Cache-Aided Scalar Linear Function Retrieval

Kai Wan*, Hua Sun†, Mingyue Ji‡, Daniela Tuninetti§, Giuseppe Caire*
*Technische Universität Berlin, 10587 Berlin, Germany, {kai.wan, caire}@tu-berlin.de
†University of North Texas, Denton, TX 76203, USA, hua.sun@unt.edu
‡University of Utah, Salt Lake City, UT 84112, USA, mingyue.ji@utah.edu
§University of Illinois at Chicago, Chicago, IL 60607, USA, danielat@uic.edu

***Abstract*—In the shared-link coded caching problem, formulated by Maddah-Ali and Niesen (MAN), each cache-aided user demands one file (i.e., single file retrieval). This paper generalizes the MAN problem so as to allow users to request *scalar linear functions* (aka, linear combinations with scalar coefficients) of the files. We propose a novel coded delivery scheme, based on MAN uncoded cache placement, that allows for the decoding of arbitrary scalar linear functions of the files on arbitrary finite fields. Surprisingly, it is shown that the load for cache-aided scalar linear function retrieval depends on the number of linearly independent functions that are demanded, akin to the cache-aided single-file retrieval problem where the load depends on the number of distinct file requests. The proposed scheme is proved to be optimal under the constraint of uncoded cache placement, in terms of worst-case load, and within a factor 2 otherwise.**

I. Introduction

Coded caching is a promising technique to smooth the network traffic by storing parts of content in the library at the users' caches. The seminal work on coded caching by Maddah-Ali and Niesen (MAN) [1] uses a combinatorial design in the placement phase (referred to as MAN cache placement), such that in the delivery phase binary messages (referred to as MAN multicast messages) can simultaneously satisfy the demands of users. Under the constraint of uncoded cache placement (i.e., each user directly caches a subset of the library bits), the MAN scheme can achieve the minimum worst-case load among all possible demands when there are less users than files [2]. On the observation that if there are files demanded multiple times some MAN multicast messages can be obtained as a binary linear combination of other MAN multicast messages, Yu, Maddah-Ali, and Avestimehr (YMA) proposed a delivery scheme that avoid the transmission of 'redundant' MAN multicast messages [3]. The YMA delivery achieves the minimum worst-case load under the constraint of uncoded cache placement in all regimes. The load cost of uncoded cache placement compared to coded cache placement is no more than a multiplicative factor of 2 [4].

In general, linear and multivariate polynomial operations are widely used fundamental primitives for building the complex queries that support on-line analytics and data mining procedures. This paper studies the fundamental tradeoff between local storage and network load when users are interested in retrieving a function of the dataset available at the server. The question we ask in this paper is, compared to the original MAN caching problem, whether the optimal worst-case load is increased when the users are allowed to request *scalar linear functions of the files* – the first non-trivial extension of the MAN single-file-retrieval problem, on the way to understand the problem of retrieving general functions. The original MAN shared-link caching problem, where each user request one file, is thus a special case of the formulated shared-link cache-aided scalar linear function retrieval problem.

In addition to the novel problem formulation, our main results are as follows:

1) **Achievable scheme for demanded functions on the binary field.** We start by considering the case of scalar linear functions on the binary field. Based on the YMA delivery, which can be thought of as interference cancellation, we propose a novel delivery scheme whose key idea is to deliver only the largest set of linearly independent functions, while the remaining ones can be reconstructed by proper linear combinations of those already retrieved. This can be thought of as the generalization of the idea to only deliver the files requested by the "leader users" in the YMA delivery [3].
2) **Generalization to demanded functions on arbitrary finite field.** We then generalize the proposed scheme to the case where the demands are scalar linear functions on an arbitrary finite field. To the best of our knowledge, even for the originally MAN coded caching problem, no caching scheme is known in the literature for arbitrary finite field, which achieves the same load as the YMA scheme. *Compared to the YMA scheme, we use different multicast message encoding (inspired by interference alignment) and decoding procedures that work on an arbitrary finite field, which are highly non-trivial and not from direct extensions from the YMA scheme.* Interestingly, the achieved load of our proposed scheme only depends on the number of linearly independent functions that are demanded, akin to the YMA's cache-aided single-file retrieval scheme where the load depends on the number of distinct file requests.
3) **Optimality.** On observation that any converse bound for the original MAN caching problem is also a converse in the considered cache-aided function retrieval problem, we prove that the proposed scheme achieves the optimal worst-cast load under the constraint of uncoded cache placement. Moreover, the achieved worst-case load of the proposed scheme is also proved to be order optimal in general within a factor of two.

