts. The placement of [9] splits each file into a fixed number of subfiles. A subfile is a set of packets of a file. We refer to any coded caching that breaks files into subfiles as Coded Subfile Caching (CSC). Notice that \(1+KM/N\) in the rate expression is the multiplicative gain due to coding. In [10], the authors proposed a decentralized CSC which allows every cache to store the content independent from the content of the other caches. The proposed scheme preserves most of the coding gain of the centralized scheme of [9] and has a worst-case delivery rate of \(R=(N/M-1)(1-(1-M/N)^K)\) packets in the asymptotic regime of \(F \to +\infty\). Since decentralized caching does not require any central coordination among the caches for placement, it is the preferred caching framework for the next generation wireless systems. As a result, the scheme of [10] has served as the building block of several other coded caching methods [13]–[17]. Both [9] and [10] considered the worst-case delivery rates which correspond to the demand vectors where all caches’ requests are distinct. In a recent work [18], the minimum average delivery rates of both centralized and decentralized coded cachings with uncoded prefetching were characterized.

**B. Motivation and Related Works**

In this paper, we consider the problem of decentralized Coded File Caching (CFC), where files are kept intact and are not broken into subfiles. Our first motivation to analyze CFC is the large size requirement of most of the existing decentralized subfile cachings. In particular, [19] has developed a finite-length analysis of the decentralized coded caching of [10] and showed that it has a multiplicative coding gain of at most 2 even when \(F\) is exponential in the asymptotic gain \(KM/N\). In [19], the authors have also proposed a new CSC scheme that requires a file size of \(\Theta([N/M]^{g+1}(\log(N/M))^{g+2}(2e)^g)\) to achieve a rate of \(4K/(3(g+1))\).

Another important limitation of the existing coded caching schemes is the large number of subfiles required to obtain the theoretically promised delivery rates. For instance, [10] requires \(2^K\) subfiles per file, and the different missing subfiles of a file requested by each cache are embedded into \(2^{K-1}\) different server messages [10]. This exponential growth in \(K\) has adverse consequences on the practical implementation of the system and its performance. In particular,
a larger number of subfiles imposes large storage and communication overheads on the network. As is pointed out in [4], during placement, the caches must also store the identity of the subfiles and the packets that are included in each subfile, as a result of the randomized placement algorithms used. This leads to a storage overhead. Similarly, during delivery, the server needs to transmit the identities of the subfiles that are embedded in each message which leads to communications overhead. Neither of these overheads are accounted for in the existing coded caching rate expressions, while they are non-negligible when the number of subfiles is relatively large. For centralized coded caching, [20] has proposed a scheme that requires a linear number of subfiles in $K$ and yields a near-optimal delivery rate. As is pointed out in [20], the construction of the proposed scheme is not directly applicable to the real caching systems as it requires $K$ to be quite large. However, it is interesting to see that a small number of subfiles can in theory result in a near-optimal performance. For decentralized caching, reference [21] has investigated the subfile caching problem in the finite packet per file regime and has proposed a greedy randomized algorithm, GRASP, to preserve the coding gain in this regime. Based on computer simulations, the authors show that with a relatively small number of subfiles, a considerable portion of the asymptotic coding gain of infinite file size regime can be recovered. This observation further motivates a closer look at the file caching problem and its performance analysis. Finally, since the conventional uncoded caching systems do not break files into large numbers of subfiles, it is easier for the existing caching systems to deploy coded caching schemes that require one or a small number of subfiles per file.

C. Contributions

Although CSC has been well investigated, the analysis of CFC is missing from the literature. Based on the motivations we discussed earlier, it is worthwhile to explore CFC for its potential in reducing the delivery rate. In this paper, we investigate the decentralized CFC problem by proposing new placement and delivery algorithms. The placement algorithm is decentralized and does not require any coordination among the caches. The first proposed delivery algorithm is in essence an online implementation of the clique cover procedure which is applied to certain side information graphs. This algorithm is designed for the general settings of CFC. The second
proposed delivery algorithm adapts the online matching procedure of [22] and is suitable for CFC in the small cache size regime.

The main contribution of this paper is the analysis of the expected delivery rates of the proposed algorithms for file caching. In particular, we provide a modeling of the dynamics of these algorithms through systems of differential equations. The resulting system can be solved to provide a tight approximation to the expected delivery rate of CFC with the proposed algorithms. We provide a concentration analysis for the derived approximations and further demonstrate their tightness by computer simulations.

Our results show that CFC is significantly more effective than uncoded caching in reducing the delivery rate, despite its simple implementation and structurally constrained design. We present a discussion on the extension of the proposed placement and delivery algorithms to subfile caching with an arbitrary number of subfiles per file. We show that a considerable portion of the gain of the existing CSC methods can be achieved by using a small number of subfiles per file, which is consistent with the observations made in [21]. Hence, our proposed method provides a means to tradeoff the delivery rate with the complexity caused by the use of a larger number of subfiles.

The remainder of this paper is organized as follows. We present our proposed CFC method in Section II. Statistical analysis of the delivery rates and concentration results are provided in Section III. We present numerical examples and simulation results in Section IV. In Section V, we discuss the generalization of the proposed placement and delivery to CSC. We conclude the paper in Section VI.

II. PLACEMENT AND DELIVERY ALGORITHMS

A. Problem Setup for CFC

Similar to [10], we consider a network with a central server and \( K \) caches. The central server is connected to caches through an error-free broadcast channel. A library of \( N \geq K \) popular files is given, where the popularity distribution is uniform over the files. Every file is of the same length of \( F \) packets. The storage capacity of every cache is \( M \) files. For the CFC problem which we consider here, each cache can only store the entirety of a file and partial storage of
a file is not allowed during content placement. This is the main element that differentiates the problem setups of CFC and CSC.

The delivery phase takes place after the placement, where at each time instant, every cache \( k \) reveals one request for a file \( d_k \) based on the request of its local users. We define the demand vector \( d = [d_1, \ldots, d_K] \) as the vector that consists of the requests made by all the caches at the current time instant. User requests are assumed to be random and independent. If the requested file is available in the corresponding cache, the user is served locally. Otherwise, the request is forwarded to the server. The server is informed of the forwarded requests and transmits a signal of size \( R(d; P, A_D) \) files over the broadcast link. The quantity \( R(d; P, A_D) \) is the delivery rate for the demand vector \( d \), given a specific placement of files \( P \) and a delivery algorithm \( A_D \).

Placement \( P \) is fixed for all the demand vectors that arrive during the delivery phase. Every cache that has forwarded its request to the server must be able to decode the broadcasted signal for the file it requested. We are interested in minimizing \( \mathbb{E}(R(d; P, A_D)) \), where the expectation is over the randomness in vector \( d \), and possible randomness in the delivery algorithm \( A_D \) and in the placement algorithm that determines \( P \).

