2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)

# Optimal Uncoded Placement and File Grouping Structure for Improved Coded Caching under Nonuniform Popularity

Yong Deng and Min Dong
Dept. of Electrical, Computer and Software Engineering, Ontario Tech University, Ontario, Canada

***Abstract*—This paper considers the caching design for coded caching under nonuniform file popularity. We investigate the optimal cache placement for the modified coded caching scheme (MCCS) recently proposed with an improved delivery strategy for rate reduction over the original coded caching scheme (CCS). We use the optimization framework for the cache placement problem to minimize the average delivery rate. Exploring several properties of the optimization problem and analyzing its structure, we obtain the file grouping structure under the optimal cache placement. We show that, regardless of file popularity, there are at most three file groups under the optimal cache placement. We further characterize the complete structure of the optimal cache placement and obtain the closed-form placement solution in these three possible file group cases. Following these, we develop a simple algorithm to obtain the final optimal cache placement solution, which only requires to compute a set of candidate solutions in closed-form. Simulation verifies the optimal solution produced by our algorithm. The optimal MCCS is shown to outperform existing schemes for both MCCS and CCS.**

## I. Introduction

Caching has recently emerged as a promising technology for future networks [1]. By storing data in distributed network storage resources near users, cache-aided systems alleviate the increasingly intensive traffic in wireless networks to meet low latency requirements. Conventional uncoded caching is inefficient for systems with multiple caches [2]. A *Coded Caching Scheme* (CCS) has recently been proposed [3] that combines a cache placement scheme specifying the (uncoded) cached contents and a coded multicasting delivery strategy. It is shown to be able to explore both global and local caching gain for substantial load reduction. As a result, coded caching has drawn considerable attention, with designs and analyses extended to various system models or network scenarios [4]–[9]. The original CCS [3] is designed to minimize the peak load in the worst-case scenario when users request distinct files. For the general case where multiple users request the same file, the CCS contains redundancy in the coded delivery. To address this, a recent study [10] has proposed a *Modified Coded Caching Scheme* (MCCS) with a modified delivery strategy to remove this redundancy, resulting in a reduced load than the CCS. The MCCS is then also applied to the device-to-device networks [11].

A key design issue in coded caching is the cache placement. An effective cache placement scheme increases caching gain and minimizes the load (rate) in the delivery phase. To study the fundamental caching limit, many existing works (*e.g.,* [4]–[11]) assume uniform file popularity. Under this assumption, the optimal cache placement that minimizes the average rate in data delivery is symmetric among files for both the CCS [12], [13], and the MCCS [10], where the latter provides the exact trade-off between the cache size and the rate. For nonuniform file popularity, the cache placement may be different among files, complicating both design and analysis. There is a fundamental question on whether to distinguish files of different popularities for cache placement and to what extent. There may be a trade-off in design complexity and performance in practical systems.

Existing works on caching design under nonuniform file popularity are all based on the CCS [2], [12]–[16]. For simplifying the problem, file grouping has first been proposed in [2], where files are divided into groups with cache allocated to different groups, and the decentralized CCS with symmetric cache placement is applied for each file group. File grouping has since been considered a tractable popular method for cache placement design. For the CCS, through heuristics, [2], [14]–[16] have proposed different file grouping methods for cache placement (typically into two groups), and different achievable rates and information-theoretical lower bounds on the rate have been obtained. These strategies are either suboptimal or designed only for certain file popularity distributions. For the MCCS, despite the improvement it is shown for uniform file popularity, there is no heuristic cache placement design proposed for nonuniform file popularity. The optimization framework is used to study the cache placement for both CCS [12], [13], and MCCS [17] under nonuniform file popularity. However, they focus on numerical methods in solving the problem and are unable to provide insight into the structure of the optimal cache placement. In particular, the optimal cache placement and its connection to file grouping remain unknown.

In this paper, we obtain the optimal cache placement for the MCCS to minimize the delivery rate for arbitrary file popularity distribution and cache size. Focusing on the placement structure, we characterize the optimal scheme and its connection to file grouping. Instead of constructing a caching scheme as in many existing works, we use the optimization framework for the cache placement problem to minimize the delivery rate. We adopt structural simplifications for the tractability of the problem, which have been numerically justified to be without loss of optimality [17]. With further exploration of the properties in the optimization problem, this allows us to transform the problem into a linear programming (LP) problem. By analyzing the problem structure, we identify the file grouping structure under the optimal cache placement.

ISBN 978-3-903176-29-4 

We show that, regardless of file popularity, there are at most three file groups under the optimal cache placement, each with a unique placement pattern. We completely characterize the structure of the optimal cache placement and obtain the closed-form placement solution in these three possible file group cases. Following this, we develop a simple and efficient algorithm to obtain the final optimal cache placement solution, which requires to compute a set of candidate solutions in closed-form and in parallel. Insight into the somewhat surprising file grouping result is also provided. The optimal cache placement solution given by our algorithm is verified through simulation. The optimal placement for the MCCS outperforms other existing schemes either for MCCS or for CCS, where the gap is more evident for a smaller cache size as a better cache placement is more critical for caching gain.

## II. System Model

Consider a cache-aided transmission system with a server and $K$ users, each with a local cache, connected over a shared error-free link. The server has a database consisting of $N$ files $\{W_1, \ldots, W_N\}$. Each file $W_n$ is of size $F$ bits and has probability $p_n$ for being requested. Let $\mathbf{p} \triangleq [p_1, \ldots, p_N]$ denote the file popularity distribution, where $\sum_{n=1}^{N} p_n = 1$. Without loss the generality, we label files according to the decreasing order of their popularities: $p_1 \geq p_2 \geq \cdots \geq p_N$. Each user $k$ has a cache size $M$ (normalized by the file size) with capacity $MF$ bits, where $M$ is arbitrary within the range $[0, N]$. Denote the file and user index sets by $\mathcal{N} \triangleq \{1, \ldots, N\}$ and $\mathcal{K} \triangleq \{1, \ldots, K\}$, respectively.

The coded caching operates in the cache placement phase and the content delivery phase. In the cache placement phase, a portion of uncoded file contents are placed in each user's local cache, specified by a cache placement scheme. Assume each user $k$ independently requests a file with index $d_k$ from the server. Let $\mathbf{d} \triangleq [d_1, \ldots, d_K]$ denote the demand vector of $K$ users. In the content delivery phase, based on the demand vector $\mathbf{d}$ and the cached contents at users, the server generates coded messages containing uncached portions of requested files and transmits to the users. Upon receiving the coded messages, each user $k$ reconstructs its requested file $W_{d_k}$ from the received coded messages and its cached content. Note that for a valid coded caching scheme, each user $k$ is able to reconstruct its requested file for any demand vector $\mathbf{d} \in \mathcal{N}^K$ over an error-free link.

## III. The Coded Caching Problem Setup

The cache placement is a key design issue in coded caching. Among existing studies for the CCS, a common approach is to propose a cache placement scheme, construct a lower bound on the minimum data rate, and evaluate the proposed scheme by comparing its performance with the lower bound. Different from this, we use an optimization approach for the cache placement design for the MCCS. Through construction, we formulate the cache placement problem into a design optimization problem.