978-1-7281-6432-8/20/$31.00 ©2020 IEEE

From the results in this paper, we can answer the question we asked at the beginning of this paper: *the optimal worst-case load under the constraint of uncoded cache placement is not increased when the users are allowed to request scalar linear functions of the files.* The key idea of this paper can be extended to cache-aided device-to-device [5], [6] and/or demand-private [7], [8] settings.

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; $\mathbb{F}_q$ represents a finite field with order q; $\text{rank}_q(\mathbb{A})$ represents the rank of matrix $\mathbb{A}$ on field $\mathbb{F}_q$; $\det(\mathbb{A})$ represents the determinant matrix $\mathbb{A}$; $\mathbb{A}_{\mathcal{S},\mathcal{V}}$ represents the sub-matrix of $\mathbb{A}$ by selecting from $\mathbb{A}$, the rows with indices in $\mathcal{S}$ and the columns with indices in $\mathcal{V}$. we let $\binom{x}{y} = 0$ if $x < 0$ or $y < 0$ or $x < y$. In this paper, for each set of integers $\mathcal{S}$, we sort the elements in $\mathcal{S}$ in an increasing order and denote the i^{th} smallest element by $\mathcal{S}(i)$, i.e., $\mathcal{S}(1) < \ldots < \mathcal{S}(|\mathcal{S}|)$.

II. System Model

A $(\mathsf{K}, \mathsf{N}, \mathsf{M}, \mathsf{q})$ shared-link cache-aided scalar linear function retrieval problem is defined as follows. A central server has access to a library of N files. The files are denoted as $F_1, \ldots, F_{\mathsf{N}}$. Each file has B independent and uniformly distributed symbols over a finite field $\mathbb{F}_{\mathsf{q}}$, for some prime-power q. The central server is connected to K users through an error-free shared-link. Each user is equipped with a cache that can store up to MB symbols, where $\mathsf{M} \in [0, \mathsf{N}]$.

The system operates in two phases.

Cache Placement Phase. During the cache placement phase, each user stores information about the N files in its local cache without knowledge of future users' demands, that is, there exist placement functions $\phi_k,\ k \in [\mathsf{K}]$, such that

$$\phi_k : [\mathbb{F}_{\mathsf{q}}]^{\mathsf{BN}} \to [\mathbb{F}_{\mathsf{q}}]^{\mathsf{BM}}, \tag{1}$$

We denote the content in the cache of user $k \in [\mathsf{K}]$ by $Z_k = \phi_k(F_1, \ldots, F_{\mathsf{N}})$.

Delivery Phase. During the delivery phase, each user requests one scalar linear function of the files. The demand of user $k \in [\mathsf{K}]$ is represented by the row vector $\mathbf{y}_k = (y_{k,1}, \ldots, y_{k,\mathsf{N}}) \in [\mathbb{F}_{\mathsf{q}}]^{\mathsf{N}}$, i.e., that user k wants to retrieve

$$B_k := y_{k,1}F_1 + \ldots + y_{k,\mathsf{N}}F_{\mathsf{N}}. \tag{2}$$

We denote the demand matrix of all users by the Matlab-like notation

$$\mathbb{D} = [\mathbf{y}_1; \ldots; \mathbf{y}_{\mathsf{K}};] \in [\mathbb{F}_{\mathsf{q}}]^{\mathsf{K}\times\mathsf{N}}. \tag{3}$$

In addition, for each set $\mathcal{S} \subseteq [\mathsf{K}]$, we denote the demand matrix of the users in $\mathcal{S}$ by $\mathbb{D}_{\mathcal{S}} := \mathbb{D}_{\mathcal{S},[\mathsf{N}]}$.