In the following, we propose algorithms for the placement and delivery of CFC.
Algorithm 1 Decentralized File Placement

Require: Library of $N$ files
1: for $k = 1, \ldots, K$ do
2:   Cache $k$ stores $M$ out of $N$ files from the library uniformly at random
3: end for

B. Placement

Algorithm 1 shows the randomized procedure that we propose for content placement. Here, each cache stores $M$ files from the library uniformly at random. The placement is decentralized as it does not require any central coordination among the caches. In Algorithm 1, all packets of $M$ files are entirely cached. Notice that each cache stores any particular file with probability $q = M/N$, independent from the other caches. However, the storage of the different files in a given cache are statistically dependent, as the total number of cached files must be less than $M$. Notice that Algorithm 1 is different from the decentralized subfile placement of [10], as in the latter, an $M/N$-th portion of the packets of every file in the library is cached and the packets of each file are cached independently from the packets of the other files.

C. Delivery

For the proposed delivery, the server forms a certain information graph for every demand vector that arrives, and delivers the requests by applying a message construction procedure to that graph. The graphs that we use are defined in the following.

Definition 1. For a given placement and demand vector, the side information digraph $\mathcal{D}$ is a directed graph on $K$ vertices. Each vertex corresponds to a cache. A vertex has a loop if and only if the file requested by the cache is available in its storage. There is a directed edge from $v$ to $w$ if and only if the file requested by cache $w$ is in cache $v$.

Digraph $\mathcal{D}$ was first introduced in the index coding literature and was used to characterize the minimum length of linear index codes [12]. This was done through computing the minrank of $\mathcal{D}$ [12, Theorem 1], which is an NP-hard problem.
Definition 2. For a given placement and demand vector, the side information graph $G$ is defined on the same set of vertices as $D$. A vertex in $G$ has a loop if and only if it has a loop in $D$. There is an (undirected) edge between $v$ and $w$ if and only if both edges from $v$ to $w$ and from $w$ to $v$ are present in $D$.

Since the requests and the content of every cache are random and independent from the requests and the content of the other caches, and since the file popularity distribution is uniform, $D$ and $G$ can be modeled as random graphs as follows.

Remark 1. In digraph $D$, every directed edge or loop is independently present with probability $q$. In graph $G$, every loop is present with probability $q$, and every edge between two distinct vertices is present with probability $q^2$, independent from the other edges and loops.

We now propose two algorithms that make use of graph $G$ for delivery.

a) Online Clique Cover Delivery Algorithm: For delivery, the server constructs multicast messages with the property that each cache can derive its desired file from such messages using the side information that it has available.

Now, consider a set of unlooped vertices in graph $G$ that form a clique. Each vertex in that set has the file requested by the other caches available, but its own requested file is missing from its local storage. As a result, if the server transmits a message that is the XOR of the files requested by the caches in that clique, the message is decodeable by all such caches for their desired files [19, Section II.A]. Hence, to deliver the requests of every cache and minimize the delivery rate, it is of interest to cover the set of unlooped vertices of $G$ with the minimum possible number of cliques and send one server message per clique to complete the delivery. Notice that the looped vertices of $G$ do not need to be covered as the files requested by such caches are available to them locally.

Finding the minimum clique cover is NP-hard [23]. In this section, we propose a polynomial time online clique cover procedure for the construction of the delivery messages of CFC whose

\[^2\text{For an undirected simple graph (no loops and no multiple edges), a clique is a subset of vertices where every pair of vertices are adjacent.}\]
Algorithm 2 Online Clique Cover Delivery

Require: \( \mathcal{G} \), vertex labels \( v_1, \ldots, v_K \)

\hspace{1em} // Form Cliques
1: \( C_i \leftarrow \emptyset \), \( i = 1, \ldots, K \) \hspace{1em} // initialize sets of cliques
2: for \( t = 1, \ldots, K \) do
3: if \( v_t \) has no loop then
4: \( \hat{c} = \arg \max_{c \in \bigcup_{i=1}^{K} C_i} \{|c| \mid (c, v_t) \text{ is a clique} \} \)
5: if \( \hat{c} = \emptyset \) then
6: \( C_1 \leftarrow C_1 \cup \{v_t\} \)
7: else
8: \( j = |\hat{c}| \)
9: \( C_{j+1} \leftarrow C_{j+1} \cup \{(\hat{c}, v_t)\} \)
10: \( C_j \leftarrow C_j \setminus \{\hat{c}\} \)
11: end if
12: end if
13: end for

\hspace{1em} // Transmission of Messages
14: for \( i = 1, \ldots, K \) do
15: for \( c \in C_i \) do
16: Transmit \( \oplus_{k \in c} X_k \)
17: end for
18: end for

Performance analysis is mathematically tractable based on the model in Remark 1. The proposed clique cover procedure is presented in Algorithm 2.

Notation 1. In Algorithm 2, symbol \( c \) represents a vector of vertices and its size is denoted by \( |c| \). Notation \( (c, v) \) is used for appending vertex \( v \) to vector \( c \). The content of the file requested by cache \( k \) is denoted by \( X_k \) and \( \oplus \) represents the bitwise XOR operation.

In Algorithm 2, a vertex \( v_t \) arrives at iteration (time) \( t \in \{1, \cdots, K\} \). If \( v_t \) is looped, we proceed to the next iteration. Otherwise, we check if there is any previously formed clique of size \( s \in \{1, \ldots, t - 1\} \) that together with \( v_t \) forms a clique of size \( s + 1 \). We call such a clique a suitable clique for \( v_t \). If suitable cliques for \( v_t \) exist, we add \( v_t \) to the largest of them to form a new clique. Finding the largest suitable clique is done in line 4. If no suitable clique exists, \( v_t \) forms a clique of size 1. Notice that the cliques formed by Algorithm 2 are disjoint and cover the set of unlooped vertices of \( \mathcal{G} \). After the clique cover procedure completes, the server sends a coded message corresponding to each clique.
The use of Algorithms 1 and 2 for placement and delivery leads to a caching scheme that we call CFC with online Clique Cover Delivery (CFC-CCD).

Remark 2. Algorithm 2 joins $v_t$ to the largest suitable clique at time $t$. The rationale for this choice is as follows. Let $Y_i(t)$ represent the number of cliques of size $i$ right before time $t$, where $i \in \{1, \cdots, K\}$. Also, let $\mathcal{N}_t$ be the event that there exists no suitable clique for $v_t$ at time $t$. Then,

$$
\mathbb{P}(\mathcal{N}_t) = \prod_{i=1}^{K} (1 - q^{2i})^{Y_i(t)}.
$$

Given $Y_i(t)$, and with the knowledge that $v_t$ has joined a clique of size $s$ at time $t$, we get $Y_i(t + 1) = Y_i(t), i \not\in \{s, s+1\}$, $Y_s(t + 1) = Y_s(t) - 1$ and $Y_{s+1}(t + 1) = Y_{s}(t) + 1$. Then, it is easy to show that

$$
\mathbb{P}(\mathcal{N}_{t+1}|v_t \text{ joined a clique of size } s \text{ at time } t, Y(t)) = \frac{1 - q^{2(s+1)}}{1 - q^{2s}} \prod_{i=1}^{K} (1 - q^{2i})^{Y_i(t)}, \quad (1)
$$

where $Y(t) = (Y_1(t), \cdots, Y_K(t))$. In order to minimize the expected rate, one has to make (1) small. Now, assume that at time $t$, multiple suitable cliques existed for $v_t$. It is straightforward to check that (1) would take its smallest value if the suitable clique with the largest size $s$ was chosen at time $t$.