*1) Cache Placement:* For $K$ users, there are total $2^K$ user subsets in $\mathcal{K}$, with subset sizes ranging from $0$ to $K$. Among them, there are $\binom{K}{l}$ different user subsets with the same size $l$, for $l = 0, \ldots, K$ (size $0$ for the empty subset $\emptyset$). They form a cache subgroup that contains all user subsets of size $l$, defined as $\mathcal{A}^l \triangleq \{\mathcal{S} : |\mathcal{S}| = l,\ \mathcal{S} \subseteq \mathcal{K}\}$ with $|\mathcal{A}^l| = \binom{K}{l}$, for $l = 0, \ldots, K$. Partition each file $W_n$ into $2^K$ non-overlapping subfiles, one for each unique user subset $\mathcal{S} \subseteq \mathcal{K}$, denoted by $W_{n,\mathcal{S}}$ (it can be $\emptyset$). Each user in user subset $\mathcal{S}$ stores subfile $W_{n,\mathcal{S}}$ in its local cache (subfile $W_{n,\emptyset}$ is not stored in any user's cache but only kept at the server). For any caching scheme, each file should be able to be reconstructed by combining all its subfiles. Thus, we have the file partitioning constraint $\sum_{l=0}^{K} \sum_{\mathcal{S} \in \mathcal{A}^l} |W_{n,\mathcal{S}}| = F$, for $n \in \mathcal{N}$. This construction through subfile and user subset partitioning is general to represent any cache placement.

To reduce the number of variables and simplify the optimization problem for its tractability, we impose the following condition: for each file $W_n$, the size of its subfile $W_{n,\mathcal{S}}$ only depends on $|\mathcal{S}|$, *i.e.,* $|W_{n,\mathcal{S}}|$ is the same for any $\mathcal{S} \in \mathcal{A}^l$ of the same size. This condition is proven to be the property of the optimal placement solution for the CCS [13]. Although the same is difficult to prove for the MCCS, it is numerically verified in [17] that imposing this condition results in no loss of optimality. Based on this condition, the subfiles of file $W_n$ are grouped into file subgroups, each denoted by $\mathcal{W}_n^l = \{W_{n,\mathcal{S}} : \mathcal{S} \in \mathcal{A}^l\}$, for $l = 0, \ldots, K$. As a result, there are $\binom{K}{l}$ subfiles of the same size in $\mathcal{W}_n^l$ (intended for user subsets in cache subgroup $\mathcal{A}^l$), and there are total $K + 1$ file subgroups. Following this, let $a_{n,l}$ denote the size of subfiles in $\mathcal{W}_n^l$, as a fraction of file $W_n$ size $F$ bits, *i.e.,* $a_{n,l} \triangleq |W_{n,\mathcal{S}}|/F$ (for $\forall \mathcal{S} \in \mathcal{A}^l$), $l = 0, \ldots, K$, $n \in \mathcal{N}$. Note that $a_{n,0}$ represents the fraction of file $W_n$ that is not stored at any user's cache but only remains at the server. Then, the file partition constraint is simplified to

$$\sum_{l=0}^{K} \binom{K}{l} a_{n,l} = 1,\ n \in \mathcal{N}. \tag{1}$$

Recall that in file partitioning, each subfile is intended for a unique user subset. During the cache placement, user $k$ stores all the subfiles in $\mathcal{W}_n^l$ that are intended for user subsets that contain the user, *i.e.,* $\{W_{n,\mathcal{S}} : k \in \mathcal{S} \text{ and } \mathcal{S} \in \mathcal{A}^l\} \subseteq \mathcal{W}_n^l$, for $l = 1, \ldots, K$. Note that in each $\mathcal{A}^l$, there are total $\binom{K-1}{l-1}$ different user subsets containing the same user $k$. Thus, there are $\sum_{l=1}^{K} \binom{K-1}{l-1}$ subfiles in each file $W_n$ that a user can store in its local cache. With subfile size $a_{n,l}$, this means that in total, a fraction $\sum_{l=1}^{K} \binom{K-1}{l-1} a_{n,l}$ of file $W_n$ is cached by a user. Given cache size $M$at each user, we have the following local cache constraint

$$\sum_{n=1}^{N} \sum_{l=1}^{K} \binom{K-1}{l-1} a_{n,l} \leq M. \tag{2}$$

*2) Content Delivery:* During the content delivery phase, the server multicasts coded messages to different user subsets. Each coded message corresponds to a user subset $\mathcal{S}$,

formed by bitwise XOR operation of the subfiles as $C_{\mathcal{S}} \triangleq \bigoplus_{k\in\mathcal{S}} W_{d_k,\mathcal{S}\backslash\{k\}}$. In the original CCS, the server simply delivers the coded messages formed by all the user subsets, for any file demand $\mathbf{d}$. However, under random demands, the same file may be requested by multiple users, causing redundant coded messages. In the MCCS, this redundancy is removed by the modified coded delivery strategy, resulting in the reduction of the average rate. Consider the following two definitions:

*Leader group*: For demand vector $\mathbf{d}$ with $\widetilde{N}(\mathbf{d})$ distinct requests, the leader group $\mathcal{D}$ is chosen from all the user subsets, where $\mathcal{D} \subseteq \mathcal{K}$ satisfies $|\mathcal{D}| = \widetilde{N}(\mathbf{d})$ and users in $\mathcal{D}$ have exactly $\widetilde{N}(\mathbf{d})$ distinct requests.

*Redundant group*: Any user subset $\mathcal{S} \subseteq \mathcal{K}$ is called a redundant group, if $\mathcal{S} \cap \mathcal{D} = \emptyset$; otherwise, it is a non-redundant group.

The delivery scheme in the MCCS is to multicast coded messages, formed by the non-redundant groups $\{C_{\mathcal{S}} : \forall \mathcal{S} \subseteq \mathcal{K}$ and $\mathcal{S}\cap\mathcal{D} \neq \emptyset\}$, to both non-redundant and redundant groups. Since subfiles in different files may be of different sizes, all the subfiles in a coded message are zero-padded to the size of the largest subfile among them. Thus, the size of $C_{\mathcal{S}}$ formed by non-redundant group $\mathcal{S}$ of size $l+1$, is determined by the largest subfile forming $C_{\mathcal{S}}$: $|C_{\mathcal{S}}| = \max_{k\in\mathcal{S}} a_{d_k,l}$.

## IV. Cache Placement Optimization Formulation

Based on the size of coded messages, the average rate $\bar{R}$ in the delivery phase is given by

$$\bar{R} = \mathbb{E}_{\mathbf{d}}\Big[\sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\cap\mathcal{D}\neq\emptyset} |C_{\mathcal{S}}|\Big] = \mathbb{E}_{\mathbf{d}}\Big[\sum_{\mathcal{S}\subseteq\mathcal{K},\mathcal{S}\cap\mathcal{D}\neq\emptyset} \max_{k\in\mathcal{S}} a_{d_k,l}\Big] \quad (3)$$

where $\mathbb{E}_{\mathbf{d}}[\cdot]$ is taken w.r.t. demand vector $\mathbf{d}$. From the definition of $a_{n,l}$, let $\mathbf{a}_n \triangleq [a_{n,0},\ldots,a_{n,K}]^T$ denote the $(K+1)\times 1$ cache placement vector for file $W_n$, $n \in \mathcal{N}$. Our goal is to optimize $\{\mathbf{a}_n\}$ to minimize the average rate $\bar{R}$ in (3). To simplify the expression in (3) for the optimization problem, following the intuition that more cache is allocated to the file with a higher popularity, we impose a *popularity-first* condition: With file popularity $p_1 \geq \ldots \geq p_N$, the following holds for the cached subfiles

$$a_{n,l} \geq a_{n+1,l}, \quad l \in \mathcal{K},\ n \in \mathcal{N}\backslash\{N\}. \quad (4)$$

This condition turns out to be the property of the optimal placement for the CCS [13] under nonuniform file popularity. It is adopted for the MCCS in [17] and is numerically shown that there is no loss of optimality. Following this, we explicitly imposing constraint (4) and formulate the cache placement optimization problem as follows

$$\mathbf{P0}: \min_{\{\mathbf{a}_n\}} \ \bar{R} \quad \text{s.t. } (1),(2),(4), \text{ and}$$
$$\mathbf{a}_n \succcurlyeq \mathbf{0},\ n \in \mathcal{N}. \quad (5)$$