Given the demand matrix $\mathbb{D}$, the server broadcasts the message $X = \psi(\mathbb{D}, F_1, \ldots, F_{\mathsf{N}})$ to each user $k \in [\mathsf{K}]$, where the encoding function ψ is such that

$$\psi : [\mathbb{F}_{\mathsf{q}}]^{\mathsf{KN}} \times [\mathbb{F}_{\mathsf{q}}]^{\mathsf{BN}} \to [\mathbb{F}_{\mathsf{q}}]^{\mathsf{BR}}, \tag{4}$$

for some non-negative R.

Decoding. Each user $k \in [\mathsf{K}]$ decode its desired function from $(\mathbb{D}, Z_k, X)$. In other words, there exist decoding functions $\xi_k,\ k \in [\mathsf{K}]$, such that

$$\xi_k : [\mathbb{F}_{\mathsf{q}}]^{\mathsf{KN}} \times [\mathbb{F}_{\mathsf{q}}]^{\mathsf{BM}} \times [\mathbb{F}_{\mathsf{q}}]^{\mathsf{BR}} \to [\mathbb{F}_{\mathsf{q}}]^{\mathsf{B}}, \tag{5}$$

$$\xi_k(\mathbb{D}, Z_k, X) = y_{k,1}F_1 + \ldots + y_{k,\mathsf{N}}F_{\mathsf{N}}. \tag{6}$$

Objective. For a given memory size $\mathsf{M} \in [0, \mathsf{N}]$, the objective is to determine the *minimum worst-case load* among all possible demands, defined as the smallest R such that there exist placement functions $\phi_k, k \in [\mathsf{K}]$, encoding function ψ, and decoding functions $\xi_k, k \in [\mathsf{K}]$, satisfying all the above constraints. The optimal load is denoted as $R^\star$.

If each user directly copies some symbols of the N files into its cache, the cache placement is said to be *uncoded*. The minimum worst-case load under the constraint of uncoded cache placement is denoted by $\mathsf{R}_{\text{u}}^\star$. Evidently $\mathsf{R}_{\text{u}}^\star \geq \mathsf{R}^\star$.

III. Main Results

The proposed caching scheme in Sections IV (for $\mathsf{q} = 2$) and V (for any prime-power q), achieves the following load.

Theorem 1 (Achievability)**.** *For the $(\mathsf{K}, \mathsf{N}, \mathsf{M}, \mathsf{q})$ shared-link cache-aided scalar linear function retrieval problem, when $\mathsf{M} = \frac{\mathsf{N}t}{\mathsf{K}}$ with $t \in [0:\mathsf{K}]$, for demand matrix $\mathbb{D}$, the load*

$$\mathsf{R}(\mathbb{D}) := \frac{\binom{\mathsf{K}}{t+1} - \binom{\mathsf{K}-\text{rank}_{\mathsf{q}}(\mathbb{D})}{t+1}}{\binom{\mathsf{K}}{t}} \tag{7}$$

is achievable. The worst-case load is attained by $\text{rank}_{\mathsf{q}}(\mathbb{D}) = \min(\mathsf{N}, \mathsf{K})$. □

Since the setting where each user demands one file is a special case of the considered cache-aided scalar linear function retrieval problem, any converse bounds for the original shared-link coded caching problem is also a converse in our considered problem. Thus we have:

Theorem 2 (Optimality)**.** *For the $(\mathsf{K}, \mathsf{N}, \mathsf{M}, \mathsf{q})$ shared-link cache-aided scalar linear function retrieval problem, the optimal worst-case load-memory tradeoff under the constraint of uncoded cache placement is the lower convex envelop of*

$$(\mathsf{M}, \mathsf{R}_{\text{u}}^\star) = \left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{\binom{\mathsf{K}}{t+1} - \binom{\mathsf{K}-\min\{\mathsf{K},\mathsf{N}\}}{t+1}}{\binom{\mathsf{K}}{t}}\right),\ \forall t \in [0:\mathsf{K}]. \tag{8}$$

Moreover, the achieved worst-case load in (8) *is optimal within a factor of* 2 *in general.* □

A. High-level ideas to derive the load in Theorem 1

File Split. Fix a $t \in [\mathsf{K}]$. **We partition the "symbol positions" set** $[\mathsf{B}]$ as follows

$$[\mathsf{B}] = \{\mathcal{I}_{\mathcal{W}} : \mathcal{W} \subseteq [\mathsf{K}], |\mathcal{W}| = t\} : |\mathcal{I}_{\mathcal{W}}| = \mathsf{B}/\binom{\mathsf{K}}{t}. \tag{9}$$