Remark 3. Algorithm 2 is online in the sense that it does not have the entire graph available from the beginning, and one vertex becomes available at each iteration. This results in a property that is essential for the performance analysis of the delivery algorithm. In particular, when vertex $v_t$ arrives, the behavior of the algorithm up to time $t - 1$ does not provide any information about the connectivity of vertex $v_t$. In other words, $v_t$ is connected to any of the previously arrived vertices with probability $q^2$. In contrast, consider a general offline greedy algorithm which traverses the vertices one by one to form cliques similar to Algorithm 2, but with the difference that instead of \{$v_1, \cdots, v_{t-1}\}, all vertices were available for $v_t$ to possibly join to for every $t$. Then, at iteration $t$, the probability of $v_t$ to be connected to a previously traversed vertex $v_s \in \{v_1, \cdots, v_{t-1}\}$ is affected by the results up to time $t$. For instance, the observation that none of the previously traversed vertices are joined to $v_t$ to form a clique before time $t$ implies a higher posterior
Algorithm 3: Online Matching Delivery

Require: \( G \), vertex labels \( v_1, \ldots, v_K \)

// Transmission of coded messages
1: \( Q \leftarrow \emptyset \) // set of matched or looped vertices
2: for \( t = 1, \ldots, K \) do
3: \( G_t \leftarrow \) subgraph induced by \( G \) on vertices \( \{ v_1, \ldots, v_t \} \setminus Q \)
4: if \( v_t \) has a loop then
5: \( Q \leftarrow Q \cup \{ v_t \} \)
6: else if \( v_t \) has a neighbor in \( G_t \) then
7: Match \( v_t \) to a random neighbor \( v_s \in G_t \)
8: Transmit \( X_{v_t} \oplus X_{v_s} \)
9: \( Q \leftarrow Q \cup \{ v_t, v_s \} \)
10: end if
11: end for

// Transmission of Uncoded Messages
12: for \( v \in \{ v_1, \ldots, v_K \} \setminus Q \) do
13: Transmit \( X_v \)
14: end for

Probability of the absence of an edge between \( v_t \) and \( v_s \), compared to probability \( 1 - q^2 \) implied by the prior graph model.

Remark 4. In Algorithm 2 each cache only needs to listen to one server message to decode the file it requested, as the entirety of the file is embedded in only one message. This is in contrast to the CSC of [10] which requires each cache to listen to \( 2^{K-1} \) out of the \( 2^K \) server messages to derive its requested content.

Finally we note that Algorithm 2 has a worst-case complexity of \( O(K^2) \). This is because there are at most \( K \) unlooped vertices in \( G \), and in each iteration of the algorithm, the adjacency of \( v_t \) with at most \( t - 1 \) vertices must be checked for finding the largest suitable clique.

b) Online Matching Delivery Algorithm: We now propose our second delivery algorithm, which is applicable to the small cache size regime, i.e., when \( q \) is small. In this regime, the probability of having large cliques is small. Hence, one can restrict the size of the cliques in the clique cover procedure to reduce the complexity without considerably affecting the delivery rate. As an extreme case, Algorithm 3 shows an online greedy matching algorithm adapted from [22] Algorithm 10], which restricts cliques to those of sizes 1 and 2. Notice that here graph \( G \)
can have loops which is different from the graph model in \cite[Algorithm 10]{22}. In Algorithm 3 if $v_t$ is looped, we remove it from the graph and proceed to the next iteration. Otherwise, we try to match it to a previously arrived unmatched vertex. If a match found, we remove both vertices from the graph and proceed to the next iteration. Otherwise, we leave $v_t$ for possible matching in the next iterations.

The use of Algorithms 1 and 3 for placement and delivery leads to a method that we call CFC with online Matching Delivery (CFC-MD).

III. PERFORMANCE ANALYSIS

In this section, we analyze the expected delivery rates of the proposed file caching schemes through a modeling of the dynamics of Algorithms 2 and 3 by differential equations. We use Wormald’s theorem \cite[Theorem 5.1]{24} for our analysis. Wormald’s theorem and its application for the analysis of discrete random processes are discussed in Appendix A.

We begin with the analysis of CFC-MD as it involves a simpler delivery algorithm.

**Theorem 1.** With probability $1 - O\left(\frac{1}{\lambda} e^{-K\lambda^3}\right)$, the delivery rate of CFC-MD is

$$R_{m}(q) = \frac{1}{2} \left[ K(1 - q) - \frac{\log(2 - (1 - q^2)^K(1-q))}{\log(1 - q^2)} \right] + O(\lambda K)$$

for any $\lambda > 0$.

**Proof:** To prove Theorem 1, we use Wormald’s theorem following an approach inspired by the approach in \cite[Section 3.4.1]{22}. Let $L$ be the set of looped vertices of $G$. Also, let $M$ and $U$ represent the sets of unlooped vertices that are matched and remain unmatched by Algorithm 3 respectively. Since a looped vertex will not be matched by Algorithm 3 these three sets are disjoint and partition the vertices of $G$.

For a given side information graph $G$, the delivery rate of Algorithm 3 is

$$R_{m}^{G} = \frac{|M|}{2} + |U| = \frac{K - |U| - |L|}{2} + |U| = \frac{1}{2}(K - |L| + |U|),$$

(2)

where we used the fact that one coded message is transmitted per each pair of matched vertices and one uncoded message is transmitted for each unlooped and unmatched vertex.
Based on (2), to analyze the statistical behavior of the delivery rate one needs to analyze \(|U|\) and \(|L|\). We do this using Theorem 3 in Appendix A. For that, we index the online matching process of Algorithm 3 by \(K\), which is the number of vertices of \(G\). Let us define the two variables \(L(t)\) and \(U(t)\) on the process. Variable \(L(t)\) is the number of looped vertices in \(\{v_1, \ldots, v_{t-1}\}\). Variable \(U(t)\) denotes the number of unlooped vertices in \(\{v_1, \ldots, v_{t-1}\}\) that are not matched by Algorithm 3 in the first \(t-1\) iterations. Notice that since \(U(t) < K\) and \(L(t) < K\), we set \(c_0 = 1\) in Theorem 3.

In the following, we verify that the three conditions of Theorem 3 are satisfied for the defined variables. Both \(L(t)\) and \(U(t)\) satisfy the boundedness condition with \(\beta = 1\) and \(\gamma = 0\), as \(|L(t+1) - L(t)| \leq 1\) and \(|U(t+1) - U(t)| \leq 1\) always hold. For the trend hypothesis, we have

\[
\mathbb{E}(U(t+1) - U(t)|\mathcal{H}_t) = 0 \times \mathbb{P}(v_t \text{ looped})
- 1 \times \mathbb{P}(v_t \text{ unlooped and matches at time } t)
+ 1 \times \mathbb{P}(v_t \text{ unlooped and does not match at time } t)
= -(1-q)(1-(1-q^2)U(t)) + (1-q)(1-q^2)U(t)
= (1-q) \left[2(1-q^2)^{U(t)} - 1\right].
\]

and

\[
\mathbb{E}(L(t+1) - L(t)|\mathcal{H}_t) = \mathbb{P}(v_t \text{ looped } ) = q,
\]

where \(\mathcal{H}_t\) is the history of the process up to time \(t\). Since the derived expectations are deterministically true, the trend hypothesis holds with \(\lambda_1 = 0\) and \(f_1(x, z_1, z_2) = (1-q)[2(1-q^2)^{Kz_1} - 1]\) and \(f_2(x, z_1, z_2) = q\), with domain \(D\) defined as \(-\epsilon < x < 1 + \epsilon\), \(-\epsilon < z_1 < 1 + \epsilon\) and \(-\epsilon < z_2 < 1 + \epsilon\), \(\epsilon > 0\). Finally, the Lipschitz hypothesis is also satisfied as \(f_1\) and \(f_2\) are Lipschitz continuous on \(\mathbb{R}^2\). This is because they are differentiable everywhere and have bounded derivatives.

