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 \( r \in [N] \), correlation is modelled as follows: each \( r \)-subset of files contains a common block. The tradeoff between the cache size and the average transmitted load is considered. We first propose a converse bound under the constraint of uncoded cache placement (i.e., each user directly caches a subset of the library bits). We then propose an interference alignment scheme for the cases where users have different requests. The proposed scheme achieves the optimal average load under uncoded cache placement when users demand distinct files. In addition, an extension of the proposed scheme achieves an optimal average load among all possible distinct requests under the constraint of uncoded cache placement. We first propose a converse bound under the constraint of uncoded cache placement (i.e., each user directly caches a subset of the library bits). We then propose an interference alignment scheme for the cases where users have different requests. The proposed scheme achieves the optimal average load under uncoded cache placement when users demand distinct files. In addition, an extension of the proposed scheme achieves the optimal average load among all possible distinct requests (i.e., not necessarily distinct demands) for \( KrM \leq 2N \) or \( KrM \geq (K-1)N \) or \( r \in \{1, 2, N-1, N\} \). As a by-product, we show that the proposed scheme reduces the load of existing schemes for the caching problem with multi-requests, and reduces the required finite field size for the distributed computation problem.

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 scheme was originally 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 such that during delivery multicast messages simultaneously satisfy the demands of different users. Under the constraint of uncoded cache placement (i.e., each user directly caches a subset of the library bits) and for worst-case load, the MAN scheme was proved to be optimal when \( N \geq K \) [2]; later, a modification on the MAN scheme was shown to be optimal for \( N < K \) [3].

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 \( K \)-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 \( r \) files (referred to as a ‘\( r \)-block’) is the same, was considered; each file contains \( \binom{N-r}{\ell} \) \( r \)-blocks. The achievable scheme in [5] contains \( N \) steps, and in step \( \ell \) only \( r \)-blocks are transmitted; there are \( \binom{N-1}{r-1} \) rounds for the transmission of step \( \ell \), 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 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 schemes designed for single requests: (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 \( r \in [N] \) and assume each file only contains \( r \)-blocks (see Section II). In Section III we derive a converse bound on the minimal average load among all possible demands under the constraint of uncoded cache placement, by leveraging the index coding converse bound ideas as in [9]. 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 \( KrM \leq 2N \) or \( KrM \geq (K-1)N \) or \( r \in \{1, 2, N-1, N\} \).

For the caching problem with multi-requests [8], our proposed scheme is optimal under the constraint of MAN placement for the only four cases with \( L = 2, K \leq 4, M = N/K \), where the scheme in [6] is sub-optimal; it also reduces the field size for the distributed computation problem [10].

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. For \(n\)-dimensional Euclidean space, \(e_{n,i}\) denotes the unit-norm length-\(n\) vector with all zero entries except the \(i\)-th one.

II. SYSTEM MODEL

In a \((N, K, M, r)\) shared-link caching problem with correlated files, a server has access to a library of \(N\) files (each of which contains \(B\) bits), where \(M \in [0, N/r]\). We denote the content in the cache of user \(k \in [K]\) by \(Z_k\) and let \(Z := \{Z_1, \ldots , Z_k\}\).

During the delivery phase, user \(k \in [K]\) demands file \(d_k \in [N]\). The demand vector \(d := (d_1, \ldots , d_{K})\) is revealed to all nodes. Given \((d, Z)\), the server broadcasts a message \(X(d, Z)\) of \(BR(d, Z)\) bits to all users. User \(k \in [K]\) must recover its desired file \(F_{d_k}\) from \(Z_k\) and \(X(d, Z)\).

A demand vector \(d\) is said to be of type \(D_{N_k}(d)\) if it has \(N_k(d) := |\{d_k \in [K]\}|\) distinct entries. Based on the uniform demand distribution, the objective is to determine the optimal average load among all demands of the same type

\[
R^*(M, s) := \min_{Z} \mathbb{E}_{d \in D_s} \left[ R(d, Z) \right], \forall s \in \{\min\{K, N\}\},
\]

and the optimal average load among all possible demands

\[
R^*(M) := \min_{Z} \mathbb{E}_{d \in [N]^K} \left[ R(d, Z) \right].
\]

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 \(W_S\) where \(S \subset [N]\) and \(|S| = r\) in to sub-blocks, \(W_S := \{W_{S,V} : V \subset [K]\}\), where \(W_{S,V}\) represents the bits of \(W_S\) which are exclusively cached by users in \(V\). The optimal loads under uncoded cache placement \(R^*_s(M, s)\) and \(R^*_s(M)\) are defined as in [1] and [2], respectively.

III. MAIN RESULTS

Theorem 1 (Converse). For a \((N, K, M, r)\) shared-link caching problem with correlated files, \(R^*_s(M, s)\) is lower bounded by the lower convex envelope of the following memory-load pairs

\[
(M, R_s) = \left( Nt/K, c^*_s x_t \right), \forall t \in [0 : K],
\]

for all \(s \in \{\min\{K, N\}\}\), where

\[
c^*_s := \sum_{j \in \{\min\{N-r+1, K-t,s\}\}} \binom{(N-j)}{(K-j)} \binom{(N-1)}{(r-1)}.
\]

In addition, \(R^*_s(M)\) is lower bounded by the lower convex envelope of the following memory-load pairs

\[
(M, R_s) = \left( Nt/K, \mathbb{E}_{d \in [N]^K} \left[ c^*_s x_t(d) \right] \right), \forall t \in [0 : K].
\]

Proof: Inspired by [11], we use the “acyclic index coding converse bound” from [9]. For a demand vector \(d\) demand of type \(D_s\) where \(s \in \{\min\{K, N\}\}\), choose \(s = N_k(d)\) users with distinct demands. Generate a directed graph for the delivery phase, where each sub-block demanded by each of these \(N_k(d)\) 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 represents by node \(j\) caches the sub-block represented by node \(i\). Consider a permutation of these \(N_k(d)\) users, denoted by \(u = (u_1, u_2, \ldots , u_{N_k(d)})\). By [2] Lemma 1, we can prove that the set of sub-blocks

\[
\bigcup_{k \in [N_k(d)]} \bigcup_{S \subset \{N\} \setminus \{d_{u_1}, \ldots , d_{u_{N_k-1}}\}} \bigcup_{|S| = r} W_{S,V}
\]

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 \(s \in \{\min\{K, N\}\}\), 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

\[
R^*_s(M, s) \geq \sum_{t=0}^{K} c^*_s x_t,
\]

where \(x_t\) represent the fraction of all bits cached exactly by \(t\) users. After Fourier-Motzkin elimination of the \(\{x_t\}\) as in [11], one obtains the converse bound in [3]. After considering all demand types, one obtains the converse bound in [3].

We propose a multi-round interference alignment based scheme in Section [Y] for the three cases described in Theorem 2. Our scheme contains three main ingredients. First, we divide the delivery phase into steps, and in each step we satisfy the demand of one user. The users to serve are chosen such that they have different demands. Second, in each step, we construct multicast messages by using a grouping method (to be described later). The multicast message for each group is useful to some users while others treat it as noise. Third, after the construction of multicast messages destined for all groups, each user can cancel (or align) all non-intended ‘symbols’ (interferences) in all multicast messages which are useful to it. Different from existing round-division methods in [5]–[8], our caching problem with correlated files, we have

1. Case 1: When \(N \geq K\), \(R^*_s(M, K)\) is equal to the lower convex envelopes of \(c^*_s t\) where \(t \in [0 : K]\).
2. Case 2: When \(r \in \{1, 2, N - 1\}\), \(R^*_s(M, s)\) where \(s \in \{\min\{K, N\}\}\) and \(R^*_s(M)\) are equal to the lower convex
Proof: Comparing the reverse 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 achieved because the reverse bound \( \sum_{i=0}^{\infty} c_i^2 \) is convex in terms of \( t \) and when \( t \in \{0, 1, 2, K-1, K\} \), our proposed scheme is optimal.

