On Coded Caching with Correlated Files

Kai Wan*, Daniela Tuninetti†, Mingyue Ji‡, Giuseppe Caire*,
*Technische Universität Berlin, Berlin, Germany, {kai.wan, caire}@tu-berlin.de
†University of Illinois at Chicago, Chicago, USA, danielat@uic.edu
‡University of Utah, Salt Lake City, USA, mingyue.ji@utah.edu

***Abstract*—This paper studies the fundamental limits of the shared-link caching problem with correlated files, where a server with a library of N files communicates with K users who can store M files. Given an integer $\mathsf{r} \in [\mathsf{N}]$, correlation is modelled as follows: each $\mathsf{r}-$subset of files contains one and one only common block. The tradeoff between the cache size and the average transmitted load is considered. First, a converse bound under the constraint of uncoded cache placement (i.e., each user directly caches a subset of the library bits) is derived. Then, an interference alignment scheme is proposed. The proposed scheme achieves the optimal average load under uncoded cache placement to within a factor of 2 in general, and it is exactly optimal for (i) users demand distinct files, (ii) large or small cache size, namely $\mathsf{KrM/N} \leq 2$ or $\mathsf{KrM/N} \geq \mathsf{K}-1$, and (iii) large or small correlation, namely $\mathsf{r} \in \{1, 2, \mathsf{N}-1, \mathsf{N}\}$. As a by-product, the proposed scheme reduces the (worst-case or average) load of existing schemes for the caching problem with multi-requests.**

I. Introduction

Cache is a network component that leverages the device memory to transparently store data so that future requests for that data can be served faster. Two phases are included in a caching system: i) cache placement phase: content is pushed into each cache without knowledge of future demands; ii) delivery phase: after each user has made its request and according to the cache contents, the server transmits coded packets in order to satisfy the user demands. The goal is to minimize the number of transmitted bits (or load or rate).

Coded caching was proposed by Maddah-Ali and Niesen (MAN) in [1] for a shared-link caching systems containing a server with a library of N equal-length files, which is connected to K users through a noiseless shared-link, each of which can store M files in its cache. Each user demands one file in the delivery phase. The MAN scheme uses a combinatorial design in the placement phase so that during the delivery phase multicast messages simultaneously satisfy the demands of different users. Under the constraint of uncoded cache placement the MAN scheme was first proved to be optimal the worst-case load when $\mathsf{N} \geq \mathsf{K}$ [2], and later in [3] for any (N, K) and any kind of demands, as well as for average load under the uniform demand distribution.

The above works assume that the N files in the library are independent. However, in practice overlaps among different files is possible (e.g., videos, image streams, etc.). Coded caching with correlated files was considered in [4], where each subset of files has an exclusively common part; a caching scheme for two-file $\mathsf{K}-$user system, and for three-file two-user system, was proved to be near-optimal for worst-case demands. In [5] the caching problem with correlated files, where the length of the common part among each ℓ files (referred to as a 'ℓ-block') is the same, was considered; each file contains $\binom{\mathsf{N}-1}{\ell-1}$ $\ell-$blocks. The achievable scheme in [5] contains N steps, and in step ℓ only ℓ-blocks are transmitted; there are $\binom{\mathsf{N}-1}{\ell-1}$ rounds for the transmission of step ℓ, where each round is treated as a MAN caching problem.

The caching problem with correlated files is a special case of the caching problem with multi-requests considered in [6], where each user demands L files from the library. If the problem is divided into L rounds, where in each round the MAN scheme in [1] is used to let each user decode one file, one can show the order optimality to within factors of 18 [6] or 11 [7]. Instead of using the MAN scheme in each round, one could use the scheme in [3] to leverage the multicast opportunities, as done in [8]. There are two main limitations in dividing the delivery into L rounds and use in each round a caching scheme designed for single-request caching problem: (1) a file may exist in different rounds and this round-division method may lose some multicast opportunities, and (2) finding the best division of the users' demands into L groups is hard.

Contributions and Paper Organization: In this paper, we consider a simplification of the model in [5]: we fix $\mathsf{r} \in [\mathsf{N}]$ and assume each file only contains r-blocks (see Section II); In Section III we derive a converse bound on the minimal average load for any demand type under the constraint of uncoded cache placement, by leveraging known index coding converse bounds [9] as we pioneered in [2]. In Section IV we propose a novel *interference alignment* scheme for the caching problem with correlated files which jointly serves users' multi-demands (instead of the round-division method). The proposed scheme achieves the optimal average load among all demands with distinct requests under the constraint of uncoded cache placement. For general demands, the scheme achieves the optimal average load under the constraint of uncoded cache placement for $\mathsf{KrM} \leq 2\mathsf{N}$ or $\mathsf{KrM} \geq (\mathsf{K}-1)\mathsf{N}$ or $\mathsf{r} \in \{1, 2, \mathsf{N}-1, \mathsf{N}\}$. Our scheme can be also used to improve the state-of-the-art for the caching problem with multi-requests [8].

Notation Convention: Calligraphic symbols denote sets, bold symbols denote vectors, and sans-serif symbols denote system parameters. We use $|\cdot|$ to represent the cardinality of a set or the length of a vector; $[a:b] := \{a, a+1, \ldots, b\}$ and $[n] := [1, 2, \ldots, n]$; $\oplus$ represents bit-wise XOR.