Since the conditions of the theorem are satisfied, the dynamics of \(z_1\) and \(z_2\) can be formulated
by the differential equations
\[
\frac{dz_1(x)}{dx} = (1 - q)\left[2(1 - q^2)^{Kz_1(x)} - 1\right], \\
\frac{dz_2(x)}{dx} = q,
\]
where the initial condition results from \( U(0) = L(0) = 0 \). Notice that the equations derived are decoupled in \( z_1 \) and \( z_2 \), and can be solved independently as
\[
z_1(x) = -\frac{\log(2 - (1 - q^2)^{Kz_1(x)})}{K\log(1 - q^2)}, \\
z_2(x) = qx.
\]
Then, for \( \lambda > \lambda_1 + c_0 K \gamma = 0 \), with probability \( 1 - O\left(\frac{1}{\lambda}e^{-K\lambda^3}\right) \), we have
\[
U(t) = Kz_1(t/K) + O(\lambda K) \\
= -\frac{\log(2 - (1 - q^2)^{(1-q)t})}{\log(1 - q^2)} + O(\lambda K), \\
L(t) = Kz_2(t/K) + O(\lambda K) = qt + O(\lambda K),
\]
uniformly for \( 0 \leq t \leq K \).

By evaluating the derived expressions for \( U(t) \) and \( L(t) \) at \( t = K \), and their respective substitution in (2) for \( |U| \) and \( |L| \), Theorem 1 results.

Following an approach similar to the proof of Theorem 1, we derive concentration results for the delivery rate of CFC-CCD.

**Theorem 2.** With probability \( 1 - O\left(\frac{K}{\lambda}e^{-K\lambda^3}\right) \), the delivery rate of CFC-CCD is
\[
R_{cc}(q) = K \sum_{i=1}^{K} z_i(1; q) + O(\lambda K),
\]
where functions \( z_i(x; q), i = 1, \ldots, K \) are given by the unique solution to the system of differ-
ential equations

\[
\begin{aligned}
\frac{dz}{dx} &= (1-q)[2g_i(z) - g_{i+1}(z) + g_{i-1}(z)]; \\
z_i(0; q) &= 0,
\end{aligned}
\]

where

\[g_i(z) = \prod_{j=i}^{K} (1 - q^{2j})^{Kz_j(x; q)}\]

and \(z = (z_1, \ldots, z_K)\).

Proof: Define \(Y_i(t)\) as the number of cliques of size \(i\) formed by Algorithm 2 up to iteration \(t - 1\), where \(i = 1, \ldots, K\). To analyze the behavior of these variables, we use the version of Wormald’s theorem that is provided in Remark 8 in Appendix A. This is because the number of variables \(Y_i\) depends on \(K\) here.

It is straightforward to show that the boundedness hypothesis of Theorem 3 is satisfied with \(\beta = 1\) and \(\gamma = 0\), as in each iteration of the algorithm, the number of cliques of each size either remains unchanged, or decrements or increments by 1. Furthermore, for the trend hypothesis, we model the expected change of each variable throughout the process by

\[
\mathbb{E}(Y_i(t+1) - Y_i(t) | Y(t)) = 0 \times \mathbb{P}(v_t \text{ looped}) \\
- \mathbb{P}(v_t \text{ unlooped and joins a clique of size } i \text{ at time } t) \\
+ \mathbb{P}(v_t \text{ unlooped and joins a clique of size } i-1 \text{ at time } t)
\]

\[
= (1-q) \left[ - \left(1 - (1-q^{2i})Y_i(t)\right) \prod_{j=i+1}^{K} (1 - q^{2j})^{Y_j(t)} \\
+ \left(1 - (1-q^{2(i-1)})Y_{i-1}(t)\right) \prod_{j=i}^{K} (1 - q^{2j})^{Y_j(t)} \right],
\]

where \(Y(t) = (Y_1(t), \ldots, Y_K(t))\). For simplicity of notation, let \(\hat{g}_i(Y) \triangleq \prod_{j=i}^{K} (1 - q^{2j})^{Y_j(t)}\). Quantity \((q^2)^j\) is the probability of \(v_t\) to be adjacent to all vertices in a clique of size \(j\). Then, \((1 - q^{2j})^{Y_j(t)}\) is the probability that none of the \(Y_j(t)\) cliques of size \(j\) are suitable for \(v_t\) at time \(t\). Therefore, \(\hat{g}_i(Y)\) is the probability that at iteration \(t\), there exists no suitable clique of sizes
equal or greater than $i$ for $v_t$. Also, \( \left( 1 - \left(1 - q^{2(i-1)} \right)^{Y_{i-1}(t)} \right) \hat{g}_i(Y) \) is the probability that the largest suitable clique for $v_t$ has size $i - 1$, which implies that $Y_i(t+1) - Y_i(t) = 1$. Following the same line of argument for when $Y_i$ decrements at time $t$, and since in Algorithm 2 vertex $v_t$ joins the largest suitable clique, (4) follows.

Let $f_i(x, z_1, \ldots, z_K)$ be the right hand side of (4) with $Y_i(t)$ replaced by $Kz_i(x; q)$. Here, we used notation $z_i(\cdot; q)$ to show that every $z_i$ is parametrized by $q$. Then, the trend hypothesis of Wormald’s theorem is satisfied for all variables $Y_i$ and functions $f_i$ with $\lambda_1 = 0$. Functions $f_i$ are differentiable with bounded derivatives, hence they are Lipschitz continuous. Thus, based on Wormald’s theorem, we get the system of differential equations in (3b) for $z_i$, where the initial conditions result from $Y_i(0) = 0$. Also, for any $\lambda > 0$, we have $Y_i(t) = Kz_i(t/K; q) + O(\lambda K)$ with probability $1 - O(\frac{K}{\lambda} e^{-K\lambda^3})$. Since the number of the server messages is the same as the number of cliques formed by Algorithm 2, (3a) results.

Since the rates in Theorems 1 and 2 are concentrated, we propose the approximations

$$\mathbb{E}(R_m) \approx \frac{1}{2} \left[ K(1 - q) - \frac{\log(2 - (1 - q^2)^{K(1-q)})}{\log(1 - q^2)} \right].$$

(5)

and

$$\mathbb{E}(R_{cc}) \approx K \sum_{i=1}^{K} z_i(1)$$

(6)

for the expected delivery rates of CFC-MD and CFC-CCD, respectively. We expect the approximations to be tight, especially for larger numbers of caches. This is because the asymptotics denoted by $O$ in Theorems 1 and 2 are for $K \to +\infty$, and the probabilities in the statement of the theorems approach 1 as $K$ increases.