Remark 1. When \( N \times K \) each user demands a distinct file, the \( (N,K,M,r) \) shared-link caching problem with correlated files, is related to the distributed computation problem in [10]. The only difference is that in [10] the link is D2D, as opposed to the shared-link considered here. In [10], the authors proposed an optimal scheme that requires to exchange messages from a large field size. We can directly extend our proposed shared-link caching scheme to the D2D case, as done in [12], so as to achieve the optimal load of [10] but by only using operations on the binary field.

Remark 2. For the caching problem with multi-requests considered in [8] where each user requests \( L \) 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 \( K \leq 4 \) users, \( M = N/K \), and \( L = 2 \), except one demand for \( K = 3 \) and three demands for \( K = 4 \). We can use the proposed scheme in this paper to achieve the optimality for those four unsolved cases. The details are given in Appendix A.

Remark 3. The proposed caching scheme can be extended to characterize the optimal worst-case load under the cases described in Theorem 2 which is equal to \( R_{K}^0(M, \min\{K, N\}) \).

IV. NOVEL INTERFERENCE ALIGNMENT BASED SCHEME

A. Example

We first examine an example to highlight the key idea. Consider an \((N,K,M,r)\) shared-link caching problem with correlated files with \( N = 3 \), \( K = 5 \), \( M = 3/5 \) and \( r = 2 \). Here \( M = 2N/(rK) \) as in Case 3 of Theorem 2. There are three 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 cache placement. Let \( t = \frac{KMr}{N} = 2 \). We divide each block into \( \binom{N}{r} \) = 10 non-overlapping and equal-length sub-blocks, \( W_{SV} = \{W_{SV} : V \subseteq [K], |V| = t \} \). Each user \( k \in [K] \) caches \( W_{SV} \) for all \( V \subseteq [K] \) of size \(|V| = t \) if \( k \in V \). Hence, each sub-block contains \( B_{k}^{(N)} / \binom{N-1}{r-1} = B/20 \) bits and each user caches \( B_{k}^{(N)} / \binom{N-1}{r-1} / \binom{N-1}{K} = MB \) bits.

Delivery Phase: Assume \( d = (1, 2, 3, 1, 2) \), which has \( N_{c}(d) = 3 \) 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 \( \min\{N - r + 1, K - t, N_c(d)\} = 2 \) steps; after Step i, the \( i^\text{th} \) element/user in the permutation can decode its desired file: after finishing all steps, the remaining users can also decode their desired file.

The delivery phase has two steps, for which we need the following notation. The union set of blocks demanded by the users in \( J \) is denoted by \( U_{J} \), and the intersection set of the same as \( I_{J} \); e.g., \( U_{\{1,2,3\}} = \{W_{[1,2]}, W_{[1,3]}, W_{[2,3]}\} \) and \( I_{\{1,2\}} = \{W_{[1,2]}\} \).

Delivery Phase Step 1. Each time we consider one set of users \( J \subseteq [K] \) where \(|J| = t + 1 \) and \( 1 \in J \) (recall that 1 is the user indexed by the first element of the chosen permutation). For example, we focus on \( J = \{1, 2, 3\} \). We divide the blocks in \( U_{\{1,2,3\}} \) into groups. For each block \( W_{SV} \in U_{J} \), we compute \( S' \cap (\cup_{k \in [1,2,3]} \{d_k\}) \). So we have \( \{1,2\} \cap (\cup_{k \in [1,2,3]} \{d_k\}) = \emptyset, \{1,3\} \cap (\cup_{k \in [1,2,3]} \{d_k\}) = \emptyset, \) and \( \{2,3\} \cap (\cup_{k \in [1,2,3]} \{d_k\}) = \emptyset \).

Hence, all blocks in \( U_{\{1,2,3\}} \) are in one group with \( B = S' \cap (\cup_{k \in [1,2,3]} \{d_k\}) = \emptyset \) for each \( W_{SV} \) in this group. In addition, since \( W_{[1,2]} \) is demanded by user 1 and by user 2, if they receive \( W_{[1,2]} \), then each of them can decode one demanded sub-block. Recall that \( e_{n,d} \) is the \( i^\text{th} \) standard basis for \( n \)-dimensional Euclidean space. Since the number of blocks in this group demanded by user 1 is two, we transmit two linear combinations,

\[
\begin{align*}
(W_{[1,2]} \oplus W_{[1,2]}, W_{[1,3]} \oplus W_{[1,3],1,3})e_{2,1} & \quad \text{(10a)} \\
(W_{[1,3]} \oplus W_{[1,3],1,2})e_{2,2} & \quad \text{(10b)} \\
(W_{[2,3]} \oplus W_{[2,3],1,2})e_{2,1} & \quad \text{(10c)}
\end{align*}
\]

In other words, the two linear combinations are,

\[
\begin{align*}
W_{[1,2]} \oplus W_{[1,2],1,2} & \quad \text{(11)} \\
W_{[1,3]} \oplus W_{[1,3],1,2} & \quad \text{(12)} \\
W_{[2,3]} \oplus W_{[2,3],1,2} & \quad \text{(12)}
\end{align*}
\]

User 1 knows \( W_{[2,3],1,2} \) and then it can decode \( W_{[1,2],1,2} \) and \( W_{[1,3],1,2} \) and \( W_{[2,3],1,2} \). Hence, user 1 can decode \( W_{[1,2],1,2}, W_{[1,3],1,2}, W_{[2,3],1,2} \). Similarly, user 2 can decode \( W_{[1,2],1,2}, W_{[2,3],1,2} \) and user 3 can decode \( W_{[1,2],1,2}, W_{[2,3],1,2} \). In addition, the linear combinations in [10] are useful to other users whose demanded file is in \( \cup_{k \in [1,2,3]} \{d_k\} \subseteq B = \{1, 2, 3\} \), i.e., users 4 and 5, where \( W_{[3,1,2]} \) is an interference to user 4 and \( W_{[1,3],2,3} \) is an interference to user 5.

Let us then focus on \( J = \{1, 2, 4\} \), where we have \( U_{\{1,2,4\}} = \{W_{[1,2]}, W_{[1,3]}, W_{[2,3]}\} \). We can compute \( \{1,2\} \cap (\cup_{k \in [1,2,4]} \{d_k\}) = \emptyset, \{1,3\} \cap (\cup_{k \in [1,2,4]} \{d_k\}) = \{3\}, \) and \( \{2,3\} \cap (\cup_{k \in [1,2,4]} \{d_k\}) = \{3\} \). Hence, we divide the blocks in \( U_{\{1,2,4\}} \) into two groups, where in the first group we have \( W_{[1,2]} \) and in the second group we have \( W_{[1,3]}, W_{[2,3]} \). In the first group with \( B = \emptyset \), the number of blocks demanded by user 1 is one. We transmit the following linear combination,

\[
(W_{[1,2]} \oplus W_{[1,2],1,2} \oplus W_{[2,3],1,2})e_{1,1} \quad \text{(13)}
\]
where 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, the linear combination in (13) is also useful to other user whose demanded file is in $(\cup_{k \in \{1,2,4\}} \{d_k\}) \cup B = \{1,2\}$, i.e., user 5. Meanwhile, for the remaining user (user 3), it treats the linear combination in (13) as noise. In the second group with $B = \{3\}$, the number of blocks demanded by user 1 is one. So we transmit

$$\left( W_{(1,3),(2,4)} + W_{(1,3),(1,2)} \right) e_{1,1} + W_{(2,3),(1,4)} e_{1,1},$$

where 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, the linear combination in (14) is also useful to other users whose demanded file is in $(\cup_{k \in \{1,2,4\}} \{d_k\}) \cup B = \{1,2,3\}$, i.e., users 3, 5, where $W_{(1,3),(2,4)} + W_{(1,3),(1,2)}$ is an interference to user 5.