Then, with a Matlab-like notation, we let

$$F_{i,\mathcal{W}} = F_i(\mathcal{I}_{\mathcal{W}}),\ \forall \mathcal{W} \subseteq [\mathsf{K}] : |\mathcal{W}| = t,\ \forall i \in [\mathsf{N}], \tag{10}$$

representing the set of symbols of F_i whose position is in $\mathcal{I}_{\mathcal{W}}$.

Uncoded Placement. User $k \in [\mathsf{K}]$ caches $F_{i,\mathcal{W}}$ if $k \in \mathcal{W}$. As in the MAN placement, the needed memory size is

$$\mathsf{M} = \mathsf{N}\binom{\mathsf{K}-1}{t-1}/\binom{\mathsf{K}}{t} = \mathsf{N}\frac{t}{\mathsf{K}}. \tag{11}$$

By this placement, any scalar linear function B_k in (2) is naturally partitioned into "blocks" as follows

$$B_k = \{B_{k,\mathcal{W}} : \mathcal{W} \subseteq [\mathsf{K}], |\mathcal{W}| = t\}, \tag{12}$$

$$B_{k,\mathcal{W}} := y_{k,1}F_1(\mathcal{I}_{\mathcal{W}}) + \ldots + y_{k,\mathsf{N}}F_{\mathsf{N}}(\mathcal{I}_{\mathcal{W}}). \tag{13}$$

We refer to $B_{k,\mathcal{W}}$ in(13) as the $\mathcal{W}$-th block of the k-th demanded function. Some blocks of the demanded functions can thus be computed based on the cache content available at each user while the remaining ones need to be delivered by the server in the delivery phase.

Multicast Messages. With this specific file split and cache placement, **we aim to operate a YMA-like delivery over the blocks instead of over the subfiles**. More precisely, we construct multicast messages as

$$W_{\mathcal{S}} := \sum_{k\in\mathcal{S}} \alpha_{\mathcal{S},k} B_{k,\mathcal{S}\setminus\{k\}}, \forall \mathcal{S} \subseteq [\mathsf{K}] : |\mathcal{S}| = t+1, \tag{14}$$

for some $\alpha_{\mathcal{S},k} \in \mathbb{F}_{\mathsf{q}} \setminus \{0\}$ and where $B_{k,\mathcal{W}}$ was defined in (12). Clearly, this scheme achieves the same load as the MAN caching scheme.

Delivery. In order to operate YMA-like delivery, we need to identify a set of leader users $\mathcal{L} \subseteq [\mathsf{K}]$, send only the multicast messages in (14) for which $\mathcal{S} : \mathcal{S} \cap \mathcal{L} \neq \emptyset$, and make sure that all non-sent multicast messages (i.e., $\mathcal{S} : \mathcal{S} \cap \mathcal{L} = \emptyset$) can be locally reconstructed from the transmitted ones.

Clearly, if $\mathcal{L} = [\mathsf{K}]$, we get a MAN-like delivery (that works with any $\alpha_{\mathcal{S},k} \in \mathbb{F}_{\mathsf{q}} \setminus \{0\}$ in (14)) to achieve $\mathsf{R} = \binom{\mathsf{K}}{t+1}/\binom{\mathsf{K}}{t} = (\mathsf{K}-t)/(1+t)$. The question is whether we can do better.

The novelty of our novel scheme in Theorem 1 is to achieve the same load as the YMA scheme in the following way: **multicast messages are constructing by alternating the coefficients of block demanded by leaders and those demanded by non-leaders between $+1$ and -1.** We propose:

1) We choose $\text{rank}_{\mathsf{q}}(\mathbb{D})$ leaders (the leader set is denoted by $\mathcal{L}$), where the demand matrix of the leaders is full-rank.
2) We separate the blocks demanded by the leaders from those demanded by non-leaders in $W_{\mathcal{S}}$ in (14) as

$$\sum_{k\in\mathcal{S}} \alpha_{\mathcal{S},k} B_{k,\mathcal{S}\setminus\{k\}} = \sum_{k_1\in\mathcal{S}\cap\mathcal{L}} \alpha_{\mathcal{S},k_1} B_{k_1,\mathcal{S}\setminus\{k_1\}} + \sum_{k_2\in\mathcal{S}\setminus\mathcal{L}} \alpha_{\mathcal{S},k_2} B_{k_2,\mathcal{S}\setminus\{k_2\}}. \tag{15}$$

We then alternate the coefficients of the desired blocks by the leaders (i.e., users in $\mathcal{S}\cap\mathcal{L}$) between $+1$ and -1, i.e., the coefficient of the desired block of the first leader is $+1$, the coefficient of the desired block of the second leader is -1, the coefficient of the desired block of the third leader is $+1$, etc. Similarly, we alternate the coefficients of the desired blocks by the non-leaders (i.e., users in $\mathcal{S}\setminus\mathcal{L}$) between $+1$ and -1.

We note that this type of code was originally proposed for the private function retrieval problem [9], where there is a cache-less user aiming to retrieval a scalar linear function of the files stored at multiple servers (each server can access to the whole library), while preserving the demand of this user from each server.

3) With the above encoding scheme, we seek decoding coefficients $\{\beta_{\mathcal{A},\mathcal{S}}\}$ such that

$$W_{\mathcal{A}} = \sum_{\mathcal{S}\subseteq\mathcal{A}\cup\mathcal{L}:|\mathcal{S}|=t+1,\mathcal{S}\cap\mathcal{L}\neq\emptyset} \beta_{\mathcal{A},\mathcal{S}} W_{\mathcal{S}}. \tag{16}$$

holds for each $\mathcal{A} \subseteq [\mathsf{K}]$ where $|\mathcal{A}| = t+1$ and $\mathcal{A}\cap\mathcal{L} = \emptyset$. In other words, each user can recover all multicast messages $W_{\mathcal{S}}$ where $\mathcal{S} \subseteq [\mathsf{K}]$ and $|\mathcal{S}| = t+1$, and thus it can recover its desired function. One can show that each decoding coefficient $\beta_{\mathcal{A},\mathcal{S}}$ is given in (36).

The load achieved by the proposed scheme is exactly the same as that of the YMA scheme for the original coded-caching file retrieval problem.

We note that the load in (7) is a generalization of the achieved load by the YMA scheme. More precisely, if each user $k \in [\mathsf{K}]$ requests one file (i.e., $\mathbf{y}_k \in [0:1]^{\mathsf{N}}$ with a unit norm), $\text{rank}_{\mathsf{q}}(\mathbb{D})$ is exactly the number of demanded files, and thus the proposed scheme achieves the load of the YMA scheme. Interestingly, the load of the proposed scheme only depends on the rank of the demand matrix of all users, instead of on the specifically demanded functions.

IV. Novel Achievable Scheme for $\mathsf{q} = 2$

In the following, we describe the proposed scheme when the demands are scalar linear functions on $\mathbb{F}_2$. We use the file split in (9)-(10), resulting in the demand split in (12).

In the delivery phase, the demand matrix $\mathbb{D}$ is revealed to all users, where each element in $\mathbb{D}$ is either 0 or 1. Among the K users we first choose $\text{rank}_2(\mathbb{D})$ leaders (assume the set of leaders is $\mathcal{L} = \{\mathcal{L}(1), \ldots, \mathcal{L}(|\mathcal{L}|)\}$), where

$$|\mathcal{L}| = \text{rank}_2(\mathbb{D}_{\mathcal{L}}) = \text{rank}_2(\mathbb{D}). \tag{17}$$

Encoding. We focus on each set $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t+1$, and generate the multicast message in (14) with $\alpha_{\mathcal{S},k} = 1$.

Delivery. The server broadcasts $W_{\mathcal{S}}$ for each $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t+1$ and $\mathcal{S}\cap\mathcal{L} \neq \emptyset$.