IV. PERFORMANCE COMPARISON AND SIMULATIONS

In this section, we present numerical examples and simulation results for the application of the CFC schemes that we proposed to demonstrate the effectiveness of CFC in reducing the

[3] Notice that $f_{n-l}$ only depends on $z_n, z_{n-1}, z_{n-(l+1)}$. Hence, by an argument similar to the one in Remark 8 one can solve for $f_{n-l}$ by restricting the equations to the ones that involve these variables.
delivery rate. We further show that the expressions in (5) and (6) tightly approximate the expected delivery rates of CFC-MD and CFC-CCD, respectively.

We use the expected delivery rates of two reference schemes to comment on the performance of our proposed CFC. The reference cases are the uncoded caching and the optimal decentralized CSC scheme derived in [18]. The latter is the state-of-the-art result on CSC which provides the optimal memory-rate tradeoff over all coded cachings with uncoded placement. Although the asymptotic average delivery rate derived in [18, Theorem 2] is valid in the infinite file size regime and the underlying caching scheme requires an exponential number of subfiles in $K$, it provides a suitable theoretical reference for our comparisons. To compute the expected delivery rate of the optimal decentralized CSC, one needs to find the expectation in the RHS of [18, Theorem 2]. This is done in Appendix B where it is shown that

$$
\mathbb{E}_d(R_{\text{CSC}}^*) = \frac{N - M}{M} \left[ 1 - \sum_{n=1}^{K} \frac{N^n}{N^K} \binom{K}{n} n! (1 - M/N)^n \right], \quad (7)
$$

where $\binom{K}{n}$ represents the Stirling number of the second kind [25, Section 5.3]. Also, the expected rate of uncoded caching is derived as

$$
\mathbb{E}_d(R_{\text{uncoded}}) = N \left( 1 - \left( 1 - \frac{1}{N} \right)^K \right) (1 - M/N) \quad \quad \quad \quad \quad \quad \quad (8)
$$

$$
= K(1 - M/N) + O(1/N^2)
$$

in Appendix B. We use the approximation $\mathbb{E}_d(R_{\text{uncoded}}) \approx K(1 - M/N)$ throughout this section, as we set $N = 10^4$ in all the examples provided here.

Fig. 2 shows the expected delivery rates of CFC-MD and CFC-CCD, as well as the average rates of the optimal CSC and the uncoded caching, for a network with $K = 50$ caches. First, we observe that CFC-CCD is notably effective in reducing the delivery rate. In particular, it reduces the delivery rate by 60 to 70 percent compared to uncoded caching. Notice that as argued in Sec. II for small cache sizes, the expected delivery rate of CFC-MD is close to that of CFC-CCD. As the cache sizes get larger, the probability of formation of larger cliques increases and therefore CFC-CCD considerably outperforms CFC-MD. Hence, in the small-cache regime, the use of CFC-MD is practically preferred to CFC-CCD because of its lower
computational complexity. Further, we observe that the theoretical expressions in (5) and (6) are in agreement with the empirical average rates obtained by simulations. This implies the tightness of the approximations derived.

For the clique cover delivery procedure in Algorithm 2, the trajectories of the expected number of cliques of different sizes are shown in Fig. 3. In this example, we consider a network with $K = 50$ caches and $M/N = 3/4$. Notice that although the cache size is large, the expected number of large cliques is relatively small. More precisely, cliques of small sizes have the most contribution to the construction of coded messages. This is because for large cache sizes, there is a high probability that each cache has the file requested by its user locally available. This results in a loop in the side information graph and therefore the corresponding vertex does not appear in any clique.

In Fig. 4, the per user expected delivery rates, i.e., the expected delivery rates normalized by $K$, are shown for different numbers of caches. Notice that although the asymptotics in the rate expressions in Theorems 1 and 2 are for $K \to \infty$, the proposed approximations closely match the simulation results for values of $K$ as small as 10. Accordingly, Fig. 5 shows the decrease in the expected delivery rate relative to uncoded caching when different coded cachings are

---

4 The trajectories are obtained by solving (3b) for $K \times z_i(x)$ over $0 \leq x \leq 1$, using MATLAB’s ode45 function.
Fig. 3: Evolution of the expected number of cliques of different sizes with iteration number $t$ for Algorithm 2. Here, $K = 50$ and $q = \frac{3}{4}$. The trajectories are shown for cliques of size 1 to 12.

Fig. 4: The expected per-user delivery rates for caching networks with different numbers of caches.

Fig. 5: The reduction in the expected delivery rates of coded caching schemes over uncoded caching.

used. First, notice that CFC-CCD significantly outperforms uncoded caching. Also, for large $KM/N$, the gap between CFC-CCD and the optimal subfile caching shrinks. This regime is of practical importance as coded caching is in general most effective when the number of caches in the network is large. Second, the performance gap between CFC-MD and CFC-CCD gets larger as $KM/N$ increases. This is because as the cache capacity and the number of caches increase, the probability of formation of larger cliques increases. Third, the improvement in the rate for CFC-MD is upper bounded by 50%. This is expected as in the extreme case of perfect
matching, the rate of CFC-MD is equal to half of the number of unlooped vertices. Hence, the use of CFC-MD is recommended in the small cache size regime.

V. EXTENSION TO SUBFILE CACHING

We have shown that the proposed CFC is an easy to implement yet effective technique to reduce the average delivery rate of caching networks. However, there is a gap between the delivery rates of the proposed CFC and the optimal CSC, as the latter allows for an arbitrary high complexity in terms of the number of subfiles used per file and also assumes infinite number of packets per file. It is of practical interest to explore the improvement in the expected delivery rate in a scenario where the system can afford a given level of complexity in terms of the number of subfiles used per file. In this section, we consider this problem and show that the placement and delivery proposed in Algorithms [1] and [2] also provide a framework for CSC when a small number of subfiles per file is used for caching. The resulting CSC scheme provides a means to trade off between the delivery rate and the implementation complexity, i.e., the number of subfiles used per file.

Throughout this section, performance evaluations are based on computer simulations as the theoretical results of Section III cannot be extended to subfile caching.

1) Placement: For subfile caching, we break each file in the library into $\Delta \geq 1$ subfiles. This leads to a library of $N\Delta$ subfiles, where each subfile is of length $F/\Delta$. Then, we apply the placement in Algorithm [1] to the library of subfiles. Notice that the proposed subfile placement is different from the decentralized placement of [10]. In particular, here all the subfiles are of the same length. Moreover, in the placement of [10], the number of packets of each file stored in each cache was $M/NF$, while this number is random here. Also notice that, for each cache, the prefetching of the subfiles that belong to different files are dependent, as the total size of the cached subfiles must be at most $M$.

Remark 5. The choice of $\Delta$ depends on the level of complexity that is tolerable to the system. The only restriction that the proposed placement imposes on $\Delta$ is for the ratio $F/\Delta$ to be an integer value. This condition can be easily satisfied in the finite file size regime.
2) Delivery: Similar to the delivery of file caching, for subfile caching the server forms the side information graphs $D$ and $G$ upon receiving the user demands. Then, it delivers the requests using Algorithm [2] by treating each subfile like a file. Notice that we do not consider CSC with online matching delivery in this section, as its coding gain is limited to 2 and this bound does not improve by increasing the number of subfiles.