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

$$\left( W_{(1,2),(2,5)} + W_{(1,2),(1,5)} + W_{(1,2),(1,2)} \right) e_{1,1},$$

$$W_{(1,3),(2,5)} e_{1,1} + (W_{(2,3),(1,5)} + W_{(2,3),(1,2)}) e_{1,1}.$$  (15) (16)

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

$$\left( W_{(1,2),(3,4)} + W_{(1,2),(1,3)} \right) e_{1,1} + W_{(2,3),(1,4)} e_{1,1},$$

$$W_{(1,3),(3,4)} + W_{(1,3),(1,4)} + W_{(1,3),(1,3)} e_{1,1}.$$  (17) (18)

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

$$\left( W_{(1,2),(3,5)} + W_{(1,2),(1,3)} \right) e_{1,1} + \left( W_{(1,3),(3,5)} + W_{(1,3),(1,5)} \right) e_{2,1} + \left( W_{(2,3),(1,5)} + W_{(2,3),(1,3)} \right) e_{2,1} + e_{2,2}.$$  (19)

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

$$\left( W_{(1,2),(4,5)} + W_{(1,2),(1,5)} + W_{(1,2),(1,4)} \right) e_{1,1},$$

$$W_{(1,3),(4,5)} + W_{(1,3),(1,5)} + W_{(1,3),(1,4)} e_{1,1}.$$  (20) (21)

So user 1 can decode $W_{(1,2)}$ and $W_{(1,3)}$ in Step 1. In addition, we prove that user 2 can decode $W_{(1,2)}$ and $W_{(2,3)}$ when $1 \in \mathcal{V}$ in Step 1. From (14), user 2 can decode $W_{(1,2),(1,3)}$ and $W_{(2,3),(1,3)}$. From (13), user 2 can decode $W_{(1,2),(1,4)}$. From (15), user 2 can decode $W_{(1,2),(1,5)}$. From (16), user 2 can decode $W_{(2,3),(1,5)}$. Since user 2 knows $W_{(1,2),(1,5)}$, it can decode $W_{(1,2),(1,3)}$.

Finally, since user 2 knows $W_{(1,2),(1,5)}$ and $W_{(1,2),(1,4)}$, from (20) it can decode $W_{(1,2),(4,5)}$. So user 2 can decode $W_{(1,2)}$ and $W_{(2,3)}$, where $1 \in \mathcal{V}$.

**Delivery Phase Step 2.** Each time we consider one set of users $J \subseteq [K] \setminus \{1\}$ where $|J| = t + 1$ and $2 \in J$ (recall that 2 is the user indexed by the second element of the chosen permutation). We first focus on $J = \{2,3,4\}$. Different from the previous step, since user 2 has decoded $W_{(1,2)}$, in Step 2, we only consider the blocks in $U_{(2,3,4)} \setminus \{W_{(1,2)}\} = \{W_{(1,3)}, W_{(2,3)}\}$. Since $(1,3) \setminus (\cup_{k \in \{2,3,4\}} \{d_k\}) = \emptyset$ and $(2,3) \setminus (\cup_{k \in \{2,3,4\}} \{d_k\}) = \emptyset$, both of these blocks are in the same group. Since in this group the number of blocks demanded by user 2 is one, we transmit

$$W_{(2,3),(3,4)} + W_{(2,3),(2,4)} e_{1,1} + (W_{(1,3),(2,4)} + W_{(1,3),(2,3)}) e_{1,1},$$

Similarly, for $J = \{2,3,5\}$, we transmit

$$(W_{(2,3),(3,5)} + W_{(2,3),(2,5)} + W_{(2,3),(2,3)}) e_{1,1}.$$  (22) (23)

For $J = \{2,4,5\}$, we transmit

$$(W_{(2,3),(4,5)} + W_{(2,3),(2,4)}) e_{1,1} + W_{(1,3),(2,5)} e_{1,1}.$$  (24)

So user 2 can decode $W_{(2,3),(3,4)}$, $W_{(2,3),(3,5)}$, and $W_{(2,3),(4,5)}$ from (22)–(24), respectively. Hence, combining with Step 1, user 2 can recover $W_{(1,2)}$ and $W_{(2,3)}$.

**How interference alignment works and successful decoding is achieved.** Each leader (here users 1, 2 and 3) uses direct decoding, meaning that it does not use any linear combination including interference in order to decode its desired sub-blocks. For example, user 2 only uses the first linear combination in (19) and does not use the second one of (19) because the second one contains $W_{(1,3),(3,5)} + W_{(1,3),(1,3)}$ which is an interference to user 2.

Each non-leader user (here users 4 and 5) uses interference alignment decoding. Let us focus on user 5 demanding $F_2$. Each sub-block $W_{1,2,v}$ or $W_{2,3,v}$ where $1 \in \mathcal{V}$ can be directly decoded from the linear combination(s) for $J = \{5\} \cup \mathcal{V}$. Similarly, each sub-block $W_{2,3,v}$ where $1 \notin \mathcal{V}$ and $2 \in \mathcal{V}$ can directly decoded from the linear combination(s) for $J = \{5\} \cup \mathcal{V}$. Since user 5 knows $W_{(2,3),(1,3)} + W_{(2,3),(1,2)}$ and $W_{(2,3),(1,3)}$, it then can decode $W_{(1,2),(2,2)}$ from (10). Similarly, it can decode $W_{(1,2),(2,4)}$ and $W_{(1,2),(3,4)}$ from (13) and (17) in Step 1. User 5 decode the above sub-blocks by direct decoding and decode $W_{(2,3),(3,4)}$ from (22) by aligning interferences. Notice that in $W_{(1,3),(2,4)} + W_{(1,3),(2,3)}$ is the interference to user 5, which should be cancelled (or aligned). In addition, in (12), $W_{(1,3),(3,3)} + W_{(1,3),(1,2)}$ is an interference to user 5. Furthermore, in $W_{(1,3),(2,4)} + W_{(1,3),(1,2)}$ is an interference of user 5. We then sum (22), (12), and (14) such that the interferences to user 5 are aligned (canceled), and we obtain $W_{(2,3),(3,4)} + W_{(2,3),(2,4)} + W_{(2,3),(1,3)} + W_{(2,3),(1,2)} + W_{(2,3),(1,4)}$. Since user 5 has decoded $W_{(2,3),(1,3)}$ from (19), $W_{(2,3),(1,2)}$ from (16), $W_{(2,3),(1,4)}$ from (17), $W_{(2,3),(2,4)}$ from (24), then it can decode $W_{(2,3),(3,4)}$.

**Performance:** Based on the above, all users are able to decode their desired blocks. We sent $\binom{N-1}{K-1} K^{-1} + \binom{N-2}{K-2} K^{-2} = 15$ linear combinations, each of length $B/\binom{N-1}{K-1} = B/20$ bits. So the load is $3/4$, which coincides with the reversed 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 $r \in \{1, 2, N - 1, N\}$, or $t = KMr/N \in \{1, 2, K - 1, K\}$, or each user has a distinct request. **Placement Phase:** For each integer $t = KMr/N$, we divide each block $W_s$ into $\binom{n}{t}$ non-overlapping and equal-length sub-blocks, $W_s = W_{s,v} : \mathcal{V} \subseteq [K], |\mathcal{V}| = t$. For each block $W_s$, each user $k \in [K]$ caches $W_{S,v}$, where $\mathcal{V} \subseteq [K]$ and
\(|V| = t\), if \(k \in V\). Each sub-block contains \(B_j/(\binom{N-1}{t-1}\binom{k}{i})\) bits and each user caches \(B_j/\left(\binom{N-1}{t-1}\binom{k}{i}\right) = MB\) bits.