At the optimality, it is easy to show that the cache memory is always fully utilized, and the local cache constraint (2) is attained with equality. Therefore, constraint (2) can be replaced by the equality constraint

$$\sum_{n=1}^{N}\sum_{l=1}^{K}\binom{K-1}{l-1}a_{n,l} = M. \quad (6)$$

Next, from the popularity-first condition (4), we conclude that if the following two inequalities hold

$$a_{1,0} \geq 0 \quad \text{and} \quad a_{N,l} \geq 0, \quad l \in \mathcal{K}, \quad (7)$$

then (5) holds. To see this, note that if $a_{N,l} \geq 0$, $l \in \mathcal{K}$, by (4), we have $a_{n,l} \geq 0$, $\forall l \in \mathcal{K}$, $\forall n \in \mathcal{N}$. Recall that $a_{n,0}$ represents the fraction of subfiles in file $W_n$ that are not stored at any user's cache. From (1), we have

$$a_{n,0} = 1 - \sum_{l=1}^{K}\binom{K}{l}a_{n,l}, \quad n \in \mathcal{N}. \quad (8)$$

Combining (4) and (8), we have $a_{1,0} \leq \ldots \leq a_{N,0}$. Thus, if $a_{1,0} \geq 0$ in (7) holds, then $a_{n,0} \geq 0$, $\forall n \in \mathcal{N}$. Combining the above, we have $\mathbf{a}_n \succcurlyeq 0$, $\forall n \in \mathcal{N}$. Thus, constraints (4) and (5) can be equivalently replaced by constraints (4) and (7).

Finally, by the popularity-first condition (4), the average rate $\bar{R}$ in (3) can be expressed by [17]

$$\bar{R} = \sum_{l=0}^{K-1}\binom{K}{l+1}\sum_{n=1}^{N}\left(\left(\sum_{n'=n}^{N}p_{n'}\right)^{l+1} - \left(\sum_{n'=n+1}^{N}p_{n'}\right)^{l+1}\right)a_{n,l}$$
$$- \sum_{u=1}^{\min\{N,K\}}\sum_{l=0}^{K-u-1}\binom{K-u}{l+1}\sum_{i=1}^{K-u}\binom{K-u-i}{l}\sum_{n=1}^{N}P'_{i,u,n}a_{n,l} \quad (9)$$

where $P'_{i,u,n}$ is the joint probability of $u$ distinct file requests, and file $W_n$ being the $i$-th least popular file requested by all the users not in the leader group. The expression of $P'_{i,u,n}$ is lengthy and non-essential in developing our result. Therefore, we omit it here, but only point out that $P'_{i,u,n}$ is not a function of $\mathbf{a}_n$. Expression in (9) indicates that $\bar{R}$ is a weighted sum of $a_{n,l}$'s. By (6)(7)(9), define $\mathbf{g}_n \triangleq [g_{n,0},\ldots,g_{n,K}]^{\mathrm{T}}$, with

$$g_{n,l} \triangleq \binom{K}{l+1}\left(\left(\sum_{n'=n}^{N}p_{n'}\right)^{l+1} - \left(\sum_{n'=n+1}^{N}p_{n'}\right)^{l+1}\right)$$
$$- \sum_{u=1}^{\min\{N,K\}}\binom{K-u}{l+1}\sum_{i=1}^{K-u}\binom{K-u-i}{l}P'_{i,u,n}, \quad (10)$$

$\mathbf{b} \triangleq [b_0,\ldots,b_K]^T$ with $b_l \triangleq \binom{K}{l}$, and $\mathbf{c} \triangleq [c_0,\ldots,c_K]^T$ with $c_l \triangleq \binom{K-1}{l-1}$[1]. The cache placement problem **P0** is then reformulated into the following equivalent LP problem

$$\mathbf{P1}: \min_{\{\mathbf{a}_n\}} \ \sum_{n=1}^{N}\mathbf{g}_n^{\mathrm{T}}\mathbf{a}_n \quad \text{s.t. } (4),(7),$$
$$\mathbf{b}^{\mathrm{T}}\mathbf{a}_n = 1, \quad n \in \mathcal{N}, \quad (11)$$
$$\sum_{n=1}^{N}\mathbf{c}^{\mathrm{T}}\mathbf{a}_n = M. \quad (12)$$

## V. Optimal Cache Placement: Solution Structure

In this section, we first present a structural property of the optimal solution for **P1**. This property helps us identify several possible optimal solution structures. Through further analysis, we obtain the closed-form solution under each different structure. Finally, we develop a simple low-complexity algorithm

[1]We define $\binom{K}{l} = 0$, for $l < 0$ or $l > K$.

based on the closed-form candidate solutions to obtain the optimal solution for **P1**. First, we define *file group* below.

**Definition 1.** *File group*: A file group is a subset of $\mathcal{N}$ that contains all files with the same cache placement vector, *i.e.*, for any two files $W_n$ and $W_{n'}$, if their placement vectors $\mathbf{a}_n = \mathbf{a}_{n'}$, then they belong to the same file group.

For $N$ files, there could be as many as $N$ file groups (*i.e.*, all $\mathbf{a}_n$'s are different), making the design of optimal cache placement a major challenge. For the CCS, file grouping has been considered to simplify the cache placement design [2], [14]–[16]. Having a fewer file groups reduces the complexity in designing placement vectors $\{\mathbf{a}_n\}$. However, how to form file groups remains heuristic in existing works. File grouping has not been considered for the MCCS. Our main result in Theorem 1 below describes the structural property of the optimal cache placement for the MCCS, in terms of file groups.

**Theorem 1.** For $N$ files with any file popularity distribution $\mathbf{p}$, any $M \leq N$, and $K$, there are at most three file groups under the optimal cache placement $\{\mathbf{a}_n\}$ for **P1**.

*Proof:* See Appendix A. ∎

Theorem 1 states that, regardless of $N$, $\mathbf{p}$, and $M$, there are only three possible file grouping structures under the optimal cache placement. The result implies that there are at most three unique vectors among the optimal cache placement vectors $\{\mathbf{a}_n\}$, one for each file group. This property drastically reduces the complexity in solving the cache placement problem, and in turn, it allows us to explore the structure and obtain the optimal solution $\{\mathbf{a}_n\}$ analytically. In the following, we examine all the three cases for **P1** to obtain the solution. Note that the result of at most three file groups, regardless of file popularity distribution $\mathbf{p}$ is somewhat surprising. We will provide some insight into this result in Section V-D, after the placement structure and solution in each case is derived. We first define the following notations:

1) Denote $\bar{\mathbf{a}}_n = [a_{n,1}, \ldots, a_{n,K}]^T$ as the sub-placement vector in $\mathbf{a}_n$. It specifies only the subfiles stored in the local cache, and $a_{n,0}$ specifies the subfile kept at the server.

2) We use notation $\bar{\mathbf{a}}_n \succcurlyeq_1 \mathbf{0}$ to indicate that there is at least one positive element in $\bar{\mathbf{a}}_n$; otherwise, $\bar{\mathbf{a}}_n = \mathbf{0}$.

3) Notation $\bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_2}$ denotes that there is at least one element in $\bar{\mathbf{a}}_{n_1}$ greater than that of $\bar{\mathbf{a}}_{n_2}$, and the rest elements of the same position in $\bar{\mathbf{a}}_{n_1}$ and $\bar{\mathbf{a}}_{n_2}$ are equal.