Side information graphs: For subfile caching, both the side information graphs $D$ and $G$ have $K\Delta$ vertices. Let $w_{k,\delta}$ represent the vertex corresponding to cache $k$ and subfile $\delta$. In $D$, vertex $w_{k,\delta}$ has a loop if subfile $\delta$ of the file requested by cache $k$ is available in cache $k$. The probability of a loop is $q$. The main structural difference between graphs $D$ for file and subfile caching is that the presence of the edges are highly dependent in subfile caching. In particular, for two given caches $k$ and $l \neq k$, and a given subfile $\delta$ of the file requested by cache $k$, there exist $\Delta$ directed edges from $w_{l,\theta}$ to $w_{k,\delta}$ for all $\theta = 1, \ldots, \Delta$, if subfile $\delta$ of the file requested by cache $k$ is available in cache $l$. This event has probability $q$. Otherwise, all $\Delta$ edges from $w_{l,\theta}$ to $w_{k,\delta}$ with $\theta = 1, \ldots, \Delta$ are absent. Also, for subfile caching, we draw no edges from $w_{k,\delta}$ to $w_{k,\theta}$ for $\delta \neq \theta$. This does not affect the output of the delivery algorithm as if subfile $\theta$ of file $X_k$ is present in cache $k$, vertex $w_{k,\theta}$ is looped and hence is ignored (not joined to any clique) by Algorithm [2]. The latter two properties make the model of the random graph $D$ for subfile caching significantly different from the model used for file caching.

Graph $G$ is built from $D$ in exactly the same way as for file caching. As a result, each vertex is looped with probability $q$. Also, any two vertices belonging to two different caches are present with marginal probability of $q^2$. However, the presence of the edges between the vertices of two caches are highly dependent because of the discussed structures in $D$.

Notice that forming the joint side information graphs for the delivery of all $\Delta$ subfiles increases the coding opportunities compared to the case with $\Delta$ separate side information graphs for the disjoint delivery of the $\delta$-th subfiles of every file. In other words, the number of edges in the joint side information graph $G$ grows superlinearly in $\Delta$ as each of the $K\Delta$ vertices can connect to $(K-1)\Delta$ other vertices. Fig. 6 shows an example of separately formed and the corresponding jointly formed side information graphs.
Fig. 6: $K = 4$ caches and $\Delta = 2$ subfiles per file. (a), (b): Side information graphs for separate delivery of $\delta$-th subfiles with $\delta = 1, \ldots, \Delta$. (c), (d): Side information graphs for joint delivery of all subfiles.

Remark 6. Coding opportunities increase by increasing $\Delta$. Hence, in practice, $\Delta$ is determined by the highest level of complexity that is tolerable to the system in terms of the number of subfiles used.

**Delivery Algorithms:** The delivery process is complete with the side information graph $G$ inputted to Algorithm 2. We call the resulting scheme CSC with online Clique Cover Delivery (CSC-CCD).

3) Simulation Results: For a system with $K = 50$ caches, Fig. 7 shows the expected delivery rates of CSC-CCD with $\Delta = 5$ and $\Delta = 25$ as well as the expected delivery rates of CFC-CCD, the optimal decentralized CSC of [18] and uncoded caching. Notice that CFC-CCD is identical to CSC-CCD with $\Delta = 1$ subfile.

An important observation is that relative to the $2^K$ subfiles required by the optimal decentralized CSC, the use of a small number of subfiles in CSC-CCD can significantly shrink the rate gap between the delivery rates of CFC-CCD and the optimal decentralized CSC. This behavior is further highlighted in Fig. 8 where the improvement in delivery rates due to coding is shown. One observes that the rate of change in the delivery rate decays as the number of subfiles increases. For instance, increasing the number of subfiles from $\Delta = 1$ to $\Delta = 5$ results in a larger reduction in the delivery rate compared to the case where the number of subfiles is increased.
from $\Delta = 5$ to $\Delta = 25$. In Figs. 7 and 8, we have also shown approximate expected delivery rates based on the theoretical results in Theorem 2. More specifically, we use the approximation $R_{cc, \text{subfile}} \approx \frac{1}{\Delta} R_{cc}(K\Delta)$. As we discussed earlier, because of the structural differences between the side information graphs for file and subfile caching, such an approximation is generally prone to large errors. However, for $\Delta \ll K$, it still results in an approximation for the expected delivery rate.

VI. CONCLUSION

We explored the decentralized coded caching problem for the scenario where files are not broken into smaller parts (subfiles) during placement and delivery. This scenario is of practical importance because of its simpler implementation. We showed that although the requirement of caching the entirety of a file puts restrictions on creation of coding opportunities, coded file caching is still an effective way to reduce the delivery traffic of the network compared to the conventional uncoded caching. In particular, we proposed the CFC-CCD file caching scheme, which performs delivery based on an online clique cover algorithm operating over a certain side information graph. We further proposed a file caching called CFC-MD, which uses a simpler
delivery algorithm, and is designed for the small cache size regime. We derived approximate expressions for the expected delivery rate of both schemes.

Although because of the more restricted setup of the file caching problem, the schemes we proposed here are suboptimal compared to the coded subfile cachings, they still promise considerable gains over uncoded caching. Further, the delivery rate gap between CFC-CCD and the state-of-the-art decentralized coded subfile cachings shrinks considerably in the regime of large total storage capacity. These findings suggest that in scenarios where the implementation of subfile caching is difficult, it is still possible to effectively gain from network coding to improve the system performance.

Finally, we discussed the generalization of the proposed file caching schemes to subfile caching with an arbitrary number of subfiles per file. We observed that a considerable portion of the performance loss due to the restrictions of file caching can be recovered by using a relatively small number of subfiles per file. While the construction of most of the coded subfile caching methods requires large numbers of subfiles per file, this observation gives grounds for the development of new decentralized subfile cachings that require a small number of subfiles without causing a considerable loss in the delivery rate.

APPENDIX A
WORMALD’S THEOREM

In this appendix, we present Wormald’s theorem [24, Theorem 5.1] and provide a brief background and introduce the notations that we use in the proofs of Theorems 1 and 2.

Wormald’s theorem establishes connections between the dynamics of the discrete random variables involved in a random process and systems of differential equations. This relationship is developed based on the idea that a system of random steps will follow the expected trends with a high probability. More precisely, this is the case if (i) the expected changes in each step of the process are small, (ii) the rates of changes can be stated in terms of some differentiable functions, and (iii) these rates do not change too quickly in time.

\[\text{Wormald's theorem was introduced in [26, Theorem 1]. The most general setting of the theorem was established in [24, Theorem 5.1]. A simplified version of the theorem is provided in [27, Section 3], and examples of its application can be found in [24, 27, Section 4], [28] and [29, Section C.4].}\]
**Statement of Theorem:** Consider a sequence of random processes indexed by \( n, n = 1, 2, \ldots \). For each \( n \), the corresponding process is a probability space and the elements of the space are \((q^{(n)}_0, q^{(n)}_1, \ldots)\), where each \( q^{(n)}_i \in S^{(n)} \). Let \( h^{(n)}_t \) denote the history of process \( n \) up to time \( t \). We use \( H^{(n)}_t \) to denote the random counterpart of \( h^{(n)}_t \). Also, we denote by \( S^{(n)+} \) the set of all \( h^{(n)}_t = (q^{(n)}_0, \ldots, q^{(n)}_t) \), where \( q^{(n)}_i \in S^{(n)} \), \( t = 0, 1, \ldots \). For simplicity of notation, the dependence on \( n \) is usually dropped in the following.