Decoding. For each set of users $\mathcal{B} \subseteq [\mathsf{K}]$, let $\mathscr{V}_{\mathcal{B}}$ be the family of subsets $\mathcal{V} \subseteq \mathcal{B}$, where $|\mathcal{V}| = |\mathcal{L}|$ and $\text{rank}_2(\mathbb{D}_{\mathcal{V}}) = |\mathcal{L}|$. We now consider each set $\mathcal{A} \subseteq [\mathsf{K}]$ where $|\mathcal{A}| = t+1$ and $\mathcal{A}\cap\mathcal{L} = \emptyset$, and focus on the binary sum with $\mathcal{B} = \mathcal{L}\cup\mathcal{A}$,

$$\bigoplus_{\mathcal{V}\in\mathscr{V}_{\mathcal{B}}} W_{\mathcal{B}\setminus\mathcal{V}}. \tag{18}$$

A subfile $F_{i,\mathcal{W}}$ appears in the sum (18) if and only if $\mathcal{W} \subseteq \mathcal{B}$ and there exists some user $k \in \mathcal{B}\setminus\mathcal{W}$ such that $\text{rank}_2(\mathbb{D}_{\mathcal{B}\setminus(\mathcal{W}\cup\{k\})}) = |\mathcal{L}|$ (i.e., $\mathbb{D}_{\mathcal{B}\setminus(\mathcal{W}\cup\{k\})}$ is full-rank) and $y_{k,i} \neq 0$. In the extended version of this paper [10, Appendix A], we show that if $F_{i,\mathcal{W}}$ appears in (18), the number of

multicast messages in the sum which contains $F_{i,\mathcal{W}}$ is even.[1] From this observation, it follows that each subfile in the sum (18) appears an even number of times, and thus the coefficient of this subfile in the sum is 0 over the binary field, which allows one to rewrite the sum in (18) as

$$W_{\mathcal{A}} = \underset{\mathcal{V}\in\mathcal{T}_{\mathcal{B}}:\mathcal{V}\neq\mathcal{L}}{\oplus} W_{\mathcal{B}\backslash\mathcal{V}}. \tag{19}$$

In other words, $W_{\mathcal{A}}$ can be reconstructed by the transmitted multicast messages.

As a result, each user k can recover each multicast message $W_{\mathcal{S}}$ where $\mathcal{S}\subseteq[\mathsf{K}]$ and $|\mathcal{S}|=t+1$, and thus it can decode its desired function.

Performance. In total, we transmit $\binom{\mathsf{K}}{t+1}-\binom{\mathsf{K}-\mathrm{rank}_2(\mathbb{D})}{t+1}$ multicast messages, each of which contains $\mathsf{B}/\binom{\mathsf{K}}{t}$ bits. Hence, the transmitted load is as in (7).

V. Novel Achievable Scheme for prime-power q

In the following, we generalize the proposed caching scheme in Section IV to the case where the demands are scalar linear functions on arbitrary finite field $\mathbb{F}_{\mathsf{q}}$. All the operations in the proposed scheme are on $\mathbb{F}_{\mathsf{q}}$. We start with an example.

A. Example: $(\mathsf{K},\mathsf{N},\mathsf{M},\mathsf{q})=(5,3,3/5,\mathsf{q})$

In this case, we have $t=\mathsf{KM}/\mathsf{N}=1$. Hence, in the cache placement, each file is partitioned into $\binom{\mathsf{K}}{t}=5$ equal-length subfiles. We use the file split in (9)-(10), resulting in the demand split in (12).

In the delivery phase, we assume that

$$\begin{aligned}&\text{user 1 demands } F_1;\\ &\text{user 2 demands } F_2;\\ &\text{user 3 demands } F_3;\\ &\text{user 4 demands } y_{4,1}F_1+y_{4,2}F_2+y_{4,3}F_3;\\ &\text{user 5 demands } y_{5,1}F_1+y_{5,2}F_2+y_{5,3}F_3.\end{aligned}$$

We choose the set of leaders $\mathcal{L}=[3]$, since $\mathrm{rank}_{\mathsf{q}}(\mathbb{D}_{[3]})=3$.

Each user $k\in[\mathsf{K}]$ should recover each block $B_{k,\mathcal{W}}=y_{k,1}F_{1,\mathcal{W}}+y_{k,2}F_{2,\mathcal{W}}+y_{k,3}F_{3,\mathcal{W}}$ in the delivery phase, where $\mathcal{W}\in[5]\setminus\{k\}$ and $|\mathcal{W}|=t=1$.