For any two files $n_1$ and $n_2$, it is easy to verify that

$$\mathbf{a}_{n_1} = \mathbf{a}_{n_2} \;\Leftrightarrow\; \bar{\mathbf{a}}_{n_1} = \bar{\mathbf{a}}_{n_2}, \tag{13}$$

$$\mathbf{a}_{n_1} \neq \mathbf{a}_{n_2} \;\Leftrightarrow\; \bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_2} \text{ and } a_{n_1,0} < a_{n_2,0}, \tag{14}$$

where "$\Leftrightarrow$" denotes being equivalent, and (14) is obtained by (4) and (8). Below, we assume each of the three possible file grouping cases under the optimal solution, and identify the complete structure of the cache placement vector and obtain the solution.

### A. One File Group

In this case, the cache placement vectors are identical for all files. Let $\mathbf{a}_1 = \cdots = \mathbf{a}_N = \mathbf{a}$. Let $P_{\mathrm{u}}(\tilde{n})$ denote the probability of having $\tilde{n}$ distinct requests in $\mathbf{d}$, for $\tilde{n} = 1, \ldots, \min\{N, K\}$. We have $P_{\mathrm{u}}(\tilde{n}) = \mathbf{S}(K, \tilde{n})\binom{N}{\tilde{n}}\frac{\tilde{n}!}{N^K}$, where $\mathbf{S}(\cdot,\cdot)$ is the Stirling number of the second [18]. Define $\tilde{\mathbf{g}}_l \triangleq [\tilde{g}_0, \ldots, \tilde{g}_K]^T$ where $\tilde{g}_l \triangleq \binom{K}{l+1} - \sum_{\tilde{n}=1}^{\min\{N,K\}} P_{\mathrm{u}}(\tilde{n})\binom{K-\tilde{n}}{l+1}$, $l = 0, \ldots, K$. Then, **P1** in this case is simplified to the following problem

$$\mathbf{P2}: \min_{\mathbf{a}} \; \tilde{\mathbf{g}}^T\mathbf{a} \quad \text{s.t. } \mathbf{b}^T\mathbf{a} = 1,\ \mathbf{c}^T\mathbf{a} = M/N,\ \mathbf{a} \succcurlyeq \mathbf{0}.$$

Problem **P2** is identical to the cache placement optimization problem for the uniform file popularity case (with the same placement $\mathbf{a}$ for all files), of which the optimal solution has been shown in [19] in closed-form. It shows that, the optimal $\mathbf{a}$ for **P2** is given as follows:

$$a_{l_o} = \frac{l_o + 1 - \frac{MK}{N}}{\binom{K}{l_o}},\ a_{l_o+1} = \frac{\frac{MK}{N} - l_o}{\binom{K}{l_o+1}},\ \text{for } l_o = \left\lfloor \frac{MK}{N} \right\rfloor, \tag{15}$$

and $a_l = 0$, for $\forall l \neq l_o, l_o + 1$. It is clear from (15) that, the optimal $\mathbf{a}$ has at most two nonzero elements which are adjacent to each other: When $MK/N$ is an integer, $\mathbf{a}$ has one nonzero element, *i.e.*, each file is partitioned into equal subfiles of size $a_{l_o}$; otherwise, $\mathbf{a}$ has two nonzero adjacent elements, *i.e.*, each file is partitioned into subfiles of two different sizes $(a_{l_o}, a_{l_o+1})$. Each subfile is cached into its intended user subset (of size $l_o$ or $l_o + 1$) as specified in Section III-1.

### B. Two File Groups

For two file groups, there are only two unique placement vectors in $\{\mathbf{a}_n\}$. This implies that $\{\mathbf{a}_n\}$ has the following structure: $\mathbf{a}_1 = \ldots = \mathbf{a}_{n_o} \neq \mathbf{a}_{n_o+1} = \ldots = \mathbf{a}_N$, for some $n_o \in \{1, \ldots, N-1\}$. By (13) and (14), this is equivalent to

$$\begin{cases} \bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_N \\ a_{1,0} = \ldots = a_{n_o,0} < a_{n_o+1,0} = \cdots = a_{N,0}. \end{cases} \tag{16}$$

It immediately follows that $a_{n_o+1,0} = \cdots = a_{N,0} > 0$.

Based on (16), we use $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ to represent the two unique placement vectors for the first and the second file group, respectively, for some $n_o \in \{1, \ldots, N-1\}$. We first characterize the structure of $\mathbf{a}_{n_o+1}$ in the second file group.

**Proposition 1.** If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$, the optimal sub-placement vector $\bar{\mathbf{a}}_{n_o+1}$ for the second file group has at most one nonzero element.

*Proof:* The proof is similar to that of Theorem 1. We use proof by contradiction. By exploring the KKT conditions of **P1** and by identifying and combining some conditions in specific ways, we show that for two file groups, if $\bar{\mathbf{a}}_{n_o+1}$ has two nonzero elements, the Lagrangian multipliers would not have feasible solutions. ∎

Proposition 1 indicates that $\bar{\mathbf{a}}_{n_o+1}$ has either zero or one nonzero element. Note that no existing studies have considered two file grouping strategies for cache placement. For the CCS, two-group placement strategies have been considered in [14]

**Algorithm 1** The cache placement for the extended two-file-group case with $\bar{\mathbf{a}}_{n_o+1} = 0$ (including the one-file group case)

**Input:** $K$, $M$, $N$, and $\mathbf{p}$

**Output:** $(\bar{R}_{\min}, n_o^*)$

1: **for** $n_o = 1$ to $N$ **do**
2: Set $l_o = \lfloor \frac{MK}{n_o} \rfloor$; Set $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$.
3: Determine $\mathbf{a}_{n_o}$ by (17).
4: Compute $\bar{R}(n_o)$ by (9).
5: **end for**
6: Compute $\bar{n}_o^* = \text{argmin}_{n_o} \bar{R}(n_o)$; $\bar{R}_{\min} = \bar{R}(n_o^*)$.

and [15], where the second file group containing less popular files remain at the server. These strategies correspond to the case where $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$. However, the location of $n_o$ was only proposed heuristically in different ways in these two works. Except these, the case of allocating cache to the second file group, *i.e.*, $\bar{\mathbf{a}}_{n_o+1} \neq \mathbf{0}$, has never been considered in the existing literature.

Following Proposition 1, we discuss the optimal cache placement in each of the two cases for $\bar{\mathbf{a}}_{n_o+1}$ below:

*1)* $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$*:* In this case, no cache is allocated to the second file group; the entire cache is given to the first file group. As a result, $\mathbf{a}_{n_o+1}$ is given by $\bar{\mathbf{a}}_{n_o+1} = \mathbf{0}$ and $a_{n_o+1,0} = 1$. The cache placement problem for $\mathbf{a}_{n_o}$ of the first group is reduced to the one in the one-file-group case in Section V-A. Specifically, let $\mathbf{a}_1 = \ldots = \mathbf{a}_{n_o} = \mathbf{a}$. We can simply treat the first file group as a new database with the number of files being $n_o$ instead of $N$. Following **P2**, the cache placement optimization problem is given by

$$\textbf{P3:} \min_{\mathbf{a}} \ \tilde{\mathbf{g}}^T \mathbf{a} \ \text{ s.t. } \quad \mathbf{b}^T \mathbf{a} = 1, \ \mathbf{c}^T \mathbf{a} = M/n_o, \ \mathbf{a} \succcurlyeq \mathbf{0}.$$

By the result in the one-file-group case, the optimal solution $\mathbf{a}$ for **P3** has at most two nonzero elements. The solution is similar to (15), except that $N$ is replaced by $n_o$, given by

$$a_{l_o} = \frac{l_o + 1 - \frac{MK}{n_o}}{\binom{K}{l_o}}, \ a_{l_o+1} = \frac{\frac{MK}{n_o} - l_o}{\binom{K}{l_o+1}}, \ \text{for } l_o = \left\lfloor \frac{MK}{n_o} \right\rfloor, \quad (17)$$

and $a_l = 0$, for $\forall l \neq l_o, l_o + 1$.