Now, consider a set of variables \( W_1(t), \ldots, W_a(t) \) defined on the components of the processes and let \( w_i(h_t) \) denote the nonrandom counterpart of \( W_i(t) \). We are interested in analyzing the behavior of these random variables throughout the process.

**Theorem 3** (Wormald’s Theorem \[24, THEOREM 5.1\]). For \( 1 \leq l \leq a \), where \( a \) is fixed, let \( w_l : S^{(n)+} \to \mathbb{R} \) and \( f_l : \mathbb{R}^{a+1} \to \mathbb{R} \), such that for some constant \( c_0 \) and all \( l \), \( |w_l(h_t)| < c_0 n \) for all \( h_t \in S^{(n)+} \) for all \( n \). Assume the following three conditions hold, where in (ii) and (iii), \( D \) is some bounded connected open set containing the closure of

\[
\{(0, z_1, \ldots, z_a) : \mathbb{P}(W_i(0) = z_{ln}, l = 1, \ldots, a) = 0\}
\]

for some \( n \), and \( T_D(W_1, \ldots, W_a) \) is the minimum \( t \) such that \( (t/n, W_1(t)/n, \ldots, W_a(t)/n) \notin D \).

(i) \((\text{Boundedness hypothesis})\) For some functions \( \beta = \beta(n) \geq 1 \) and \( \gamma = \gamma(n) \), the probability that

\[
\max_{i=1, \ldots, a} |W_i(t+1) - W_i(t)| \leq \beta
\]

conditional upon \( H_t \), is at least \( 1 - \gamma \) for \( t < T_D \).

(ii) \((\text{Trend hypothesis})\) For some function \( \lambda_1 = \lambda_1(n) = o(1) \), for all \( l \leq a \)

\[
|\mathbb{E}(W_1(t+1) - W_i(t) \mid H_t) - f_l(t/n, W_1(t)/n, \ldots, W_a(t)/n) | \leq \lambda_1
\]

for \( t < T_D \).

\(^6\)For instance, the different processes can be the outcomes of a procedure implemented on graphs with different numbers of vertices \( n \). Then, each process associates with one of the graphs.
(iii) (Lipschitz hypothesis) Each function $f_i$ is continuous and satisfies a Lipschitz condition on 
$$D \cap \{(t, z_1, \ldots, z_a) : t \geq 0\},$$ 
with the same Lipschitz constant for each $l$.

Then, the following are true.

(a) For $(0, \hat{z}_1, \ldots, \hat{z}_a) \in D$, the system of differential equations defined by

$$\frac{dz_l}{dx} = f_i(x, z_1, \ldots, z_a), \quad l = 1, \ldots, a$$

has a unique solution that passes through $z_l(0) = \hat{z}_l$, $l = 1, \ldots, a$, which extends to points 
arbitrarily close to the boundary of $D$.

(b) Let $\lambda > \lambda_1 + c_3 n \gamma$ with $\lambda = o(1)$. For a sufficiently large constant $C$, with probability 

$$1 - O(n \gamma + \frac{\beta^3}{\lambda^3} e^{-n \lambda^3/\beta^3})$$

$$W_l(t) = nz_l(t/n) + O(\lambda n)$$

uniformly for all $0 \leq t \leq \sigma n$ and for each $l$, where $z_l(x)$ is the solution in (a) with 

$\hat{z}_l = \frac{1}{n} W_l(0)$, and $\sigma = \sigma(n)$ is the supremum of those $x$ to which the solution can be 
extended before reaching within $l^\infty$-distance $C\lambda$ of the boundary of $D$.

Hence, functions $z_l(x)$ model the behavior of $W_l(nx)/n$ for each $n$, and the solution to the 
system of differential equations provides a deterministic approximation to the dynamics of the 
process.

Remark 7. In the statement of the theorem, both variables $W_l$ and time $t$ are normalized by $n$. 
This is because in many applications, this normalization leads to only one set of differential 
equations for all $n$, instead of different systems for each $n$. Also, the asymptotics denoted by $O$ 
are for $n \to +\infty$, and the term “uniformly” in (b) refers to the convergence implicit in the $O$ 
terms.

Remark 8. A version of Theorem also holds when $a$ is a function of $n$, with the probability in 
(b) replaced by $1 - O(an \gamma + \frac{a^3}{\lambda^3} e^{-n \lambda^3/\beta^3})$, under the condition that all functions $f_i$ are uniformly 
bounded by some Lipschitz constant and $f_i$ depends only on the variables $x, z_1, \ldots, z_l$. The 
latter condition is because as $n \to \infty$, one needs to solve a system of infinite number of
differential equations which involves complicated technical issues. However, when \( f_i \) depends only on \( x, z_1, \ldots, z_l \), one can solve the finite systems obtained for each \( f_i \) by restricting the equations to the ones that involve \( x, z_1, \ldots, z_l \).

**APPENDIX B**

**DERIVATION OF (7) AND (8)**

In this appendix, we derive the expressions in (7) and (8) that we use to evaluate the expected delivery rate of uncoded caching and the expected rate in [18, Theorem 2].

As in [18], let \( N_e(d) \) be the number of distinct requests in demand vector \( d \). Since \( d \) is a random vector, \( N_e(d) \) is a random number in \( \{1, \ldots, K\} \). Given placement \( \mathcal{P} \), if the rate of a delivery algorithm \( A_D \) only depends on \( N_e(d) \) and not the individual files requested in \( d \), then, the expected delivery rate will be

\[
\mathbb{E}_d(R_{A_D}) = \sum_{m=1}^{K} \mathbb{P}(N_e(d) = m) R(d; \mathcal{P}, A_D).
\]

(9)

Assuming that the popularity distribution of files is uniform, we have

\[
\mathbb{P}(N_e(d) = m) = \frac{\binom{N}{m} \{\binom{K}{m}\} m!}{N^K}.
\]

(10)

This is because there are \( \binom{N}{m} \) ways to select \( m \) files out of \( N \) files. Also, there are \( \{\binom{K}{m}\} \) ways to partition a set of \( K \) objects into \( m \) non-empty subsets, where \( \{\binom{K}{m}\} \) is the Stirling number of the second kind [25, Section 5.3]. Further, there are \( m! \) ways to assign one of the \( m \) selected files to each subset. Therefore, there are \( \binom{N}{m} \{\binom{K}{m}\} m! \) demand vectors of length \( K \) with \( m \) distinct files from a library of \( N \) files. Since the total number of demand vectors is \( N^K \) and they are equiprobable, (10) results.