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

In Step \(j \in \min\{|N - r + 1, K - t, N_c(d)|\}\), we consider each set of users \(J \subseteq \{K \setminus \{u_1, \ldots, u_{j-1}\}\}\) where \(|J| = t + 1\) and \(u_j \in J\). We divide the blocks in \(C^j_{J} = \cup_{T \subseteq \{u_{j_1}, \ldots, u_{j_k}\}}\{T\}\) into non-overlapping groups, \(G^j_{J,B} := \{W \in C^j_{J} : S \notin \{u_{j_1}, \ldots, u_{j_k}\}\} = B\), where \(B \subseteq \{K \setminus \{u_{j_1}, \ldots, u_{j_k}\}\}\). Since \(u_{j} \in J\), all blocks \(W_S\) where \(S \subseteq \{K \setminus \{u_{j_1}, \ldots, u_{j_k}\}\}\), \(|S| = r\), and \(u_j \in S\), are in \(C^j_{J}\). Hence, in \(C^j_{J}\), there are \(\binom{N-1}{t-1}\) blocks demanded by user \(u_j\). For each group \(G^j_{J,B}\), we assume the blocks in \(G^j_{J,B}\) are \(W_{S_j, \ldots, W_{S_{|J|-1}}}\), the first \(n^j_{J,B} := \{u_i_{j}, \ldots, u_{j_{|J|-1}}\} - 1\) of which are demanded by user \(u_j\). We transmit \(n^j_{J,B}\) linear combinations,

\[
\left(\sum_{k \in J_{J,B}} W_{S_{\lambda_j}}(k)\right) e_{n^j_{J,B}} + \sum_{q \in \{G^j_{J,B}\}} \left(\sum_{k \in J_{J,B}} W_{S_{\lambda_j}}(k)\right) e_{n^j_{J,B}}^q.
\]

It is proved in Appendix [Appendix B] that for each user in \(J\), the sub-blocks in (25) which has not been cached or decoded previously, are decodable from (25). The linear combinations for this group are also useful to users in \(K \setminus \{J \cup \{u_{j_1}, \ldots, u_{j_k}\}\}\), whose demanded file is in \((u_{j_1}, \ldots, u_{j_k})\) \(B\). For each of these users (assumed to be \(k'\)), \(W_{S_{j_1}}(k)\) is an interference, where \(i_1 \in \{G^j_{J,B}\}\) and \(k' \notin S_{j_1}\), and all other sub-blocks in (25) are desired. The linear combinations for this group are treated as noise for each user in \(\{u_{j_1}, \ldots, u_{j_k}\}\) and for each user whose demanded file is not in \((u_{j_1}, \ldots, u_{j_k})\) \(B\). Moreover, for each leader in \(\{u_{j_1, \ldots, u_{j_k}}(d)\}\) whose demanded file is in \((u_{j_1}, \ldots, u_{j_k})\) \(B\), any linear combination for this group in (25) which includes interference(s) to it, is also treated as noise.

**Performance:** After all steps are done, by Lemma [Lemma 1] next whose proof can be found Appendices [Appendix B] and [Appendix C] the users’ interferences can be cancelled (or aligned). So each user can recover its desired blocks. In step \(j \in \min\{|N - r + 1, K - t, N_c(d)|\}\) we transmit \(\frac{B_j}{\binom{N-1}{t-1}\binom{k}{i}}\) bits. Summing the load in each step, the total load is \(s_t\) in (5).

**Lemma 1 (Decodability).** In a \((N, K, M, r)\) shared-link caching problem with correlated files, in any of the three cases described in Theorem [Appendix A] from our proposed interference alignment scheme, considering a permutation of leaders \(u = (u_1, u_2, \ldots, u_{N_c(d)})\), for any user \(k \in [K]\), we have

1) if \(k\) is a leader, it can decode \(W_s, W_{S_{i_1}}, W_{S_{j_1}}\) at the end of Step \(j \in \min\{|N - r + 1, K - t, N_c(d)|\}\) by direct decoding, where \(S \subseteq \{N\}, |S| = r\), \(\{d_{u_1, \ldots, d_{u_j}\}} \cap S \neq \emptyset\), \(d_{u} \in S\), \(S_t \subseteq \{N\}, |S_t| = r, d_k \in S_t\), and \(\{u_{1, \ldots, u_{j}}\} \cap V_1 \neq \emptyset\); in addition, it can decode the remaining sub-blocks from the remaining steps by interference alignment decoding.

**Numerical Evaluations:** In Fig. 1 for demand type \(D_k\), we compare the average loads achieved by our proposed scheme and the scheme in [5] for a \((N, K, r) = (5, 20, 2)\). Our proposed scheme outperforms the existing scheme and coincides with the converse bound in Theorem [Theorem 1].

**ACKNOWLEDGEMENT**

The work of K. Wan is supported by the project CARENET (ERC Grant).

**APPENDIX A**

**CODES FOR REMARK**

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

1) \(d_1 = \{F_1, F_2\}, d_2 = \{F_1, F_3\}, d_3 = \{F_2, F_3\}\) (case \(D_7\) in [8]). We use the MAN placement and divide each file \(F_i\), where \(i \in [N]\) into \(\hat{K}\) non-overlapping and equal-length subfiles, \(F_i = \{F_i, W_s : W \subseteq [K], |W| = t\}\), where \(t = KM/N = 1\). It can be seen this case is equivalent to our considered \((N, K, M, r) = (3, 3, 1, 2)\) shared-link caching problem with correlated files. Hence, we can directly use the proposed delivery phase in this
paper to transmit the linear combinations (with leader permutation (1, 2, 3))

\[ F_{1, (2)} \oplus F_{1, (1)}, F_{1, (3)} \oplus F_{3, (1)}, F_{2, (2)} \oplus F_{3, (1)}, F_{2, (3)} \oplus F_{2, (1)}, F_{3, (3)} \oplus F_{3, (2)}. \]

Hence, the load is 5/3 which coincides with the converse bound under the constraint of MAN placement in [8], while the proposed caching scheme in [8] achieves 2.

2) \( d_1 = \{F_1, F_2\}, d_2 = \{F_1, F_3\}, d_3 = \{F_2, F_3\}, \) and \( d_4 = \{F_1, F_2\} \) (case \( D_{1,7}^{r} \) in [8]). It can be seen that if we only focus on the demands of users 1, 2, 3, it is equivalent to our considered \((N, K, M, r) = (3, 3, 1, 2)\) shared-link caching problem with correlated files. Hence, we first satisfy the demands of user 4 and then use the codes for our considered \((N, K, M, r) = (3, 3, 1, 2)\) shared-link caching problem with correlated files. Thus we transmit (with leader permutation (4, 1, 2, 3))

\[ F_{4, (1)} \oplus F_{1, (4)}, F_{4, (2)} \oplus F_{1, (4)}, F_{4, (3)} \oplus F_{3, (4)}, F_{5, (1)} \oplus F_{2, (4)}, F_{5, (2)} \oplus F_{3, (4)}, F_{5, (3)} \oplus F_{2, (4)}, F_{1, (2)} \oplus F_{1, (4)}, F_{1, (3)} \oplus F_{3, (1)}, F_{2, (2)} \oplus F_{3, (1)}, F_{2, (3)} \oplus F_{2, (1)}, F_{3, (3)} \oplus F_{3, (2)} \]

Hence, the load is 11/4 which coincides with the converse bound under the constraint of MAN placement in [8], while the proposed caching scheme in [8] achieves 3.

3) \( d_1 = \{F_1, F_2\}, d_2 = \{F_1, F_3\}, d_3 = \{F_1, F_4\}, \) and \( d_4 = \{F_2, F_3\} \) (case \( D_{1,7}^{r} \) in [8]). It can be seen that if we only focus on the demands of users 1, 2, 4, it is equivalent to our considered \((N, K, M, r) = (3, 3, 1, 2)\) shared-link caching problem with correlated files. Hence, by extending the proposed scheme for \((N, K, M, r) = (3, 3, 1, 2)\) shared-link caching problem with correlated files, we transmit (with leader permutation (3, 4, 1, 2))

\[ F_{1, (1)} \oplus F_{1, (3)}, F_{1, (2)} \oplus F_{1, (3)}, F_{1, (4)} \oplus F_{3, (3)}, F_{1, (1)} \oplus F_{2, (3)}, F_{1, (2)} \oplus F_{3, (3)}, F_{1, (4)} \oplus F_{2, (3)}, F_{3, (1)} \oplus F_{2, (4)}, F_{3, (2)} \oplus F_{3, (4)}, F_{2, (2)} \oplus F_{2, (4)}, F_{2, (3)} \oplus F_{1, (4)} \]

Hence, the load is 10/4 which coincides with the converse bound under the constraint of MAN placement in [8], while the proposed caching scheme in [8] achieves 11/4.