Combining (15) and (17), we can extend the two-file-group case to include the one-file-group as a special case where $n_o = N$. In this case, the optimal cache placement solution is given by (17) with $n_o \in \{1, \ldots, N\}$. What remains is to obtain the optimal $n_o^*$ to determine $\{\mathbf{a}_n\}$ that minimizes the average rate objective in **P1**. The optimal $n_o^*$ depends on $(N, \mathbf{p}, M, K)$, which is challenging to obtain analytically. Nonetheless, $\bar{R}$ can be easily computed using (17) for $n_o = 1, \ldots, N$, and we can do a search for $n_o$ to determine $n_o^*$ that gives the minimum $\bar{R}$. The method in finding the placement solution $\{\mathbf{a}_n\}$ in this case is summarized Algorithm 1. Since the algorithm uses a 1-D search for $n_o \in \mathcal{N}$ for the optimal $n_o^*$, it computes $\bar{R}$ using the closed-form expression in (9) by $N$ times.

*2)* $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$*:* In this case, for the second file group, by Proposition 1, $\bar{\mathbf{a}}_{n_o+1}$ has only one nonzero element. Assuming it is the $l_o$-th element, then $a_{n_o+1,l_o} > 0$, and $a_{n_o+1,l} = 0$, $\forall l \neq l_o$, $l \in \mathcal{K}$. For the first file group, Proposition 2 characterizes $\mathbf{a}_{n_o}$, and Proposition 3 specifies the differences of $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$ for the two file groups. The proofs use similar techniques as in that of Proposition 1 and are omitted.

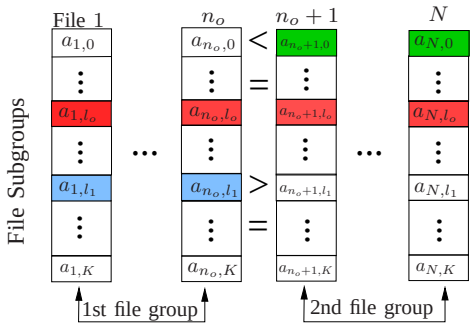


Fig. 1. An example of the optimal cache placement $\{\mathbf{a}_n\}$ in the case of two file groups with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$: $a_{n_o+1,0} > a_{n_o,0} = 0$. Between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$: 1) $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$; 2) $a_{n_o,l_o} = a_{n_o+1,l_o} > 0$; 3) $a_{n_o,l} = a_{n_o+1,l} = 0$, $\forall l \in \mathcal{K}$, $l \neq l_o, l_1$.

**Proposition 2.** If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$ with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, then $a_{1,0} = \ldots = a_{n_o,0} = 0$.

**Proposition 3.** If there are two file groups under the optimal cache placement $\{\mathbf{a}_n\}$ with $\bar{\mathbf{a}}_{n_o+1} \succcurlyeq_1 \mathbf{0}$, then $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are different by only one element.

Proposition 2 indicates that each file in the first file group has all its subfiles cached among $K$ users, and no subfile solely remains in the server. Recall that $\bar{\mathbf{a}}_{n_o+1}$ has only one nonzero element $a_{n_o+1,l_o} > 0$. By Proposition 3, the different element between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ can be either at index $l_o$ or some $l_1$, for $l_1 \neq l_o$. From the popularity-first condition (4), it follows that either of the following two cases holds: *2.i)* $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$; or *2.ii)* $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$, for some $l_1 \neq l_o$, $l_1 \in \mathcal{K}$. The structure of $\{\mathbf{a}_n\}$ in Case 2.ii) is illustrated in Fig. 1. We point out that $l_o$ and $l_1$ are not necessarily adjacent to each other. Now we derive the solution $(\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1})$ in each of these two cases:

***Case 2.i)*** $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$: In this case, $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are only different at the nonzero element in $\bar{\mathbf{a}}_{n_o+1}$. It follows that $a_{n_o,l} = a_{n_o+1,l} = 0$, for $\forall l \neq l_o$. By Proposition 2, we conclude that $a_{n_o,l_o}$ is the only nonzero element in $\mathbf{a}_{n_o}$. From (11) and (12), we have $b_{l_o} a_{n_o,l_o} = 1$, and $n_o c_{l_o} a_{n_o,l_o} + (N - n_o) c_{l_o} a_{n_o+1,l_o} = M$. Solving these two equations, we have

$$a_{n_o,l_o} = \frac{1}{b_{l_o}}; \ a_{n_o+1,l_o} = \frac{1}{N - n_o} \left( \frac{M}{c_{l_o}} - \frac{n_o}{b_{l_o}} \right) \quad (18)$$

where the expressions of $b_{l_o}$ and $c_{l_o}$ are given below (10).

***Case 2.ii)*** $a_{n_o,l_1} > a_{n_o+1,l_1} = 0$, $l_1 \neq l_o$: In this case, $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_o+1}$ are different at one of the zero elements in $\bar{\mathbf{a}}_{n_o+1}$. It follows that $a_{n_o,l_o} = a_{n_o+1,l_o} > 0$. By Proposition 2, we conclude that $\mathbf{a}_{n_o}$ has two nonzero elements: $a_{n_o,l_o}$ and $a_{n_o,l_1}$. Also recall $a_{n_o+1,0} > 0$; thus $\mathbf{a}_{n_o+1}$ has two nonzero elements: $a_{n_o+1,0}$ and $a_{n_o+1,l_o}$ (shown in Fig. 1 as colored elements). For the rest, we have $a_{n_o,l} = 0$, $\forall l \neq l_o$ or $l_1, l \in \mathcal{K} \cup \{0\}$, and $a_{n_o+1,l} = 0$, $\forall l \neq 0, l_o$, $l \in \mathcal{K} \cup \{0\}$ (shown

in Fig. 1 as the uncolored elements). With the two nonzero elements in $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$, and from (11), we have $b_{l_o}a_{n_o,l_o}+b_{l_1}a_{n_o,l_1}=1$, and $b_0 a_{n_o+1,0}+b_{l_o}a_{n_o,l_o}=1$. Also, from (12), we have $Nc_{l_o}a_{n_o,l_o}+n_o c_{l_1}a_{n_o,l_1}=M$. Solving there three questions, we obtain

$$a_{n_o,l_o}=\frac{b_{l_1}M-n_o c_{l_1}}{b_{l_1}Nc_{l_o}-b_{l_o}n_o c_{l_1}},\quad a_{n_o,l_1}=\frac{b_{l_o}M-Nc_{l_o}}{b_{l_o}n_o c_{l_1}-b_{l_1}Nc_{l_o}} \tag{19}$$

$$a_{n_o+1,0}=1-b_{l_o}a_{n,l_o}. \tag{20}$$

For given $(n_o,l_o,l_1)$, (18) in Case 2.i) (when $l_1=l_o$), or (19) and (20) in Case 2.ii) (when $l_1\neq l_o$) completely determines $\mathbf{a}_{n_o}$ and $\mathbf{a}_{n_o+1}$, and thus all $\{\mathbf{a}_n\}$ by (16). As a result, the average rate $\bar{R}$ in **P1** is a function of $(n_o,l_o,l_1)$. We can search over all possible values of $n_o\in\{1,\ldots,N-1\}$ and $l_o,l_1\in\mathcal{K}$ for the best tuple $(n_o,l_o,l_1)$ that gives minimum $\bar{R}$.

Algorithm 2 summarizes the steps to obtain the best solution $\{\mathbf{a}_n\}$ in the two-file-group case with $\bar{\mathbf{a}}_{n_o+1}\succcurlyeq_1\mathbf{0}$. The algorithm computes $\bar{R}$ using a closed-form expression for $(N-1)K^2$ times with different $(n_o,l_o,l_1)$, which can be done in parallel. Thus, the complexity of the algorithm is very low.