II. System Model

In an $(\mathsf{N}, \mathsf{K}, \mathsf{M}, \mathsf{r})$ shared-link caching problem with correlated files, a server has access to a library of N files (each file

has B bits) denoted by $\{F_1, \cdots, F_{\mathsf{N}}\}$. The server is connected to K users through an error-free link. Each file F_i where $i \in [\mathsf{N}]$, is composed of $\binom{\mathsf{N}-1}{\mathsf{r}-1}$ independent and equal-length blocks, $F_i = \{W_{\mathcal{S}} : \mathcal{S} \subseteq [\mathsf{N}], |\mathcal{S}| = \mathsf{r}, i \in \mathcal{S}\}$, where the block $W_{\mathcal{S}}$ represents the exclusive common part of all files in $\mathcal{S}$. So in the library, there are $\binom{\mathsf{N}}{\mathsf{r}}$ independent blocks, each of which has $\mathsf{B}/\binom{\mathsf{N}-1}{\mathsf{r}-1}$ bits. During the cache placement phase, user $k \in [\mathsf{K}]$ stores information about the N files in its cache of size MB bits, where $\mathsf{M} \in [0, \mathsf{N}/\mathsf{r}]$. Denote the content in the cache of user $k \in [\mathsf{K}]$ by Z_k and let $\mathbf{Z} := (Z_1, \ldots, Z_{\mathsf{K}})$. During the delivery phase, user $k \in [\mathsf{K}]$ independently demands file $d_k \in [\mathsf{N}]$. The demand vector $\mathbf{d} := (d_1, \ldots, d_{\mathsf{K}})$ is revealed to all nodes. Given $(\mathbf{d}, \mathbf{Z})$, the server broadcasts a message $X(\mathbf{d}, \mathbf{Z})$ of $\mathsf{B}\mathsf{R}(\mathbf{d}, \mathbf{Z})$ bits to all users. User $k \in [\mathsf{K}]$ must recover its desired file F_{d_k} from Z_k and $X(\mathbf{d}, \mathbf{Z})$.

A demand vector $\mathbf{d}$ is said to be of type $\mathcal{D}_{N_{\mathrm{e}}(\mathbf{d})}$ if it has $N_{\mathrm{e}}(\mathbf{d}) := |\{d_k : k \in [\mathsf{K}]\}|$ distinct entries. Based on the uniform demand distribution, the objective is to determine the optimal average load among all demands of the same type

$$\mathsf{R}^\star(\mathsf{M}, s) := \min_{\mathbf{Z}} \ \mathbb{E}_{\mathbf{d} \in \mathcal{D}_s}[\mathsf{R}(\mathbf{d}, \mathbf{Z})], \ \forall s \in [\min\{\mathsf{K}, \mathsf{N}\}], \quad (1)$$

and the optimal average load among all possible demands

$$\mathsf{R}^\star(\mathsf{M}) := \min_{\mathbf{Z}} \ \mathbb{E}_{\mathbf{d} \in [\mathsf{N}]^{\mathsf{K}}}[\mathsf{R}(\mathbf{d}, \mathbf{Z})]. \quad (2)$$

The cache placement is *uncoded* if each user directly copies some bits into its cache. Under the constraint of uncoded cache placement, we divide each block $W_{\mathcal{S}}$ where $\mathcal{S} \subseteq [\mathsf{N}]$ and $|\mathcal{S}| = \mathsf{r}$ into sub-blocks, $W_{\mathcal{S}} = \{W_{\mathcal{S},\mathcal{V}} : \mathcal{V} \subseteq [\mathsf{K}]\}$, where $W_{\mathcal{S},\mathcal{V}}$ represents the bits of $W_{\mathcal{S}}$ which are exclusively cached by users in $\mathcal{V}$. The optimal loads under uncoded cache placement $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M}, s)$ and $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M})$ are defined as in (1) and (2), respectively.

Relation to the Coded Caching Problem with Multiple Requests: If we identify the $\binom{\mathsf{N}}{\mathsf{r}}$ independent blocks as files of the library, and allow each cache-equipped user to request $\binom{\mathsf{N}-1}{\mathsf{r}-1}$ such blocks/files, the considered caching problem with correlated sources can be thought of as the *symmetric* caching problem with multiple requests considered in [6], where 'symmetric' means that each user requests the same number of files. There is a subtle difference between our model and the one in [6]: in our model one file corresponds to $\binom{\mathsf{N}}{\mathsf{r}}$ *distinct* blocks, thus our model corresponds to the one in [6] under the constraint that a user has multiple but distinct requests. Moreover, we consider the average load as our performance metric, while in [6] the worst-case load was considered.

III. Main Results

Theorem 1 (Converse)**.** *For an* $(\mathsf{N}, \mathsf{K}, \mathsf{M}, \mathsf{r})$ *shared-link caching problem with correlated files,* $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M}, s)$ *is lower bounded by the lower convex envelope of the following memory-load pairs*

$$(\mathsf{M}, \mathsf{R}_{\mathrm{u}}) = (\mathsf{N}t/\mathsf{K}, c_t^s), \ \forall t \in [0 : \mathsf{K}], \quad (3)$$

for all $s \in [\min\{\mathsf{K}, \mathsf{N}\}]$*, where*

$$c_t^s := \frac{\sum_{j \in [\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, s\}]} \binom{\mathsf{N}-j}{\mathsf{r}-1}\binom{\mathsf{K}-j}{t}}{\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}}. \quad (4)$$

In addition, $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M})$ *is lower bounded by the lower convex envelope of the following memory-load pairs*

$$(\mathsf{M}, \mathsf{R}_{\mathrm{u}}) = \left(\mathsf{N}t/\mathsf{K}, \mathbb{E}_{\mathbf{d} \in [\mathsf{N}]^{\mathsf{K}}}[c_t^{N_{\mathrm{e}}(\mathbf{d})}]\right), \ \forall t \in [0 : \mathsf{K}]. \quad (5)$$