**Uncoded Caching:** Consider the cases of uncoded caching where delivery messages are uncoded. For placement every cache either stores the same set of \( M \) files out of the \( N \) files in the library, or every cache stores the first \( M/NF \) packets of every file. These correspond to uncoded file and subfile cachings. The rate required to delivery demand vector \( d \) with uncoded
messages is $N_e(d)(1 - M/N)$ files. Hence, based on (9) and (10), we get

$$
\mathbb{E}_d(R_{uncoded}) = \sum_{m=1}^{K} \binom{N}{m} \binom{K}{m} m! \times m \left(1 - \frac{M}{N}\right)
$$

$$
= N \left(1 - \left(1 - \frac{1}{N}\right)^K\right) \left(1 - \frac{M}{N}\right),
$$

where we used

$$
\sum_{m=1}^{K} m \binom{N}{m} \binom{K}{m} m! = N^{K+1} - N(N - 1)^K. \quad (11)
$$

The reason (11) holds is as follows. Consider a set of $K + 1$ objects that we want to partition into $m$ subsets such that the subset containing the first object has cardinality greater than 1, i.e., the first object is not the only object in the corresponding subset. This can be done in $m \binom{K}{m}$ ways as we can partition the rest of the $(K + 1) - 1$ objects into $m$ subsets and then add the first object to one of the resulting $m$ subsets. Assume that the subsets can be labeled distinctly from a set of $N \geq K$ labels. Then, there are $\sum_{m=1}^{K} m \binom{K}{m} \binom{N}{m} m!$ ways to partition $K + 1$ objects and label them with distinct labels such that the subset containing the first object has cardinality greater than 1. This sum is equal to the total number of ways to partition $K + 1$ objects and distinctly label them with the $N$ available labels, minus the number of ways to do the same but have the first object as the only element in its corresponding subset. The former counts to $N^{K+1}$, and the latter can be done in $N(N - 1)^K$ different ways. This proves (11).

**Optimal CSC:** From (9), the expected rate in the RHS of [18, eq. (27)] can be written as

$$
\mathbb{E}_d(R_{CSC}^*) = \sum_{m=1}^{K} \mathbb{P}(N_e(d) = m) \left[\frac{N - M}{M}(1 - (1 - M/N)^m)\right],
$$

which by substitution of (10), results in (7).

**REFERENCES**

[1] D. Liu, B. Chen, C. Yang, and A. F. Molisch, “Caching at the wireless edge: design aspects, challenges, and future directions,” *IEEE Commun. Mag.*, vol. 54, pp. 22–28, Sep. 2016.

[2] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, “Big data caching for networking: moving from cloud to edge,” *IEEE Commun. Mag.*, vol. 54, pp. 36–42, Sep. 2016.
[3] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, “Wireless caching: technical misconceptions and business barriers,” IEEE Commun. Mag., vol. 54, pp. 16–22, Aug. 2016.

[4] Y. Fadlallah, A. M. Tulino, D. Barone, G. Vettigli, J. Llorca, and J. M. Gorce, “Coding for caching in 5g networks,” IEEE Commun. Mag., vol. 55, pp. 106–113, Feb. 2017.

[5] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, “Femtocaching: Wireless content delivery through distributed caching helpers,” IEEE Trans. Inf. Theory, vol. 59, pp. 8402–8413, Dec. 2013.

[6] J. Qiao, Y. He, and X. S. Shen, “Proactive caching for mobile video streaming in millimeter wave 5g networks,” IEEE Trans. Wireless Commun., vol. 15, pp. 7187–7198, Oct. 2016.

[7] W. Han, A. Liu, and V. K. N. Lau, “Phy-caching in 5g wireless networks: design and analysis,” IEEE Commun. Mag., vol. 54, pp. 30–36, Aug. 2016.

[8] M. A. Maddah-Ali and U. Niesen, “Coding for caching: fundamental limits and practical challenges,” IEEE Commun. Mag., vol. 54, pp. 23–29, Aug. 2016.

[9] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” IEEE Trans. Inf. Theory, vol. 60, pp. 2856–2867, May 2014.

[10] M. A. Maddah-Ali and U. Niesen, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” IEEE/ACM Trans. Networking, vol. 23, pp. 1029–1040, Aug. 2015.

[11] N. Alon, E. Lubetzky, U. Stav, A. Weinstein, and A. Hassidim, “Broadcasting with side information,” in 2008 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 823–832, Oct. 2008.

[12] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, “Index coding with side information,” IEEE Trans. Inf. Theory, vol. 57, pp. 1479–1494, Mar. 2011.

[13] U. Niesen and M. A. Maddah-Ali, “Coded caching with nonuniform demands,” in IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 221–226, Apr. 2014.

[14] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, “Online coded caching,” in Proc. IEEE Int. Conf. Communications, pp. 1878–1883, June 2014.

[15] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. Diggavi, “Hierarchical coded caching,” in Proc. IEEE Int. Symp. Information Theory, pp. 2142–2146, June 2014.

[16] J. Hachem, N. Karamchandani, and S. N. Diggavi, “Effect of number of users in multi-level coded caching,” in Proc. IEEE Int. Symp. Information Theory, pp. 1701–1705, June 2015.

[17] J. Zhang, X. Lin, and X. Wang, “Coded caching under arbitrary popularity distributions,” in Proc. Information Theory and Applications Workshop (ITA), Feb. 2015.

[18] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory tradeoff for caching with uncoded prefetching,” arXiv preprint arXiv:1609.07817, 2016.

[19] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., “Finite-length analysis of caching-aided coded multicasting,” IEEE Trans. Inf. Theory, vol. 62, pp. 5524–5537, Oct. 2016.

[20] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, “Coded caching with linear subpacketization is possible using ruzsa-szeméredi graphs,” arXiv preprint arXiv:1701.07115, 2017.

[21] G. Vettigli, M. Ji, A. M. Tulino, J. Llorca, and P. Festa, “An efficient coded multicasting scheme preserving the multiplicative
caching gain,” in 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 251–256, Apr. 2015.

[22] D. A. Mastin, Analysis of approximation and uncertainty in optimization. PhD thesis, Massachusetts Institute of Technology, 2015. Available at [http://hdl.handle.net/1721.1/97761](http://hdl.handle.net/1721.1/97761)

[23] R. M. Karp, Reducibility among Combinatorial Problems, pp. 85–103. Boston, MA: Springer US, 1972.

[24] N. C. Wormald, “The differential equation method for random graph processes and greedy algorithms,” Lectures on approximation and randomized algorithms, pp. 73–155, 1999.

[25] P. J. Cameron, Combinatorics: topics, techniques, algorithms. Cambridge University Press, 1994.

[26] N. C. Wormald, “Differential equations for random processes and random graphs,” Ann. Appl. Probab., vol. 5, pp. 1217–1235, Nov. 1995.

[27] J. Díaz and D. Mitsche, “Survey: The cook-book approach to the differential equation method,” Comput. Sci. Rev., vol. 4, pp. 129–151, Aug. 2010.

[28] S. Yang and R. W. Yeung, “Batched sparse codes,” IEEE Trans. Inf. Theory, vol. 60, pp. 5322–5346, Sep. 2014.

[29] T. Richardson and R. Urbanke, Modern Coding Theory. New York, NY, USA: Cambridge University Press, 2008.