**Algorithm 2** The cache placement for two file groups with $\bar{\mathbf{a}}_{n_o+1}\succcurlyeq_1\mathbf{0}$

**Input:** $K$, $M$, $N$, and $\mathbf{p}$
**Output:** $(\bar{R}_{\min},n_o^*,l_o^*,l_1^*)$
For each $(n_o,l_o,l_1)$, $n_o\in[1,N-1]$, $l_o,l_1\in[1,K]$:
  **if** $l_1==l_o$, compute $\{\mathbf{a}_n\}$ by (18);
  **else** compute $\{\mathbf{a}_n\}$ by (19) and (20).
  Compute $\bar{R}(n_o,l_o,l_1)$ by (9).
Compute $(n_o^*,l_o^*,l_1^*)=\text{argmin}_{n_o,l_o,l_1}\bar{R}(n_o,l_o,l_1)$; $\bar{R}_{\min}=\bar{R}(n_o^*,l_o^*,l_1^*)$.

**Algorithm 3** The cache placement for three file groups

**Input:** $K$, $M$, $N$, and $\mathbf{p}$
**Output:** $(\bar{R}_{\min}, n_o^*, n_1^*, l_o^*, l_1^*)$
**for** $n_1=2$ to $N-1$ **do**
  $\bar{R}_1(n_o,n_1,l_o,l_1)=$
    Algorithm $2(K,M,n_1,[p_1,\ldots,p_{n_1}]^T)$;
  $\bar{R}_2(n_1)=\sum_{n=n_1+1}^{N}g_{n,0}$;
  Compute $\bar{R}(n_o,n_1,l_o,l_1)=\bar{R}_1+\bar{R}_2$
**end for**
Compute $(n_o^*,n_1^*,l_o^*,l_1^*)=\text{argmin}_{n_o,n_1,l_o,l_1}\bar{R}(n_o,n_1,l_o,l_1)$; $\bar{R}_{\min}=\bar{R}(n_o^*,n_1^*,l_o^*,l_1^*)$.

### C. Three File Groups

If there are three file groups under the optimal placement vectors $\{\mathbf{a}_n\}$ , the relation among $\mathbf{a}_n$'s is given by $\mathbf{a}_1=\ldots=\mathbf{a}_{n_o}\neq\mathbf{a}_{n_o+1}=\ldots=\mathbf{a}_{n_1}\neq\mathbf{a}_{n_1+1}=\ldots=\mathbf{a}_N$, for $1\leq n_o<n_1\leq N-1$. We use $\mathbf{a}_{n_o}$, $\mathbf{a}_{n_1}$ and $\mathbf{a}_{n_1+1}$ to represent the three unique cache placement vectors for the 1st, 2nd, and 3rd file group, respectively. We first present the cache placement $\mathbf{a}_{n_1+1}$ in the 3rd file group below. The proof uses the similar technique as outlined in Proposition 1 and is omitted.

**Proposition 4.** If there are three file groups under the optimal cache placement $\{\mathbf{a}_n\}$, the optimal placement vector $\mathbf{a}_{n_1+1}$ for the third file group is given by $\bar{\mathbf{a}}_{n_1+1}=\mathbf{0}$, and $a_{n_1+1,0}=1$.

Proposition 4 indicates that when there are three file groups under the optimal $\{\mathbf{a}_n\}$, all the cache will be allocated to the first two file groups, and the files in the 3rd file group are not cached and remain in the server. Following this, we only need to obtain the two unique cache placement vectors $(\mathbf{a}_{n_o},\mathbf{a}_{n_1})$ in the first two groups, respectively.

Note that since $\mathbf{a}_{n_1}\neq\mathbf{a}_{n_1+1}$, similar to (16), we have $\bar{\mathbf{a}}_{n_1}\succcurlyeq_1\bar{\mathbf{a}}_{n_1+1}=\mathbf{0}$ and $a_{n_1,0}<a_{n_1+1,0}=1$. Thus, the cache placement $(\mathbf{a}_{n_o},\mathbf{a}_{n_1})$ is the same as that of the two-file-group case with $\bar{\mathbf{a}}_{n_1}\succcurlyeq_1\mathbf{0}$ for the 2nd group in Section V-B2. For $\bar{\mathbf{a}}_{n_1}\succcurlyeq_1 0$, Propositions 1 and 3 indicate that $\bar{\mathbf{a}}_{n_1}$ has one nonzero element, and $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_1}$ are different by one element. Assume for some $l_o\in\mathcal{K}$, $a_{n_1,l_o}>0$. Between $\bar{\mathbf{a}}_{n_o}$ and $\bar{\mathbf{a}}_{n_1}$, the different element can be either at $l_o$: $a_{n_o,l_o}>a_{n_1,l_o}$, or at $l_1\neq l_o, l_1\in\mathcal{K}$: $a_{n_o,l_1}>a_{n_1,l_1}$ (the former is shown in Fig. 2). Detailed solution for $(\mathbf{a}_{n_o},\mathbf{a}_{n_1})$ can be obtained from Section V-B2.

Based on the above discussion, for the case of three file groups, given $(n_o,n_1,l_o,l_1)$, the solution $\{\mathbf{a}_n\}$ can be determined, and the average rate $\bar{R}$ in **P1** can be computed, as a function of $(n_o,n_1,l_o,l_1)$. Again, we can search over all possible values of $n_1\in\{2,N-1\}$, $n_o\in\{1,\ldots,n_1-1\}$, and $l_o,l_1\in\mathcal{K}$ for the best tuple $(n_o,n_1,l_o,l_1)$ that gives minimum $\bar{R}$. Algorithm 3 summarizes the steps to obtain the best solution $\{\mathbf{a}_n\}$ in the three file groups. It uses Algorithm 2 to obtain the best $(n_o,l_o,l_1)$ in the two-file-group subproblem, for each $n_1\in\{2,\ldots,N-1\}$. Thus, the algorithm simply computes $\bar{R}$ using a closed-form expression for $(N-1)(N-2)K^2/2$ times with different $(n_o,n_1,l_o,l_1)$, which can be done very efficiently.

### D. The Optimal Cache Placement Solution

The results on the possible structure of the optimal cache placement in Sections V-A to V-C lead to a simple method to obtain the optimal solution for **P1**. By Theorem 1, the optimal cache placement problem **P1** is reduced to three subproblems, assuming one to three file groups, respectively. Each subproblem returns the candidate optimal solution $\{\mathbf{a}_n\}$ with the minimum $\bar{R}$ for this subproblem. The optimal $\{\mathbf{a}_n\}$ can then be obtained by taking the one gives the minimum $\bar{R}$ among the three subproblems. The algorithm calls Algorithms 1–3 and selects the one that returns the minimum $\bar{R}$ as the optimal solution. It requires minimum complexity and can be done efficiently. Algorithms 1–3 each involves computing a closed-form expression of $\bar{R}$ for multiple times. In total, $\bar{R}$ is computed for $(N-1)(3N-4)K^2/2+N$ times, all can be done in parallel.

**Remark 1.** We provide some insights on Theorem 1 based on the results obtained. We can correspond three file groups to groups of "most popular," "moderately popular," and "not popular" files. Depending on $\mathbf{p}$ and relative cache size $M$ to $N$, files can be grouped into one of these three categories.