Proof: Inspired by [10], we use the "acyclic index coding converse bound" from [9]. For a demand vector $\mathbf{d}$ demand of type $\mathcal{D}_s$ where $s \in [\min\{\mathsf{K}, \mathsf{N}\}]$, choose $s = N_{\mathrm{e}}(\mathbf{d})$ users with distinct demands. Generate a directed graph for the delivery phase, where each sub-block demanded by each of these $N_{\mathrm{e}}(\mathbf{d})$ users represents one node in the graph. There is a directed edge from node i to node j if the user demanding the sub-block represented by node j caches the sub-block represented by node i. Consider a permutation of these $N_{\mathrm{e}}(\mathbf{d})$ users, denoted by $\mathbf{u} = (u_1, u_2, ..., u_{N_{\mathrm{e}}(\mathbf{d})})$. By [2, Lemma 1], we can prove that the set of sub-blocks

$$\bigcup_{k \in [N_{\mathrm{e}}(\mathbf{d})]} \ \bigcup_{\substack{\mathcal{S} \subseteq [\mathsf{N}] \setminus \{d_{u_1}, \ldots, d_{u_{k-1}}\} \\ : |\mathcal{S}| = \mathsf{r}, d_{u_k} \in \mathcal{S}}} \ \bigcup_{\mathcal{V} \subseteq [\mathsf{K}] \setminus \{u_1, \ldots, u_k\}} W_{\mathcal{S},\mathcal{V}},$$

does not contain a directed cycle. By the "acyclic index coding converse bound", the number of transmitted bits is not less than total number of bits of the sub-blocks in this set. Consider all the demands of type $s \in [\min\{\mathsf{K}, \mathsf{N}\}]$, all sets of users with different s distinct demands, and all permutations of those users; by summing all the resulting "acyclic index coding converse bound" inequalities, we obtain

$$\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M}, s) \geq \sum_{t=0}^{K} c_t^s x_t, \quad (6a)$$

$$x_0 + x_1 + ... + x_{\mathsf{K}} = 1, \quad (6b)$$

$$x_1 + 2x_2 + ... + t x_t + ... + \mathsf{K} x_{\mathsf{K}} \leq \mathsf{K}\mathsf{M}\mathsf{r}/\mathsf{N}, \quad (6c)$$

$$x_t := \sum_{\mathcal{S} \subseteq [\mathsf{N}] : |\mathcal{S}| = \mathsf{r}} \ \sum_{\mathcal{V} \subseteq [\mathsf{K}] : |\mathcal{V}| = t} \mathsf{r}|W_{\mathcal{S},\mathcal{V}}|/(\mathsf{N}\mathsf{B}) \quad (6d)$$

where x_t represent the fraction of all bits cached exactly by t users. After Fourier-Motzkin elimination of the $\{x_t\}$ as in [10], one obtains the converse bound in (3). ∎

We next propose a multi-round interference alignment scheme (details in Section IV) which is optimal for the three cases described in Theorem 2. Different from existing round-division methods in [5]–[8], our scheme is designed to jointly serve users' multi-demands. The main ingredients of the scheme are as follows. We pick a leader user among the users demanding the same file; we then divide the delivery phase into steps, and in each step we satisfy the demand of one leader user after the construction of multicast messages destined for all steps, each of the unsatisfied user can cancel (or align) all non-intended 'symbols' (interferences) in all multicast messages which are useful to it.

Theorem 2 (Optimality)**.** *For an* $(\mathsf{N}, \mathsf{K}, \mathsf{M}, \mathsf{r})$ *shared-link caching problem with correlated files, we have*

1) *Case 1: When* $\mathsf{N} \geq \mathsf{K}$*,* $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M}, \mathsf{K})$ *is equal to the lower convex envelops of* c_t^{K} *where* $t \in [0 : \mathsf{K}]$*.*
2) *Case 2: When* $\mathsf{r} \in \{1, 2, \mathsf{N}-1, \mathsf{N}\}$*,* $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M}, s)$ *where* $s \in [\min\{\mathsf{K}, \mathsf{N}\}]$ *and* $\mathsf{R}^\star_{\mathrm{u}}(\mathsf{M})$ *are equal to the lower convex*

envelops of c_t^s *and of* $\mathbb{E}_{\mathbf{d}\in[\mathsf{N}]^{\mathsf{K}}}[c_t^{N_{\mathrm{e}}(\mathbf{d})}]$ *where* $t \in [0:\mathsf{K}]$*, respectively.*

3) *Case 3: When* $\mathsf{M} \le 2\mathsf{N}/(\mathsf{Kr})$ *or* $\mathsf{M} \ge (\mathsf{K}-1)\mathsf{N}/(\mathsf{Kr})$*,* $\mathsf{R}_{\mathrm{u}}^{\star}(\mathsf{M}, s)$ *where* $s \in [\min\{\mathsf{K},\mathsf{N}\}]$ *and* $\mathsf{R}_{\mathrm{u}}^{\star}(\mathsf{M})$ *are equal to the lower convex envelops of* c_t^s *and of* $\mathbb{E}_{\mathbf{d}\in[\mathsf{N}]^{\mathsf{K}}}[c_t^{N_{\mathrm{e}}(\mathbf{d})}]$ *where* $t \in \{0,1,2,\mathsf{K}-1,\mathsf{K}\}$*, respectively.*

Proof: Comparing the converse bound in Theorem 1 and the achieved load of our scheme (given in the *Performance* paragraph of Section IV-B), we have the optimality for Cases 1 and 2. The optimality for Case 3 is due to the fact that $c_t^{N_{\mathrm{e}}(\mathbf{d})}$ is convex in terms of t, and when $t \in \{0,1,2,\mathsf{K}-1,\mathsf{K}\}$, our proposed scheme is optimal (more details in [11]). ■

Remark 1. We can extend our scheme to general case and derive an order optimality for any system parameters and any demand type to within a factor of 2 under the constraint of uncoded cache placement (more details in [11]).

Remark 2. For the caching problem with multi-requests considered in [8] where each user demands L uncorrelated and equal-length files, the scheme in [8] was proved to be optimal under the constraint of the MAN placement for most demands with $\mathsf{K} \le 4$ users, $\mathsf{M} = \mathsf{N}/\mathsf{K}$, and $\mathsf{L} = 2$, except one demand for $\mathsf{K} = 3$ and three demands for $\mathsf{K} = 4$. We can use the proposed scheme in this paper to achieve the optimality for those four unsolved cases (more details in [11]).

Hence, the proposed results in this paper also shed light in the very relevant and intricate problem of how to handle optimally the case, where each user makes a sequence of requests of independent files (blocks). The fact that there are repeated elements in such sequence of requests is a 'fundamental' aspect of caching (also in practice), where one needs to devise schemes that take advantage of previous requests and do not send the same stuff multiple times.