4) \( d_1 = \{F_1, F_2\}, d_2 = \{F_1, F_2\}, d_3 = \{F_1, F_3\}, \) and \( d_4 = \{F_2, F_3\} \) (case \( D_{20}^{r} \) in [8]). It can be seen this case is equivalent to our considered \((N, K, M, r) = (3, 4, 1, 2)\) shared-link caching problem with correlated files. Hence, we can directly use the proposed delivery phase in this paper to transmit the linear combinations (with leader permutation (1, 3, 4))

\[ F_{1, (2)} \oplus F_{1, (1)}, F_{1, (3)} \oplus F_{1, (1)}, F_{1, (4)} \oplus F_{3, (1)}, F_{2, (2)} \oplus F_{2, (1)}, F_{2, (3)} \oplus F_{3, (1)}, F_{2, (4)} \oplus F_{2, (1)}, F_{3, (2)} \oplus F_{2, (3)}, F_{3, (4)} \oplus F_{3, (3)} \]

Hence, the load is 2, which coincides with the converse bound under the constraint of MAN placement in [8], while the proposed caching scheme in [8] achieves 9/4.

**APPENDIX B**

**PROOF OF LEMMA 1.1**

We focus on the cases where \( r \in \{1, 2, N - 1, N\}, \) or \( t = KMr/N \in \{1, 2, K - 1, K\}, \) or each user has a distinct request. It can be seen that for the cases where \( r \in \{1, N\}, \) or \( t = K, \) the problem is equivalent to the shared-link caching scheme for single request proposed in [1] and our proposed caching scheme is equivalent to the optimal caching scheme under the constraint of uncoded cache placement proposed in [3].

For the remaining cases, we pick leader set \( \{u_1, \ldots, u_{N(e)}(d)\} \), and denote the leader who demands file \( F_j \) by \( u_{g(i)} \). In Section IV, we decode the desired sub-blocks by one user by two different decoding methods: direct decoding (treat interference as noise) and interference alignment decoding. In Appendix B we introduce how to decode the sub-blocks with direct decoding. Notice that, the proof direct decoding is suitable for any \( t \) and any \( r \). Once each user has decoded all sub-blocks which can be decodable by direct decoding, in Appendix C we introduce how to decode the remaining sub-blocks by interference alignment.

**From the direct decoding, we have the following lemma.**

**Lemma 2. From the direct decoding, we have**

1) In each step \( j \in \min \{N - r + 1, K - t, N(e)(d)\} \) of the proposed caching scheme in Section IV-B for each \( J \subseteq [K] \) where \( i \in J \) and \( |J| = t + 1 \) and each group \( G_{J, B} \) from (25), each user in \( J \) can decode all the sub-blocks in (25) which it has not cached or decoded previously.

2) For each user \( k' \in [K] \), at the end of each step \( j \in \min \{N - r + 1, K - t, (d'a_{d' - 1})\} \) of the proposed caching scheme in Section IV-B user \( k' \) can decode \( W_{S_1, V_1} \) where \( d' \in S_1 \) and \( \{u_{1}, \ldots, u_{j}\} \cap V \neq \emptyset \).

3) For each user \( q \in [K] \), at the end of each step \( j \in \min \{N - r + 1, K - t, g(d_q) - 1\} \) of the proposed caching scheme in Section IV-B user \( q \) can decode \( W_S \), where \( d_q \in S \) and \( \{u_{1}, \ldots, u_{j}\} \cap S \neq \emptyset \).

**Proof:** We first focus on Step 1. We focus on one group \( J \subseteq [K] \) where \( u_i \in J \) and \( |J| = t + 1 \), one group \( G_{J, B} \). We can express the transmitted linear combinations for this group in (25) by a coding matrix denoted by \( C_{J, B}^{T} \), where each column corresponds to one transmitted sub-block appearing in (25). Notice that the columns for the sub-blocks in the same block are the same by our construction. Hence, for simplicity, the column for one block means the column for each of its appearing sub-blocks in (25).

In (25), the number of sub-blocks which each user in \( J \) has not cached, is \( \frac{n_{J, B}^1}{r - |B|} + 1 \), which is also the number of linear combinations in (25). For user \( u_i \), these \( n_{J, B}^1 \) sub-blocks are in different lines of the coding matrix for
this group. Hence, user $u_1$ can recover all of these sub-blocks. For any other user $k'' \in (\mathcal{J} \setminus \{u_1\})$, we should prove that the columns, each of which represents one demanded sub-block by user $k''$ in (25), are independent. The column for each block $W_{S'}$ where $\{u_1, k''\} \subseteq S'$, is a standard basis. It can also be checked in (25) that, each standard basis for the block $W_{S'}$ where $d_{u_1} \in S'$ and $d_{k''} \notin S'$ can be obtained by the sum of the columns for $W_{(S' \cup \{d_{k''}\}) \setminus \{u_1\}}$ where $i_1 \in S'$. So we prove the independence of the columns, each of which represents one demanded sub-block by user $k''$ in (25). Hence, user $k''$ can recover all of these sub-blocks in (25). In addition, each of these desired sub-blocks of user $k''$ are known by users in $\mathcal{J} \setminus \{k''\}$, which includes user $u_1$. Hence, we prove that after considering all sets of users $\mathcal{J} \subseteq [K]$, where $u_1 \in \mathcal{J}$ and $|\mathcal{J}| = t + 1$, and all groups in Step 1, each user $k' \in ([K] \setminus \{u_1\})$ can recover each sub-block $W_{S_k, V_1}$ where $d_{k'} \in S_1$ and $u_1 \in V_1$. We then focus on user $u_2$, whose demanded file is in $[N] \setminus \{d_{u_1}\}$, and one sub-block $W_{S_2, V_2}$ where $\{d_{u_1}, d_{u_2}\} \subseteq S_2$ and $\{u_1, q\} \cap V_2 = \emptyset$. We can prove that user $u_2$ can decode $W_{S_2, V_2}$ from the linear combinations for the group $\mathcal{G}_{2}^{1}(u_1, S_2) \setminus \{d_{u_2}(V_2 \setminus \{u_1\})\}(d_{u_2})$. More precisely, if there does not exist any user in $V_2$ who also demands $F_{d_{u_2}}$, we must have $d_{q} \in (S_2 \setminus \{k_1 \in (V_2 \setminus \{u_1\})\}(d_{k_1}))$. Hence, each sub-block in the linear combinations for this group is desired by user $q$. In addition, among all sub-blocks, there are $n_{T, B}^{1}$ sub-blocks which are not known by user $u_1$. Previously, we proved user $u_2$ can decode each sub-block $W_{S_2, V_1}$ where $d_{q} \in S_1$ and $u_1 \in V_1$. Hence, user $q$ knows all sub-blocks in the linear combinations for this group, except the above $n_{T, B}^{1}$ ones which are not known by user $u_1$. So as user $u_2$, user $q$ can decode these $n_{T, B}^{1}$ sub-blocks that contain $W_{S_2, V_2}$. If there exists some user in $V_2$ who also demands $F_{d_{u_2}}$, we assume the column of block $W_{S_2, V_2}$, and the group with the same standard basis as the block $W_{(S \setminus \{d_{u_1}\}) \cup d_{u_2}}$ is full-rank, such that user $k''$ can decode all of the desired sub-blocks in the linear combinations for this group, which it has not cached nor decoded. So we prove the first part of Lemma 2 and the second part of Lemma 2.