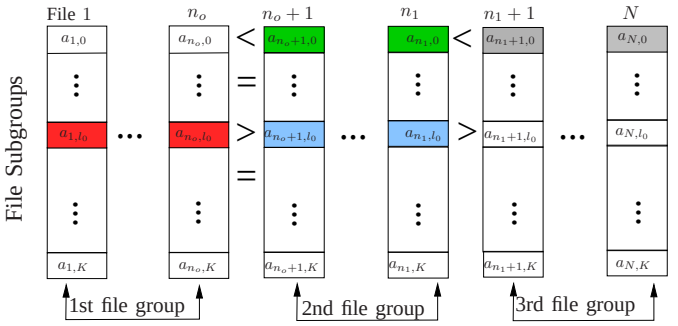


Fig. 2. An example of the optimal cache placement $\{\mathbf{a}_n\}$ in the case of three file groups. No cache is allocated to the 3rd file group: $a_{n_1+1,0} = 1$. For $\mathbf{a}_{n_o}, \mathbf{a}_{n_o+1}$ in the 1st and 2nd groups: $1 > a_{n_o+1,0} > a_{n_o,0} = 0$; $a_{n_o,l_o} > a_{n_o+1,l_o} > 0$, $l_o \in \mathcal{K}$; $a_{n_o,l} = a_{n_o+1,l} = 0$, $\forall l \in \mathcal{K}, l \neq l_o$.

Based on the structure of optimal $\{\mathbf{a}_n\}$, the optimal caching strategy is to cache all subfiles of the "most popular" files (into $K$ users); if there are "moderately popular" files, then a portion of subfiles in these files are cached, and some are left solely in the server; and if there are "not popular" files, they are all stored in the server uncached.

Sections V-A to V-C provide the possible structure of the optimal cache placement in three file grouping cases. However, analytically determining the optimal file grouping (*i.e.*, the number of file groups and $(n_o, n_1)$) is challenging, the same for $(l_o, l_1)$ for the nonzero element(s) in $\mathbf{a}_n$. It depends on the file popularity $\mathbf{p}$, as well as the relative cache capacity w.r.t. database (*i.e.*, $M$ vs. $N$). Nonetheless, using the obtained file group structures, Algorithms 1–3 significantly simplify the solving of **P1**, by providing a set of candidate solutions in closed-form to compare.

## VI. Numerical Results

We first verify the optimal cache placement solution structure of the MCCS. We generate file popularity using Zipf distribution given by $p_n = n^{-\theta}/\sum_{i=1}^{N} i^{-\theta}$. We obtain the solution $\{\mathbf{a}_n\}$ produced by our proposed algorithm, and verify that they match the optimal $\{\mathbf{a}_n\}$ obtained by solving **P1** numerically. As examples, for $K = 5$, $N = 9$, $\theta = 1.5$, Tables I and II show the optimal $\{\mathbf{a}_n\}$ (obtained numerically) for $M = 1$ and 3.5, respectively. In Table I ($M = 1$), we see two file groups $\{W_1, W_2\}$ and $\{W_3, \ldots, W_9\}$ under the optimal $\{\mathbf{a}_n\}$. It matches the case discussed in Section V-B1, where the cache is entirely allocated to the 1st group of the two most popular files, and the rest files are only stored at the server ($a_{3,0} = \ldots = a_{9,0} = 1$). The optimal $\mathbf{a}_n$'s for the 1st group are the same with two nonzero adjacent elements, and files are split into subfiles of two different sizes cached by users. This is the case with small cache size, and intuitively, only a few popular files are cached, and the rest remain in the server, and therefore there are two file groups under the optimal cache placement. Table II ($M = 3.5$) shows a different cache placement strategy, where the files are divided into three file groups, resembling the example shown in Fig. 2 (Section V-B2), where no cache is allocated to the 3rd file group $\{W_6, W_7, W_8, W_9\}$, a portion of the file is cached for $W_5$ in the 2nd file group, and for the 1st file group, files, split into subfiles of a single size, are all cached to users. This echoes the intuition in Remark 1.

TABLE I
THE CACHE PLACEMENT $\{\mathbf{a}_n\}$ FOR $K = 5$, $N = 9$, $M = 1$, $\theta = 1.2$ (ONLY $l = 0, 2, 3$ ARE SHOWN. FOR $l = 1, 4, 5$, $a_{n,l} = 0$, $\forall n \in \mathcal{N}$).

| $l$ | Cache placement vectors of the files | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{a}_7$ | $\mathbf{a}_8$ | $\mathbf{a}_9$ |
| 0 | 0 | 0 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 2 | 0.0500 | 0.0500 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0.0500 | 0.0500 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

TABLE II
THE CACHE PLACEMENT $\{\mathbf{a}_n\}$ FOR $K = 5$, $N = 9$ ,$M = 3.5$, $\theta = 1.2$ (ONLY $l = 0, 4$ ARE SHOWN. FOR $l = 1, 2, 3, 5$, $a_{n,l} = 0$, $\forall n \in \mathcal{N}$).

| $l$ | Cache placement vectors of the files | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mathbf{a}_1$ | $\mathbf{a}_2$ | $\mathbf{a}_3$ | $\mathbf{a}_4$ | $\mathbf{a}_5$ | $\mathbf{a}_6$ | $\mathbf{a}_7$ | $\mathbf{a}_8$ | $\mathbf{a}_9$ |
| 0 | 0 | 0 | 0 | 0 | 0.6250 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 4 | 0.2000 | 0.2000 | 0.2000 | 0.2000 | 0.0750 | 0 | 0 | 0 | 0 |

Next, we study the performance of the MCCS under the optimal cache placement. For comparison, we consider the multiple-file-group placement strategy proposed for the CCS [2] and directly apply it to the MCCS (Multigroup MCCS), centralized and decentralized single group symmetric placement for the MCCS [10] (one-group centralized/decentralized MCCS), and the existing strategies for the CCS, including the optimal CCS, a two-file-group scheme named RLFU-GCC (two-group) [14], and the mixed caching strategy in [15]. Fig. 3 shows the average rate $\bar{R}$ vs. $M$ by the optimal cache placement for MCCS (optimal MCCS), for Zipf distribution with $\theta = 1.5$, $N = 10$, and $K = 5$. In Fig. 4, we consider the case studied in [15] with $N = 12$, $K = 5$, and a step-function file popularity distribution: $p_1 = 7/12$, $p_n = 1/18$, $n = 2, \ldots, 7$, and $p_n = 1/60$, $n = 8, \ldots, 12$. Both Figs. 3 and 4 show that the optimal placement for the MCCS gives the lowest $\bar{R}$ among all the strategies, for all values of $M$. In particular, the performance gap is large when $M$ is small, and the gap reduces as $M$ becomes large. This trend is because for small cache size, the performance is more sensitive to the cache placement for better coded caching gain.

## VII. Conclusion

This paper aims at obtaining the optimal cache placement and its structure for the MCCS under nonuniform file popularity. Using an optimization approach, we characterized the inherent file grouping structure under the optimal placement and show there are at most three file groups regardless of system configurations. We completely characterized the cache placement solution in each file grouping case in closed-form, and then developed a simple and efficient algorithm to obtain the optimal cache placement solution via a set of candidate solutions. Insights into the structure of the optimal solution were also given. Simulation studies verified the optimal cache placement solution structure and showed superior performance over existing schemes for both MCCS and CCS.

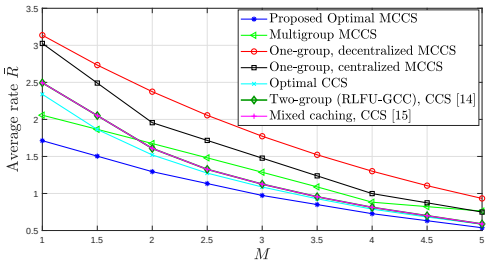


Fig. 3. $\bar{R}$ vs. cache size $M$ ($N = 10$, $K = 5$, Zipf distribution $\theta = 1.5$).