Remark 3. The proposed caching scheme characterizes the worst-case load under the constraint of uncoded cache placement by setting $s = \min\{\mathsf{K},\mathsf{N}\}$ in Theorem 2.

IV. Novel Interference Alignment Based Scheme

A. Example

Consider an $(\mathsf{N},\mathsf{K},\mathsf{M},\mathsf{r})$ shared-link caching problem with correlated files with $\mathsf{N} = 3$, $\mathsf{K} = 5$, $\mathsf{M} = 3/5$ and $\mathsf{r} = 2$. Here $\mathsf{M} = 2\mathsf{N}/(\mathsf{rK})$ as in Case 3 of Theorem 2. There are 3 blocks, $W_{\{1,2\}}, W_{\{1,3\}}, W_{\{2,3\}}$. The files are $F_1 = \{W_{\{1,2\}}, W_{\{1,3\}}\}$, $F_2 = \{W_{\{1,2\}}, W_{\{2,3\}}\}$ and $F_3 = \{W_{\{1,3\}}, W_{\{2,3\}}\}$.

Placement phase: We use the MAN placement, as naturally inspired by our converse bound. Here $t = \frac{\mathsf{KMr}}{\mathsf{N}} = 2$ so we partition each block into $\binom{\mathsf{K}}{t} = 10$ equal-length sub-blocks, $W_{\mathcal{S}} = \{W_{\mathcal{S},\mathcal{V}} : \mathcal{V} \subseteq [\mathsf{K}], |\mathcal{V}| = t\}$. User $k \in [\mathsf{K}]$ caches $W_{\mathcal{S},\mathcal{V}}$ for all $\mathcal{V} \subseteq [\mathsf{K}]$ of size $|\mathcal{V}| = t$ if $k \in \mathcal{V}$. Hence, each sub-block contains $\mathsf{B}/\left(\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}\right) = \mathsf{B}/20$ bits and each user caches $\mathsf{B}\binom{\mathsf{N}}{\mathsf{r}}\binom{\mathsf{K}-1}{t-1}/\left(\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}\right) = 3\mathsf{B}/5$ bits.

Delivery Phase: Assume $\mathbf{d} = (1,2,3,1,2)$, which has $N_{\mathrm{e}}(\mathbf{d}) = 3$ distinct demanded files. Pick one user demanding a distinct file, and refer to it as the "leader user" among those demanding the same file. Assume here that the set of leaders is $\{1,2,3\}$. Consider next a permutation of the leaders, say $(1,2,3)$. Our proposed delivery scheme contains $\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, N_{\mathrm{e}}(\mathbf{d})\} = 2$ steps; after Step i, the i^{th} element/user in the permutation can decode its desired file; after finishing all steps, the remaining users can also decode their desired file. We next describe the two steps in the delivery phase for this example, where each step we send multicast messages of the type for some $\mathcal{T} \subseteq [\mathsf{K}]$ with $|\mathcal{T}| = t+1$ and $\mathcal{H} \subseteq [\mathsf{N}]$,

$$C_{\mathcal{T},\mathcal{H}} := \bigoplus_{k\in\mathcal{T}} \bigoplus_{\substack{\mathcal{S}\subseteq\mathcal{N}(\mathcal{T})\cup\mathcal{H}:\\ |\mathcal{S}|=\mathsf{r},\mathcal{H}\subseteq\mathcal{S},d_k\in\mathcal{S}}} W_{\mathcal{S},\mathcal{T}\setminus\{k\}}, \tag{7}$$

where $\mathcal{N}(\mathcal{T}) := \cup_{k\in\mathcal{T}}\{d_k\}$ is the set of demanded files by the users in $\mathcal{T}$. In words, the multicast message $C_{\mathcal{T},\mathcal{H}}$ in (7) is the binary sum of each sub-block (from blocks $W_{\mathcal{S}}$ where $\mathcal{S} \subseteq \mathcal{N}(\mathcal{T}) \cup \mathcal{H}$) desired by one user in $\mathcal{T}$ and known by all the other users in $\mathcal{T}$. Note that, when $\mathsf{r} = 1$ (in which case our model reduces to the MAN caching system in [1]), $C_{\mathcal{T},\emptyset}$ in (7) is equivalent to the MAN multicast message.

Delivery Phase Step 1 (to satisfy leader user $u_1 = 1$*).* Each time we consider one set of users $\mathcal{J} \subseteq [\mathsf{K}]$ where $|\mathcal{J}| = t+1 = 3$ and $u_1 \in \mathcal{J}$ (recall that $u_1 = 1$, $d_{u_1} = 1$), and one set of files $\mathcal{B} \subseteq [\mathsf{N}] \setminus \{d_{u_1}\}$ where $|\mathcal{B}| = \mathsf{r}-1 = 1$.

Consider $\mathcal{J} = \{1,2,3\}$. For $\mathcal{B} = \{2\}$ we transmit

$$C_{\{1,2,3\},\{2\}} = W_{\{1,2\},\{2,3\}} \oplus W_{\{1,2\},\{1,3\}} \oplus W_{\{2,3\},\{1,3\}} \oplus W_{\{2,3\},\{1,2\}}. \tag{8}$$

and for $\mathcal{B} = \{3\}$ we transmit

$$C_{\{1,2,3\},\{3\}} = W_{\{1,3\},\{2,3\}} \oplus W_{\{1,3\},\{1,2\}} \oplus W_{\{2,3\},\{1,3\}} \oplus W_{\{2,3\},\{1,2\}}. \tag{9}$$

By leveraging its cached content, user 1 decodes $W_{\{1,2\},\{2,3\}}$ from $C_{\{1,2,3\},\{2\}}$, and $W_{\{1,3\},\{2,3\}}$ from $C_{\{1,2,3\},\{3\}}$. Similarly, user 2 decodes first $W_{\{2,3\},\{1,3\}}$ from $C_{\{1,2,3\},\{3\}}$, and then uses $W_{\{2,3\},\{1,3\}}$ and its cached content to recover $W_{\{1,2\},\{1,3\}}$ from $C_{\{1,2,3\},\{2\}}$. Similarly, user 3 decodes $W_{\{2,3\},\{1,2\}}$ and $W_{\{1,3\},\{1,2\}}$. In other words, for the transmitted $C_{\mathcal{J},\mathcal{B}}$'s, each user in $\mathcal{J}$ can eventually recover each not-cached sub-block in $C_{\mathcal{J},\mathcal{B}}$.