From the second and third parts of Lemma 2 at the end of Step $\min\{N - r + 1, K - t, j - 1\}$, it can be seen that for each leader $u_j$ where $j \in [Nc(d)]$, it can decode its desired sub-blocks $W_{S_1, V_1}$ where $d_{u_j} \in S_1$ and $\{u_1, \ldots, u_{\min\{N - r + 1, K - t, j - 1\}}\} \cap V_1 \neq \emptyset$, and its desired blocks $W_S$ where $d_{u_j} \in S$ and $\{d_{u_1}, \ldots, d_{\min\{N - r + 1, K - t, j - 1\}}\} \cap S \neq \emptyset$. Hence, if $j > \min\{N - r + 1, K - t\}$, we have for each desired sub-block of leader $u_j$, it is either in the file $F_i$ where $i \in \{d_{u_1}, \ldots, d_{\min\{N - r + 1, K - t, j - 1\}}\}$, or cached by some leaders in $\{u_1, \ldots, u_{\min\{N - r + 1, K - t\}}\}$, and thus leader $u_j$ can recover all its desired sub-blocks; otherwise, by the first part of Lemma 2, it can decode all remaining desired sub-blocks in Step $j$. Hence, we prove Lemma 2.

Appendix C
Proof of Lemma 2

By direct decoding, each non-leader $k$, from the second and third parts of Lemma 2 at the end of Step $\min\{N - r + 1, K - t, g(d_k) - 1\}$, can decode its desired sub-blocks $W_{S_1, V_1}$ where $d_k \in S_1$ and $\{u_1, \ldots, u_{\min\{N - r + 1, K - t, g(d_k) - 1\}}\} \cap V_1 \neq \emptyset$, and its desired blocks $W_S$ where $d_k \in S$ and $\{d_{u_1}, \ldots, d_{\min\{N - r + 1, K - t, g(d_k) - 1\}}\} \cap S \neq \emptyset$. Hence, if $g(d_k) > \min\{N - r + 1, K - t\}$, we have for each desired sub-block of user $k$, it is either in the file $F_i$ where $i \in \{d_{u_1}, \ldots, d_{\min\{N - r + 1, K - t, g(d_k) - 1\}}\}$, or cached by some leaders in $\{u_1, \ldots, u_{\min\{N - r + 1, K - t\}}\}$, and thus user $k$ can recover all its desired sub-blocks. In the rest of the proof, we focus on the second case where $g(d_k) \leq \min\{N - r + 1, K - t\}$. We divide the proof for the interference alignment decoding into four cases, where $r = 2$, $N = 1$, $t = 1$, and $t = 2$, respectively.
Notice that if there exist some files not demanded by any user (the set of such files is assumed to be $F_d$), the proposed scheme divides all blocks into $2^{|P|}$ classes. For each set $P \subseteq F_d$, the class is defined as $c_P$ and in this class we only transmit all blocks $W_g$ where $S \cap F_d = P$. For Class $c_P$, the transmission is equivalent to the one for the $(N, K, r) = (N - |F_d|, K, r - |P|)$ shared-link problem with correlated sources, where each file is demanded.

$A$. $r = 2$

Notice that the transmission for all the blocks $W_{i, i'}$ where $F_i$ is demanded by some users and $F_{i'}$ is a file not demanded by any user, is equivalent to the shared-link caching scheme with uncorrelated files in [3]. Hence, the decodability for these blocks can be proved. In the rest of this subsection, we focus on the transmission for Class $c_0$.

We first introduce the following definition. For each set of users $T \subseteq [K]$ where $|T| = t + 1$, and each set of files $H \subseteq ([N] \setminus F_d)$ where $|H| = r - 1$ and $|\cup_{k \in T} \{d_k\} \cup H| \geq r$, we define that

$$C_{T, H} := \bigoplus_{k_1 \in T} S'_{\cup \{d_k\} \cup H} \bigoplus_{|S'| = r, H \subseteq S', d_k \in S'} W_{S', T \setminus \{k_1\}}. \ (26)$$

We will prove the following lemma in Appendix D.

**Lemma 3.** If $r = 2$, from the proposed coding scheme, each user can recover $C_{T, i}$ for each set $T \subseteq [K]$ where $|T| = t + 1$, and each file $i \in ([N] \setminus F_d)$ where $|\cup_{k \in T} \{d_k\} \cup \{i\}| \geq 2$.

In the rest of this subsection, we will prove each non-leader can recover its remaining sub-blocks by using $C_{T, i}$ where $T \subseteq [K]$, $|T| = t + 1$, and each file $i \in ([N] \setminus F_d)$ where $|\cup_{k \in T} \{d_k\} \cup \{i\}| \geq 2$. In other words, one explanation of our proposed scheme in Section IV for the case $r = 2$, is that we can construct our codes by transmitting $C_{T, i}$ for each set $T \subseteq [K]$ where $|T| = t + 1$, and each file $i \in ([N] \setminus F_d)$ where $|\cup_{k \in T} \{d_k\} \cup \{i\}| \geq 2$. However, some $C_{T, i}$ can be obtained by the linear combinations of others, such that these $C_{T, i}$ are redundant. By removing the redundant $C_{T, i}$, we have our coding scheme in Section IV.

Now we focus on user $k$ which is not a leader and the leader demanding $d_k$ is $u_f$. We want to prove that it can decode $W_{\{d_k, d_{a_1}\}, V}$ where $a \in \{f + 1, \ldots, N_e(d)\}$ and $\{u_1, \ldots, u_f, k\} \cap V = \emptyset$.

In the following, we prove user $k$ can decode $W_{\{d_k, d_{a_1}\}, V}$ from the second part of Lemma 2 we see that if $\{u_1, \ldots, u_f\} \cap V \neq \emptyset$, user $k$ can decode $W_{\{d_k, d_{a_1}\}, V}$. So we consider $\{u_1, \ldots, u_f\} \cap V = \emptyset$. We first focus on the case where $u_{f+1} \in V$. From $C_{V \cup \{k\}, \{d_k\}}$, we have that user $k$ has cached or decoded all sub-blocks in $C_{V \cup \{k\}, \{d_k\}}$ except $W_{\{d_k, d_{a_1}\}, V}$ for $i_1 \in (\cup_{k \in V} \{d_k\} \setminus \{d_{a_1}, \ldots, d_{f}\})$. In addition, for each $i_2 \in (\cup_{k \in V} \{d_k\} \setminus \{d_{a_1}, \ldots, d_{f}\})$ where $i_2 \neq d_{u_{f+1}}$, by the first part of Lemma 2 user $k$ can recover $W_{\{d_k, i_2\}, V}$ from the linear combinations for the group $G_{V \cup \{k\}, V \cup \{d_k\}}$. Hence, user $k$ can recover each sub-block $W_{\{d_k, d_{u_{f+1}}\}, V}$ where $u_{f+1} \in V$. We then focus on the case where $u_{f+1} \notin V$ and $h \notin V$, where $h \notin \{u_1, \ldots, u_{f+1}\}$. From $C_{V \cup \{u_{f+1}\}, \{d_k\}}$, it can be seen that for each sub-block $W_{\{d_k, i_3\}, V}$ where $u_{f+1} \in V$ and $i_3 \in \{d_{a_1}, \ldots, d_{u_{f+1}}\}$, user $k$ has already decoded it. For each sub-block $W_{\{d_k, i_4\}, V}$ where $u_{f+1} \in V$ and $i_4 \notin \{d_{a_1}, \ldots, d_{u_{f+1}}\}$, user $k$ can decode it by the first part of Lemma 2 from the linear combinations for the group $G_{V \cup \{k\}, (i_4) \cup (\cup_{k \in V} \{d_k\})}$. Hence, we prove user $k$ can decode all sub-blocks in $C_{V \cup \{u_{f+1}\}, \{d_k\}}$ except $W_{\{d_k, d_{u_{f+1}}\}, V}$. Thus user $k$ can decode $W_{\{d_k, d_{u_{f+1}}\}, V}$. So we prove user $k$ can decode all sub-blocks $W_{\{d_k, d_{u_{f+1}}\}, V}$.

We then focus on $W_{\{d_k, d_{u_{f+1}}\}, V}$. It is easy to see if $\{u_1, \ldots, u_f\} \cap V \neq \emptyset$, user $k$ can decode it. Moreover, we can use the above method to prove that from $C_{V \cup \{k\}, \{d_k\}}$, user $k$ can decode all sub-blocks $W_{\{d_k, d_{u_{f+1}}\}, V}$ where $u_{f+1} \notin V$. Then, we can prove from $C_{V \cup \{u_{f+1}\}, \{d_k\}}$, user $k$ can decode all sub-blocks $W_{\{d_k, d_{u_{f+1}}\}, V}$ where $u_{f+1} \notin V$.