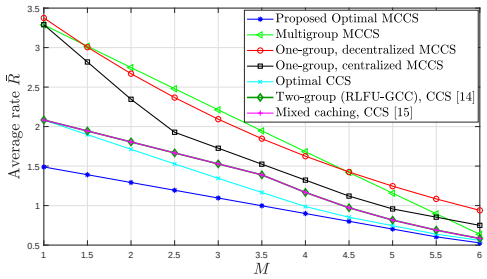


Fig. 4. $\bar{R}$ vs. cache size $M$ ($N = 12$,$K = 5$, step function popularity).

## APPENDIX A
## PROOF OF THEOREM 1

*Proof:* The Lagrangian function of **P1** is given by $L = \sum_{n=1}^{N} \mathbf{g}_n^T \mathbf{a}_n - \sum_{n=1}^{N-1}\sum_{l=1}^{K} \gamma_{n,l}(a_{n,l} - a_{n+1,l}) - \sum_{l=1}^{K} \rho_l a_{N,l} - \rho_0 a_{1,0} + \lambda(\sum_{n=1}^{N} \mathbf{c}^T \mathbf{a}_n - M) + \sum_{n=1}^{N} \nu_n(\mathbf{b}^T \mathbf{a}_n - 1)$, where we have the Lagrange multipliers $\{\gamma_{n,k}\}$ for (4), $\rho_0$ and $\{\rho_k\}$ for (7) , $\{\nu_n\}$ for (11), and $\lambda$ for (12). Since **P1** is an LP, the Karush-Kuhn-Tucker (KKT) conditions for **P1** hold. Due to space, we only list the KKT conditions we need in the proof below:

$$\gamma_{n,l}(a_{n,l} - a_{n+1,l}) = 0, \gamma_{n,l} \geq 0, \quad n \in \mathcal{N}\backslash\{N\}, l \in \mathcal{K}, \quad (21)$$

$$\frac{\partial L}{\partial a_{n,l}} = g_{n,l} - \gamma_{n,l} + \gamma_{n-1,l} + \lambda c_l + \nu_n b_l = 0, n \in \mathcal{N}\backslash\{1, N\}, l \in \mathcal{K} \quad (22)$$

$$\frac{\partial L}{\partial a_{1,l}} = g_{1,l} - \gamma_{1,l} + \lambda c_l + \nu_1 b_l = 0, l \in \mathcal{K} \quad (23)$$

$$\frac{\partial L}{\partial a_{n,0}} = g_{n,0} + \lambda c_0 + \nu_n b_0 = 0, \quad n \in \mathcal{N}\backslash\{1\}. \quad (24)$$

From (22) and (23), we have

$$\sum_{i=1}^{m} \frac{\partial L}{\partial a_{i,l}} = \sum_{i=1}^{m} g_{i,l} - \gamma_{m,l} + m\lambda c_l + \sum_{i=1}^{m} \nu_i b_l = 0. \quad (25)$$

Using these KKT conditions, we prove Theorem 1 by contradiction. Assume there is an optimal solution $\{\mathbf{a}_n\}$ that divides the files into four file groups. The structure of the sub-placement vectors $\bar{\mathbf{a}}_n$'s can be expressed as $\bar{\mathbf{a}}_1 = \ldots = \bar{\mathbf{a}}_{n_o} \succcurlyeq_1 \bar{\mathbf{a}}_{n_o+1} = \ldots = \bar{\mathbf{a}}_{n_1} \succcurlyeq_1 \bar{\mathbf{a}}_{n_1+1} = \ldots = \bar{\mathbf{a}}_{n_2} \succcurlyeq_1 \bar{\mathbf{a}}_{n_2+1} = \ldots = \bar{\mathbf{a}}_N$, for $1 \leq n_o < n_1 < n_2 \leq N - 1$. By (4), we assume $a_{n_o,l_o} > a_{n_o+1,l_o}$, $a_{n_1,l_1} > a_{n_1+1,l_1}$ and $a_{n_2,l_2} > a_{n_2+1,l_2}$. From (21) and (24) with $c_0 = 0$ and $b_0 = 1$, we have

$$\gamma_{n_o,l_o} = \gamma_{n_1,l_1} = \gamma_{n_2,l_2} = 0; \ \nu_n = -g_{n,0}, \ n \in \mathcal{N}\backslash\{1\}. \quad (26)$$

Based on (25), we have the following equation

$$\sum_{i=1}^{n_j} g_{i,l_j} - \gamma_{n_j,l_j} + n_j\lambda c_{l_j} + \sum_{i=1}^{n_j} \nu_i b_{l_j} = 0, \quad j = 0, 1, 2. \quad (27)$$

Substituting the values of $\gamma_{n_o,l_o}$, $\gamma_{n_1,l_1}$, $\gamma_{n_2,l_2}$ and $\nu_n$ in (26) into (27), we have

$$\lambda n_j c_{l_j} + \nu_1 b_{l_j} = -\sum_{i=2}^{n_j} g_{i,0} b_{l_j} - \sum_{i=1}^{n_j} g_{i,l_j}, \quad j = 0, 1, 2. \quad (28)$$

We can stack (28) for $j = 0, 1, 2$ in matrix form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is the coefficient matrix, $\mathbf{x} = [\lambda, \nu_1]^T$, and consists of right hand side of (28) for $j = 0, 1, 2$. Since $[\mathbf{A}, \mathbf{b}]$ is full rank, there is no feasible solution for $\lambda, \nu_1$, contradicting the assumption that an optimal $\{\mathbf{a}_n\}$ with four file groups exists. Similar argument follows for more than four file groups. Thus, we have the result in Theorem 1. ■

## REFERENCES


[1] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Commun. Mag.*, vol. 52, pp. 82–89, Aug. 2014.

[2] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inform. Theory*, vol. 63, pp. 1146–1158, Dec. 2017.

[3] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inform. Theory*, vol. 60, pp. 2856–2867, Mar. 2014.

[4] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, pp. 1029–1040, Aug. 2015.

[5] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 836–845, Apr. 2016.

[6] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Trans. Inform. Theory*, vol. 62, pp. 7253–7271, Dec. 2016.

[7] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Trans. Inform. Theory*, vol. 62, pp. 3212–3229, Jun. 2016.

[8] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inform. Theory*, vol. 62, pp. 849–869, Feb. 2016.

[9] F. Xu, M. Tao, and K. Liu, "Fundamental tradeoff between storage and latency in cache-aided wireless interference networks," *IEEE Trans. Inform. Theory*, vol. 63, pp. 7464–7491, Jun. 2017.

[10] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inform. Theory*, vol. 64, pp. 1281–1296, Feb. 2018.

[11] Ç. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of D2D coded caching with uncoded cache placement and one-shot delivery," *IEEE Trans. Commun.*, vol. 67, pp. 8179–8192, Dec. 2019.

[12] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *IEEE Trans. Inform. Theory*, vol. 66, pp. 1893–1919, Mar. 2020.

[13] S. Jin, Y. Cui, H. Liu, and G. Caire, "Structural properties of uncoded placement optimization for coded delivery," *arXiv preprint arXiv:1707.07146*, Jul. 2017.

[14] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Trans. Inform. Theory*, vol. 63, pp. 3923–3949, Apr. 2017.

[15] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Trans. Inform. Theory*, vol. 64, pp. 349–366, Nov. 2018.

[16] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inform. Theory*, vol. 63, pp. 3108–3141, Mar. 2017.

[17] S. Jin, Y. Cui, H. Liu, and G. Caire, "Uncoded placement optimization for coded delivery," *arXiv preprint arXiv:1709.06462*, Jul. 2018.

[18] J. Riordan, *Introduction to combinatorial analysis*. Courier Corporation, 2012.

[19] Y. Deng and M. Dong, "Optimal cache placement for modified coded caching with arbitrary cache size," in *Proc. IEEE SPAWC*, Jul. 2019.