In addition, $C_{\mathcal{J},\mathcal{B}}$ is useful to other users whose demanded file is in $\mathcal{N}(\mathcal{J}) \cup \mathcal{B}$ (recal that $\mathcal{N}(\mathcal{J}) := \cup_{k\in\mathcal{J}}\{d_k\}$), e.g., $C_{\{1,2,3\},\{3\}}$ is useful to users 4 and 5, where $W_{\{2,3\},\{1,3\}} \oplus W_{\{2,3\},\{1,2\}}$ is an 'interference' to user 4, as well as $W_{\{1,3\},\{2,3\}} \oplus W_{\{1,3\},\{1,2\}}$ is an 'interference' to user 5.

We then focus on $\mathcal{J} = \{1,2,4\}$. For $\mathcal{B} = \{2\}$ we transmit

$$C_{\{1,2,4\},\{2\}} = W_{\{1,2\},\{2,4\}} \oplus W_{\{1,2\},\{1,4\}} \oplus W_{\{1,2\},\{1,2\}}, \tag{10}$$

and f $\mathcal{B} = \{3\}$ we transmit

$$C_{\{1,2,4\},\{3\}} = W_{\{1,3\},\{2,4\}} \oplus W_{\{1,3\},\{1,2\}} \oplus W_{\{2,3\},\{1,4\}}. \tag{11}$$

From $C_{\{1,2,4\},\{2\}}$, users $1,2,4$ can decode $W_{\{1,2\},\{2,4\}}$, $W_{\{1,2\},\{1,4\}}$, and $W_{\{1,2\},\{1,2\}}$, respectively. In addition, $C_{\{1,2,4\},\{2\}}$ is also useful to other user whose demanded

file is in $\mathcal{N}(\{1,2,4\}) \cup \mathcal{B} = \{1,2\}$, i.e., user 5. Meanwhile, the remaining users (user 3 demanding F_3), neglect $C_{\{1,2,4\},\{2\}}$. From $C_{\{1,2,4\},\{3\}}$, users $1,2,4$ can decode $W_{\{1,3\},\{2,4\}}$, $W_{\{2,3\},\{1,4\}}$, and $W_{\{1,3\},\{1,2\}}$, respectively. In addition, $C_{\{1,2,4\},\{3\}}$ is also useful to other users whose demanded file is in $\mathcal{N}(\{1,2,4\})\cup\mathcal{B} = \{1,2,3\}$, i.e., users $3,5$, where $W_{\{1,3\},\{2,4\}} \oplus W_{\{1,3\},\{1,2\}}$ is an interference to user 5.

Similarly, for $\mathcal{J} = \{1,2,5\}$, we transmit

$$C_{\{1,2,5\},\{2\}} = W_{\{1,2\},\{2,5\}} \oplus W_{\{1,2\},\{1,5\}} \oplus W_{\{1,2\},\{1,2\}}, \quad (12)$$

$$C_{\{1,2,5\},\{3\}} = W_{\{1,3\},\{2,5\}} \oplus W_{\{2,3\},\{1,5\}} \oplus W_{\{2,3\},\{1,2\}}. \quad (13)$$

For $\mathcal{J} = \{1,3,4\}$, we transmit

$$C_{\{1,3,4\},\{2\}} = W_{\{1,2\},\{3,4\}} \oplus W_{\{1,2\},\{1,3\}} \oplus W_{\{2,3\},\{1,4\}}, \quad (14)$$

$$C_{\{1,3,4\},\{3\}} = W_{\{1,3\},\{3,4\}} \oplus W_{\{1,3\},\{1,4\}} \oplus W_{\{1,3\},\{1,3\}}. \quad (15)$$

For $\mathcal{J} = \{1,3,5\}$, we transmit

$$C_{\{1,3,5\},\{2\}} = W_{\{1,2\},\{3,5\}} \oplus W_{\{1,2\},\{1,3\}} \oplus W_{\{2,3\},\{1,5\}} \oplus W_{\{2,3\},\{1,3\}}, \quad (16)$$

$$C_{\{1,3,5\},\{3\}} = W_{\{1,3\},\{3,5\}} \oplus W_{\{1,3\},\{1,5\}} \oplus W_{\{2,3\},\{1,5\}} \oplus W_{\{2,3\},\{1,3\}}. \quad (17)$$

For $\mathcal{J} = \{1,4,5\}$, we transmit

$$C_{\{1,4,5\},\{2\}} = W_{\{1,2\},\{4,5\}} \oplus W_{\{1,2\},\{1,5\}} \oplus W_{\{1,2\},\{1,4\}}, \quad (18)$$

$$C_{\{1,4,5\},\{3\}} = W_{\{1,3\},\{4,5\}} \oplus W_{\{1,3\},\{1,5\}} \oplus W_{\{2,3\},\{1,4\}}. \quad (19)$$

So user 1 can decode $W_{\{1,2\}}$ and $W_{\{1,3\}}$ in Step 1. In addition, from (8) and (9), user 2 can decode $W_{\{1,2\},\{1,3\}}$ and $W_{\{2,3\},\{1,3\}}$. From (10), user 2 can decode $W_{\{1,2\},\{1,4\}}$. From (11), user 2 can decode $W_{\{2,3\},\{1,4\}}$. From (12), user 2 can decode $W_{\{1,2\},\{1,5\}}$. From (13), user 2 can decode $W_{\{2,3\},\{1,5\}}$. Since user 2 knows $W_{\{2,3\},\{1,4\}}$ and $W_{\{1,2\},\{1,3\}}$, from (14) it can decode $W_{\{1,2\},\{3,4\}}$. Since user 2 knows $W_{\{2,3\},\{1,3\}}$, $W_{\{2,3\},\{1,5\}}$, and $W_{\{1,2\},\{1,3\}}$, from (16) it can decode $W_{\{1,2\},\{3,5\}}$. Finally, since user 2 knows $W_{\{1,2\},\{1,5\}}$ and $W_{\{1,2\},\{1,4\}}$, from (18) it can decode $W_{\{1,2\},\{4,5\}}$. So user 2 can decode $W_{\{1,2\}}$ and $W_{\{2,3\},\mathcal{V}}$ where $1 \in \mathcal{V}$ in Step 1.