In conclusion, we can prove each user which is not a leader, can also decode all its desired sub-blocks.

$B$. $t = 2$

Now we focus the case where $t = 2$. In the proof, we only consider Class $c_0$. For other class $c_P$ where $P \subseteq F_d$, the proposed scheme is equivalent to the one for the $(N, K, r) = (N - |F_d|, K, r - |P|)$ shared-link problem with correlated sources, where each file is demanded, and $t = 2$. Since the cases where $r \leq 2$ have been proved, we consider $r \geq 3$.

We also focus on user $k$ which is not a leader and the leader demanding $d_k$ is $u_f$. We want to prove that it can decode all sub-blocks $W_{S, V}$ where $S \cap \{d_{a_1}, \ldots, d_{u_{f+1}}\} = \emptyset$, $d_k \in S$, and $\{u_1, \ldots, u_k, f\} \cap V = \emptyset$. We divide these sub-blocks into three hierarchies.

$a$) Hierarchy 1: $(\cup_{k \in V} \{d_k\}) \subseteq S$. We consider the linear combination for the group $G_{S \cup \{u_f\}, S \cup \{u_1, \ldots, u_{f+1}\}}$ including the sub-block $W_{S, V}$. It can be seen in this linear combination, all sub-blocks are from the block $W_S$. In addition, user $u_f$ caches all sub-blocks in this linear combination except $W_{S, V}$. Since user $k$ has decoded each sub-block of $W_S$ which is cached by user $u_f$, user $k$ can decode $W_{S, V}$.

$b$) Hierarchy 2: $(\cup_{k \in V} \{d_k\}) \cap (S \setminus \{d_k\}) = \emptyset$. So for each group containing $W_{S, V}$, assumed to be $G'$ in $T'$, we have $|B'| = r - 1$. From the similar proof of Lemma 3 we can prove each user can recover $C_{T, H}$ where $H \subseteq ([N] \setminus F_d)$, $|H| = r - 1$, $T \subseteq [K]$, $|T| = t + 1$, and $H \cap (\cup_{k \in T} \{d_k\}) = \emptyset$. Notice that each desired sub-block of user $k$ in Hierarchy 2 is in some $C_{T, H}$.

We then follow the similar method proposed in $r = 2$ to prove that each user can recover its desired sub-blocks in Hierarchy 2.

$c$) Hierarchy 3: The sub-blocks neither in Hierarchy 1 nor Hierarchy 2. When $t = 2$, for each $W_{S, V}$ in Hierarchy 3, we have $|\cup_{k \in V} \{d_k\} \cup \{d_k\}| = t + 1 = 3$ and $|S \cap (\cup_{k \in V} \{d_k\} \cup \{d_k\})| = 2$. Notice that if $|\cup_{k \in V} \{d_k\} \cup \{d_k\}| < 3$, $W_{S, V}$ is either in Hierarchy 1 or in Hierarchy 2. If $|S \cap (\cup_{k \in V} \{d_k\} \cup \{d_k\})| = 1$, $W_{S, V}$ is in Hierarchy 2. If $|S \cap (\cup_{k \in V} \{d_k\} \cup \{d_k\})| = 3$, $W_{S, V}$ is in Hierarchy 1.
It can also be seen that for each desired sub-block of user $k$ in Hierarchy 3, it must be in the linear combinations for some group $G_f^j$, $B'$ where the users in $j'$ have different demands, $y_j \in j'$, and $|B'| = r - 2$. Hence, in the following, we prove for each group $G_f^{j_1,B_1}$ where the users in $j_1$ have different demands, $y_{j_1} \in j_1$, and $|B_1| = r - 2$, user $k$ can decode all sub-blocks of file $F_{d_k}$ appearing in the linear combinations of this group. We assume that $j_1 = \{u_j, y_1, y_2\}$.

We first focus on the case where $j_1$ contains some users whose demand is in $\{d_{u_1}, \ldots, d_{u_{r-1}}\}$. If both of the users $y_1$ and $y_2$ have demands in $\{d_{u_1}, \ldots, d_{u_{r-1}}\}$, by the construction of (25), there is no linear combination in $G_f^{j_1,B_1}$. Hence, we consider that only user $y_1$ demands a file in $\{d_{u_1}, \ldots, d_{u_{r-1}}\}$. In this case, $G_f^{j_1}$ only contains one linear combination.

From the similar proof as (25), we have

\[
C_{j_1,B_1 \cup \{u_j\}} = \text{Sum}(G_f^{j_1,B_1}) \oplus \text{Sum}(G_f^{y_1(d_{u_1})}) \oplus C_{j_1 \cup \{y_1(d_{u_1})\} \cup \{y_2, B_1 \cup \{d_{u_2}\}}),
\]

where $u_j(d_{y_1})$ is the leader demanding $F_{d_{y_1}}$ and we define $\text{Sum}(G_f^{j,b})$ as the binary sum of all the linear combinations in (25) for the group $G_f^{j,b}$. $C_{j_1 \cup \{y_1(d_{u_1})\} \cup \{y_2, B_1 \cup \{d_{u_2}\}}$ is the linear combination for the group $G_f^{y_1(d_{u_1})} \cup \{y_2, B_1 \cup \{d_{u_2}\}}$ including the sub-blocks of block $W_{\{u_j(d_{y_1})\} \cup \{y_2, B_1 \cup \{d_{u_2}\}}$. From $C_{j_1,B_1 \cup \{u_j\}}$, there are two sub-blocks of $W_{B_1 \cup \{u_j\} \cup \{d_{u_1}\}}$ which have been decoded by user $k$, and two sub-blocks of $W_{B_1 \cup \{u_j,d_{y_1}\}}$, one of which is cached by user $y_1$. In addition, by the second part of Lemma 2, user $k$ has decoded the sub-blocks of $W_{B_1 \cup \{u_j,d_{y_1}\}}$ cached by user $y_1$. Hence, user $k$ can decode all sub-blocks in this group.

We then focus on the case where $j_1$ does not contain any users whose demand is in $\{d_{u_1}, \ldots, d_{u_{r-1}}\}$. In this case, $G_f^{j_1,B_1}$ contains two linear combinations while the interferences to user $k$ are from sub-blocks of $W_{\{d_{u_1},d_{u_2}\}}$. The first linear combination is,

\[
W_{\{d_{u_1},d_{u_2}\}} \oplus W_{\{d_{u_1},d_{y_1}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_2}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}}.
\]

and the second linear combination is

\[
W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}} \oplus W_{\{d_{u_2},d_{y_2}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}} \oplus W_{\{d_{u_2},d_{y_2}\}}.
\]

We use the induction method, started by considering each group $G_f^{j_1,B_1}$, where $u_{j+1} \in j_1$. Assuming $y_1 = u_{j+1}$ in (29), $W_{\{d_{u_1},d_{y_2}\}}$ has already been decoded by user $k$. In addition, by the first part of Lemma 2, we can see that $W_{\{d_{u_1},d_{y_2}\}}$ can be decoded by user $k$ from the linear combinations for the group $G_f^{y_1,y_2}$. Hence, user $k$ can decode the interference $W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_2}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}}$ from (29). Then it can decode $W_{\{d_{u_1},d_{y_1}\}}$ in (28).