Similarly to user 2, each user $k \in \{2,3,5\}$ whose demanded file is not F_1 can recover $W_{\mathcal{S}}$ where $\{d_{u_1}, d_k\} \subseteq \mathcal{S}$, and can also recover $W_{\mathcal{S}_1,\mathcal{V}_1}$ where $d_k \in \mathcal{S}_1$ and $u_1 \in \mathcal{V}_1$, after Step 1. In addition, we can check that user 4 whose demanded file is F_1, can recover $W_{\{1,2\},\{1,2\}}$ from $C_{\{1,2,4\},\{2\}}$. Similarly, each non-leader k_1 whose demanded file is $F_{d_{u_1}}$, can recover $W_{\mathcal{S}_2,\mathcal{V}_2}$ where $d_{u_1} \in \mathcal{S}_2$ and $u_1 \in \mathcal{V}_2$.

Delivery Phase Step 2 (to satisfy leader user $u_2 = 2$). Each time we consider one set of users $\mathcal{J} \subseteq ([\mathsf{K}] \setminus \{u_1\})$ where $|\mathcal{J}| = t+1 = 3$ and $u_2 \in \mathcal{J}$, and one set of files $\mathcal{B} \subseteq ([\mathsf{N}] \setminus \{d_{u_1}, d_{u_2}\})$ where $|\mathcal{B}| = \mathsf{r}-1 = 1$ (recall that $u_1 = d_{u_1} = 1, u_2 = d_{u_2} = 2$). Here we only have $\mathcal{B} = \{3\}$, and we transmit

$$C_{\{2,3,4\},\{3\}} = W_{\{2,3\},\{3,4\}} \oplus W_{\{2,3\},\{2,4\}} \oplus W_{\{1,3\},\{2,4\}} \oplus W_{\{1,3\},\{2,3\}}, \quad (20)$$

$$C_{\{2,3,5\},\{3\}} = W_{\{2,3\},\{3,5\}} \oplus W_{\{2,3\},\{2,5\}} \oplus W_{\{2,3\},\{2,3\}}, \quad (21)$$

$$C_{\{2,4,5\},\{3\}} = W_{\{2,3\},\{4,5\}} \oplus W_{\{2,3\},\{2,4\}} \oplus W_{\{1,3\},\{2,5\}}. \quad (22)$$

One can easily see that user 2 can decode $W_{\{2,3\},\{3,4\}}$, $W_{\{2,3\},\{3,5\}}$, and $W_{\{2,3\},\{4,5\}}$ from (20)-(22), respectively. Hence, combining with what user 2 decoded in Step 1, user 2 can recover $W_{\{1,2\}}$ and $W_{\{2,3\}}$.

Up to this point, we showed that two leader users are satisfied. We now show that also the remaining leaders are also satisfied (here only user $u_3 = 3$). User 3 can directly recover $W_{\{2,3\},\{2,5\}}$ from (21). Since user 3 has recovered $W_{\{1,3\}}$ in Step 1, it then can recover $W_{\{2,3\},\{2,4\}}$ from (20). Hence, user 3 can then recover $W_{\{2,3\},\{4,5\}}$ from (22).

Generally, at the end of Step 2, every user $k \in \{3,4,5\}$ whose demanded file is neither F_1 nor F_2 (here only user 3), can recover $W_{\mathcal{S}}$ where $d_k \in \mathcal{S}$, $\{d_{u_1}, d_{u_2}\} \cap \mathcal{S} \neq \emptyset$, and also recover $W_{\mathcal{S}_1,\mathcal{V}_1}$ where $d_k \in \mathcal{S}_1$ and $\{u_1,u_2\} \cap \mathcal{V}_1 \neq \emptyset$. In addition, it can be checked that user 5 whose demanded file is F_2, can recover $W_{\{2,3\},\{2,4\}}$ from $C_{\{2,4,5\},\{3\}}$. Similarly, at the end of Step 2, each non-leader k_1 whose demanded file is $F_{d_{u_2}}$, can recover $W_{\mathcal{S}_2,\mathcal{V}_2}$ where $d_{u_2} \in \mathcal{S}_2$ and $\{u_1,u_2\}\cap\mathcal{V}_2 \neq \emptyset$, and also recover $W_{\mathcal{S}_3}$ where $d_{u_2} \in \mathcal{S}_3$, $\{d_{u_1}\} \cap \mathcal{S}_3 \neq \emptyset$ (as shown at the end of Step 1).

Decoding for the leader users. In conclusion, at the end of Step 2, each leader (here users $1,2$ and 3) uses **direct decoding** to decode each of its desired sub-block, meaning that it does not use any linear combination/multicast message that included 'interference' sub-blocks. For example, user 2 uses $C_{\{1,3,5\},\{2\}}$ in (16), but it does not use $C_{\{1,3,5\},\{3\}}$ in (17) because $C_{\{1,3,5\},\{3\}}$ contains $W_{\{1,3\},\{3,5\}} \oplus W_{\{1,3\},\{1,5\}}$ which is an interference to user 2.

Decoding for non-leader users. Each non-leader user (here users 4 and 5) uses **direct decoding** and **interference alignment decoding**. Let us focus on user 5 demanding F_2. It has been shown that by direct decoding, user 5 can recover $W_{\{1,2\}}$ from Step 1, and also $W_{\{2,3\},\mathcal{V}}$ where $\{1,2\} \cap \mathcal{V} \neq \emptyset$ at the end of Step 2. Up to this point, we further need to show $W_{\{2,3\},\{3,4\}}$ transmitted in (20) can be recovered by user 5 by **interference alignment decoding**. Notice that in (20), $W_{\{1,3\},\{2,4\}} \oplus W_{\{1,3\},\{2,3\}}$ is the interference to user 5, which should be cancelled (or aligned). In addition, in (9), $W_{\{1,3\},\{2,3\}} \oplus W_{\{1,3\},\{1,2\}}$ is an interference to user 5. Furthermore, in (11), $W_{\{1,3\},\{2,4\}} \oplus W_{\{1,3\},\{1,2\}}$ is an interference of user 5. We then sum (20),(9), and (11) such that the interferences to user 5 are aligned (cancelled), and we obtain $W_{\{2,3\},\{3,4\}} \oplus W_{\{2,3\},\{2,4\}} \oplus W_{\{2,3\},\{1,3\}} \oplus W_{\{2,3\},\{1,2\}} \oplus W_{\{2,3\},\{1,4\}}$. Since user 5 has decoded $W_{\{2,3\},\{1,3\}}$ from (17), $W_{\{2,3\},\{1,2\}}$ from (13), $W_{\{2,3\},\{1,4\}}$ from (19), $W_{\{2,3\},\{2,4\}}$ from (22), it then can decode $W_{\{2,3\},\{3,4\}}$. This shows that non-leader users can recovered all the desired sub-blocks.

Performance: Based on the above, all users are able to decode their desired blocks. We sent $\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}-1}{t} + \binom{\mathsf{N}-2}{\mathsf{r}-1}\binom{\mathsf{K}-2}{t} = 15$ linear combinations, each of length

$\mathsf{B}/\left(\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}\right) = \mathsf{B}/20$ bits. So the load is $3/4$, which coincides with the conversed bound in Theorem 1 for $s = 3$. Note that the scheme in [5] only achieves a load of $9/10$.

B. General Scheme

We focus on the cases where $\mathsf{r} \in \{1, 2, \mathsf{N}-1, \mathsf{N}\}$, or $t = \mathsf{KMr}/\mathsf{N} \in \{1, 2, \mathsf{K}-1, \mathsf{K}\}$, or each user has a distinct request.

Placement Phase: Given an integer t, we partition each block $W_{\mathcal{S}}$ into $\binom{\mathsf{K}}{t}$ equal-length sub-blocks, $W_{\mathcal{S}} = \{W_{\mathcal{S},\mathcal{V}} : \mathcal{V} \subseteq [\mathsf{K}], |\mathcal{V}| = t\}$. For each block $W_{\mathcal{S}}$, each user $k \in [\mathsf{K}]$ caches $W_{\mathcal{S},\mathcal{V}}$, where $\mathcal{V} \subseteq [\mathsf{K}]$ and $|\mathcal{V}| = t$, if $k \in \mathcal{V}$. Each sub-block contains $\mathsf{B}/\left(\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}\right)$ bits and each user caches $\mathsf{B}\binom{\mathsf{N}}{\mathsf{r}}\binom{\mathsf{K}-1}{t-1}/\left(\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}\right) = \mathsf{MB}$ bits.

Delivery Phase: We consider a demand vector $\mathbf{d}$; for each demanded file in $\mathbf{d}$, we pick a leader user and we let $\mathbf{u} = (u_1, u_2, ..., u_{N_{\mathrm{e}}(\mathbf{d})})$ be a permutation of this leader set. Our scheme contains $\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, N_{\mathrm{e}}(\mathbf{d})\}$ steps.

In Step $j \in [\min\{N_{\mathrm{e}}(\mathbf{d}), \mathsf{N}-\mathsf{r}+1, \mathsf{K}-t\}]$ we satisfy the demand of leader user u_j as follows. For each set of users $\mathcal{J} \subseteq ([\mathsf{K}] \setminus \{u_1, \ldots, u_{j-1}\})$ where $|\mathcal{J}| = t+1$ and $u_j \in \mathcal{J}$, and for each set of files $\mathcal{B} \subseteq ([\mathsf{N}] \setminus \{d_{u_1}, \ldots, d_{u_j}\})$ where $|\mathcal{B}| = \mathsf{r}-1$, we transmit $C_{\mathcal{J},\mathcal{B}}$ as defined in (7).

We then focus on one multicast message $C_{\mathcal{J},\mathcal{B}}$ transmitted in Step j. From its construction, $C_{\mathcal{J},\mathcal{B}}$ contains only one sub-block desired by user u_j (which is $W_{\{d_{u_j}\}\cup\mathcal{B},\mathcal{J}\setminus\{u_j\}}$), while all other sub-blocks are cached by user u_j; this is so because $|\mathcal{B}| = \mathsf{r}-1$ and $d_{u_j} \notin \mathcal{B}$. Furthermore, it is proved in the extended version of this paper [11] that, for each user in $\mathcal{J}$, the sub-blocks in $C_{\mathcal{J},\mathcal{B}}$ which it has not cached or decoded previously, are decodable from the transmission in Step j. In addition, $C_{\mathcal{J},\mathcal{B}}$ is also useful to users in $[\mathsf{K}] \setminus (\mathcal{J} \cup \{d_{u_1}, ..., d_{u_{j-1}}\})$, whose demanded file is in $\mathcal{N}(\mathcal{J}) \cup \mathcal{B}$. For each of these users (assumed to be k'),

$$\bigoplus_{k\in\mathcal{J}} \bigoplus_{\substack{\mathcal{S}\subseteq(\mathcal{N}(\mathcal{J})\cup\mathcal{H}\setminus\{d_{k'}\}):\\ |\mathcal{S}|=\mathsf{r},\mathcal{B}\subseteq\mathcal{S},d_k\in\mathcal{S}}} W_{\mathcal{S},\mathcal{J}\setminus\{k\}},$$

is an interference, and all other sub-blocks in $C_{\mathcal{J},\mathcal{B}}$ are desired. $C_{\mathcal{J},\mathcal{B}}$ is treated as noise for each user in $\{d_{u_1}, ..., d_{u_{j-1}}\}$ and for each user whose demanded file is not in $\mathcal{N}(\mathcal{J})\cup\mathcal{B}$. Moreover, for each leader in $\{u_{j+1}, ..., u_{N_{\mathrm{e}}(\mathbf{d})}\}$ whose demanded file is in $\mathcal{N}(\mathcal{J})\cup\mathcal{B}$, $C_{\mathcal{J},\mathcal{B}}$ is also treated as noise if it includes interference(s) to this leader. Decodability is formally stated in the following Lemma (proved in [11]).