We then consider each group $G_f^{j_1,B_1}$, where $d_{u_{j+1}} \in B_1$. It can be seen that user $k$ can decode the interference to it in the linear combinations for the group $G_f^{\{u_j,d_{y_1},d_{y_2}\} \cup \{B_1 \backslash \{d_{u_{j+1}}\}\}}$ which is

\[
W_{\{d_{y_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_2}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}}.
\]

It can be seen that user $k$ can decode the interference to it in the linear combinations for the group $G_f^{\{y_1,u_j,d_{y_2}\} \cup \{B_1 \backslash \{d_{u_{j+1}}\}\}}$ which is

\[
W_{\{d_{y_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_2}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}}.
\]

By summing (30) and (31), user $k$ can decode $W_{\{d_{y_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_2}\}} \oplus W_{\{d_{u_1},d_{y_2}\}} \oplus W_{\{d_{u_2},d_{y_1}\}} \oplus W_{\{d_{u_1},d_{y_1}\}}$, which is the interference to user $k$ in (28) and (29). Hence, user $k$ can decode all sub-blocks of file $F_{d_k}$ appearing in group $G_f^{j_1,B_1}$.

Next, we consider each group $G_f^{j_1,B_1}$ where $y_1$ is not a leader and $y_1$ demands $F_{d_{u_{j+2}}}$. By the similar method as above, we can prove user $k$ can decode the interference to it in this group, by summing the interferences in the groups $G_f^{\{u_j,y_1,d_{y_1}\} \cup \{B_1 \backslash \{d_{u_{j+1}}\}\}} \cup \{d_{u_{j+1}}\}$. Notice the decoding for $G_f^{\{u_j,y_1,d_{y_1}\} \cup \{d_{u_{j+1}}\}}$ is in Hierarchy 2.

Similarly, we then consider $G_f^{j_1,B_1}$, where $d_{u_{j+2}} \in B_1$. We do the same procedure step-by-step until the last leader. Hence, we can prove user $k$ can decode all its desired sub-blocks in Hierarchy 3.

C. $t \in \{1, K - 1\}$ or $t = N - 1$

Notice that for any $t$ and $r$, the decoding for Hierarchies 1 and 2 is using the same procedure described above. So the difficulty is to decode the sub-blocks in Hierarchy 3. When $r = 2$, it can be seen that the decoding method for the three hierarchies is the same. For the case where $t = 1$ or $r = N - 1$, the proof is similar to the one for $t = 2$. When $t = 1$, the sub-blocks desired by each user are divided into two hierarchies, where there is no Hierarchy 3. When $r = N - 1$, for each non-leader $k$, there is only one block which user $k$ does not intend. Hence, we can use the similar method for Hierarchy 3 of $t = 2$ to let user $k$ cancel (or align) the interferences and decode the desired sub-blocks in Hierarchy 3.

For the case where $t = K - 1$, it can be seen that there is only one step and each user in the same considered set $J = [K]$. Hence, by the first part of Lemma 2 each user can recover its desired sub-blocks.

Appendix D

Proof of Lemma 3

We want to prove each user can recover $C_{T,i}$ for each set $T \subseteq [K]$ where $|T| = t + 1$, and each file $i \in ([N] \setminus F_d)$ where $|U_{k \in T} \{d_{k}\} \cup \{i\}| \geq 2$.

We first focus on $T$ where $u_1 \in T$. In this case, there exists some users in $T$ who does not demand $F_{d_{u_1}}$.

If $i \in U_{k \in T} \{d_{k}\}$, we have two cases: $i = d_{u_1}$ and $i \neq d_{u_1}$. For the case where $i = d_{u_1}$, we can sum all linear combinations in (25) for the group $\mathcal{J} = T$ and $\mathcal{B} = \emptyset$. With $\mathcal{J} = T$ and $\mathcal{B} = \emptyset$, it can be seen for each sub-block in (25)
which is in the file $F_{d_{u_1}}$ appears in one linear combination of (25), while each sub-block in (25) which is not in the file $F_{d_{u_1}}$ appears in two linear combinations of (25). So if we sum all linear combinations in (25) for the group $J = T$ and $B = \emptyset$, we have $C_{T, (i)}$. For the case where $i \neq d_{u_1}$, $C_{T, (i)}$ is equal to the linear combination in (25) for the group $J = T$ and $B = \emptyset$, which includes the sub-block $W_{\{d_{u_1}, i\}\cap \{u_1\}}$.

If $i \not\in \cup k_2 \in T \{d_{k_2}\}$, $C_{T, (i)}$ is equal to the linear combination for the group $G^i_{T, (i)}$.

Let us then focus on $C_{T, (d_{u_1})}$ where $u_1 \notin T$. In the following, we will prove

$$ C_{T, (d_{u_1})} = \bigoplus_{k_2 \in T} C_i (T \cup \{u_1\}) \{k_2\}, \{d_{u_2}\}. \tag{32} $$

**Proof:** To prove (32), it is equivalent to prove that

$$ C_{T, (d_{u_1})} \bigoplus_{k_2 \in T} C_i (T \cup \{u_1\}) \{k_2\}, \{d_{u_2}\} = 0. \tag{33} $$

It can be seen that for each sub-block $W_{S,V}$ appearing in each $C_{T, (i)}$ appearing (33), we have $T \supseteq V$ and $|T| - |V| = 1$.

We focus on one sub-block $W_{S,V}$ appearing in (33) and assume that $W_{S,V}$ appears in $C_{T, (i)}$. Since $|T \cup \{u_1\}| - |V| = 2$, $W_{S,V}$ can only appear in another $C_{T, (i)}$ where $T_2 \supseteq V$ besides $C_{T, (i)}$. Thus the only possibility is that $T_2 = \{T \cup \{u_1\}\} \setminus \{T \setminus V\}$ and $\{d_{k_1}\}$ is the user in $T \setminus V$. In other words, each sub-block appearing twice in (33) and thus we prove (33).

In the following, we focus on $C_{T, (i)}$ where $u_1 \notin T$, $u_2 \in T$, and $i \neq d_{u_1}$. If $u_1 \notin \cup k_2 \in T \{d_{k_2}\}$, it is easy to see each $C_{T, (i)}$ can be recovered by the same method as $u_1 \in T$. So we focus on that $u_1 \notin \cup k_2 \in T \{d_{k_2}\}$. We have three cases: $i \in \cup k_2 \in T \{u_1\} \{d_{k_2}\}$, $i \notin \cup k_2 \in T \{d_{k_2}\}$, and $i = d_{u_2}$. If $i \notin \cup k_2 \in T \{u_1\} \{d_{k_2}\}$, $C_{T, (i)}$ is equal to the linear combination in (25) for the group $G^i_{T, (i)}$, which includes the sub-block $W_{\{d_{u_1}, i\}\cap \{u_1\}}$. If $i \notin \cup k_2 \in T \{d_{k_2}\}$, $C_{T, (i)}$ is equal to the linear combination for the group $G^i_{T, (i)}$. Lastly, we focus on $i = d_{u_2}$. If $k_2 \not\in \cup \{d_{k_2}\}, \{d_{u_1}, i\}\cap \{u_1\}$ or $k_1 \not\in \cup \{d_{k_2}\}$, $W_{\{d_{u_1}, i\}\cap \{u_1\}}$. In the previous cases, we have

$$ C_{T, (d_{u_1})} \bigoplus_{k_2 \in T} C_i (T \cup \{u_1\}) \{k_2\}, \{d_{u_2}\}, \{d_{u_1}, i\}\cap \{u_1\} = 0. \tag{36} $$

For leader $u_j$, the case where $i = d_{u_2}$ and $\cap \{u_1, u_2\} = \emptyset$, is equivalent to the one where $i = d_{u_1}$ and $u_1 \notin T$. Hence, we prove Lemma 3.

**REFERENCES**

[1] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” IEEE Trans. Inf. 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 Info. 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. Ekrip, “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. Tulino, J. Llorca, and G. Caire, “Caching-aided coded multicasting with multiple random requests,” in Proc. IEEE Inf. Theory Workshop (IFW), May 2015.

[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. Abajbuhofa, 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] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” IEEE Trans. Inf. Theory, vol. PP (99), Sep. 2017.

[11] K. Wan, D. Tuninetti, and P. Piantanida, “On caching with more users than files,” in IEEE Int. Symp. Inf. Theory, Jul. 2016.

[12] M. Ji, G. Caire, and A. Molisch, “Fundamental limits of caching in wireless d2d networks,” IEEE Trans. Inf. Theory, vol. 62, no. 1, pp. 849–869, 2016.