Lemma 1 (Decodability)**.** *In an* $(\mathsf{N}, \mathsf{K}, \mathsf{M}, \mathsf{r})$ *shared-link caching problem with correlated files, in any of the three cases described in Theorem 2, from our proposed interference alignment scheme, for any user* $k \in [\mathsf{K}]$*, we have:*

1) *if* k *is a leader, it can decode* $W_{\mathcal{S}}$ *and* $W_{\mathcal{S}_1,\mathcal{V}_1}$ *at the end of Step* $j \in [\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, N_{\mathrm{e}}(\mathbf{d})\}]$ *by direct decoding, where* $\mathcal{S} \subseteq [\mathsf{N}]$, $|\mathcal{S}| = \mathsf{r}$, $\{d_{u_1}, ..., d_{u_j}\} \cap \mathcal{S} \neq \emptyset$, $d_k \in \mathcal{S}$, $\mathcal{S}_1 \subseteq [\mathsf{N}]$, $|\mathcal{S}_1| = \mathsf{r}$, $d_k \in \mathcal{S}_1$, *and* $\{u_1, ..., u_j\} \cap \mathcal{V}_1 \neq \emptyset$*;*
2) *if* k *is not a leader and the leader demanding* d_k *is user* u_f*, by direct decoding user* k *can decode* $W_{\mathcal{S}}$ *at the end of Step* $j_1 \in [\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, N_{\mathrm{e}}(\mathbf{d}), f-1\}]$*, where* $\mathcal{S} \subseteq [\mathsf{N}]$, $|\mathcal{S}| = \mathsf{r}$, $\{d_{u_1}, ..., d_{u_{j_1}}\} \cap \mathcal{S} \neq \emptyset$, $d_k \in \mathcal{S}$*, and also decode* $W_{\mathcal{S}_1,\mathcal{V}_1}$ *at the end of Step* $j_2 \in [\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, N_{\mathrm{e}}(\mathbf{d}), f\}]$*, where* $\mathcal{S}_1 \subseteq [\mathsf{N}]$, $|\mathcal{S}_1| = \mathsf{r}$, $d_k \in \mathcal{S}_1$*, and* $\{u_1, ..., u_{j_2}\} \cap \mathcal{V}_1 \neq \emptyset$*; in addition, it can decode the remaining sub-blocks from the remaining steps by direct decoding and interference alignment decoding.*

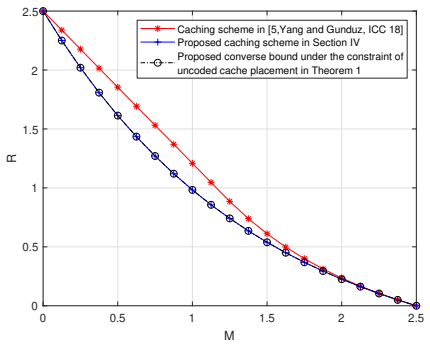

Fig. 1: $(\mathsf{N}, \mathsf{K}, \mathsf{r}) = (5, 20, 2)$ caching problem with correlated files.

Performance: After all steps, by Lemma 1, each user can recover its desired blocks. In step $j \in [\min\{\mathsf{N}-\mathsf{r}+1, \mathsf{K}-t, N_{\mathrm{e}}(\mathbf{d})\}]$ we transmit $\frac{\mathsf{B}\binom{\mathsf{N}-j}{\mathsf{r}-1}\binom{\mathsf{K}-j}{t}}{\binom{\mathsf{N}-1}{\mathsf{r}-1}\binom{\mathsf{K}}{t}}$ bits. Summing the load of all steps, the total load is $c_t^{N_{\mathrm{e}}(\mathbf{d})}$ in (5).

Numerical Evaluations: In Fig. 1, for demand type $\mathcal{D}_{\mathsf{N}}$, we compare the average loads achieved by our proposed scheme and the scheme in [5] for an $(\mathsf{N}, \mathsf{K}, \mathsf{r}) = (5, 20, 2)$. Our proposed scheme outperforms the existing scheme and meet out the converse bound in Theorem 1.

REFERENCES

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Infor. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[2] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," *in IEEE Infor. Theory Workshop*, Sep. 2016.
[3] Q. Yu, M. A. Maddah-Ali, and S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *in IEEE Int. Symp. Inf. Theory*, Jun. 2017.
[4] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-memory trade-off for caching and delivery of correlated sources," *arXiv:1806.07333*, Jan. 2018.
[5] Q. Yang and D. Gunduz, "Centralized coded caching of correlated contents," *in IEEE Intern. Conf. Commun (ICC 2018)*, May 2018.
[6] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching and coded multicasting: Multiple groupcast index coding," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 881–885, 2014.
[7] A. Sengupta and R. Tandon, "Improved approximation of storage-rate tradeoff for caching with multiple demands," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 1940–1955, May. 2017.
[8] Y. Wei and S. Ulukus, "Coded caching with multiple file requests," *in 55th Annual Allerton Conf. on Commun., Control, and Computing (Allerton)*, Oct. 2017.
[9] F. Arbabjolfaei, B. Bandemer, Y.-H. Kim, E. Sasoglu, and L. Wang, "On the capacity region for index coding," *in IEEE Int. Symp. Inf. Theory*, Jul. 2013.
[10] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," *in IEEE Int. Symp. Inf. Theory*, Jul. 2016.
[11] K. Wan, D. Tuninetti, M. Ji, and G. Caire, "On coded caching with correlated files," *available at arXiv:1901.05732*, Jan. 2019.