Encoding. For each set $\mathcal{S}\subseteq[\mathsf{K}]$ where $|\mathcal{S}|=t+1=2$, the multicast message $W_{\mathcal{S}}$ is given in (15). We alternate the coefficients (either 1 or -1) of the desired blocks of the leaders in $\mathcal{S}$, and then alternate the coefficients (either 1 or -1) of the desired blocks of the non-leaders in $\mathcal{S}$. With this, we can list all the multicast messages as

$$\begin{aligned}
W_{\{1,2\}} &= F_{1,\{2\}}-F_{2,\{1\}};\\
W_{\{1,3\}} &= F_{1,\{3\}}-F_{3,\{1\}};\\
W_{\{1,4\}} &= F_{1,\{4\}}+(y_{4,1}F_{1,\{1\}}+y_{4,2}F_{2,\{1\}}+y_{4,3}F_{3,\{1\}});\\
W_{\{1,5\}} &= F_{1,\{5\}}+(y_{5,1}F_{1,\{1\}}+y_{5,2}F_{2,\{1\}}+y_{5,3}F_{3,\{1\}});\\
W_{\{2,3\}} &= F_{2,\{3\}}-F_{3,\{2\}};\\
W_{\{2,4\}} &= F_{2,\{4\}}+(y_{4,1}F_{1,\{2\}}+y_{4,2}F_{2,\{2\}}+y_{4,3}F_{3,\{2\}});\\
W_{\{2,5\}} &= F_{2,\{5\}}+(y_{5,1}F_{1,\{2\}}+y_{5,2}F_{2,\{2\}}+y_{5,3}F_{3,\{2\}});\\
W_{\{3,4\}} &= F_{3,\{4\}}+(y_{4,1}F_{1,\{3\}}+y_{4,2}F_{2,\{3\}}+y_{4,3}F_{3,\{3\}});\\
W_{\{3,5\}} &= F_{3,\{5\}}+(y_{5,1}F_{1,\{3\}}+y_{5,2}F_{2,\{3\}}+y_{5,3}F_{3,\{3\}});\\
W_{\{4,5\}} &= (y_{4,1}F_{1,\{5\}}+y_{4,2}F_{2,\{5\}}+y_{4,3}F_{3,\{5\}})\\
&\quad-(y_{5,1}F_{1,\{4\}}+y_{5,2}F_{2,\{4\}}+y_{5,3}F_{3,\{4\}}).
\end{aligned}$$

Delivery. The server broadcasts $W_{\mathcal{S}}$ for each $\mathcal{S}\subseteq[\mathsf{K}]$ where $|\mathcal{S}|=t+1=2$ and $\mathcal{S}\cap\mathcal{L}\neq\emptyset$. In other words, the server broadcasts all the multicast messages except for $W_{\{4,5\}}$.

Decoding. We first show the untransmitted multicast message $W_{\{4,5\}}$ can be reconstructed by the transmitted multicast messages. More precisely, we aim to choose the decoding coefficients $\beta_{\{4,5\},\mathcal{S}}\in\mathbb{F}_{\mathsf{q}}$ in (16) for each $\mathcal{S}\subseteq[\mathsf{K}]$ where $|\mathcal{S}|=t+1$ and $\mathcal{S}\cap\mathcal{L}\neq\emptyset$, such that

$$W_{\{4,5\}}=\sum_{\mathcal{S}\subseteq[\mathsf{K}]:|\mathcal{S}|=t+1,\mathcal{S}\cap\mathcal{L}\neq\emptyset}\beta_{\{4,5\},\mathcal{S}}W_{\mathcal{S}}. \tag{20}$$

Since on the RHS of (20) the subfile $F_{1,\{4\}}$ only appears in $W_{\{1,4\}}$ and on the LHS of (20) the coefficient of $F_{1,\{4\}}$ is $-y_{5,1}$, in order to have the same coefficient for $F_{1,\{4\}}$ on both sides of (20), we let[2]

$$\beta_{\{4,5\},\{1,4\}}=-y_{5,1}=-\det([y_{5,1}]). \tag{21}$$

Similarly for $F_{2,\{4\}}$, we let

$$\beta_{\{4,5\},\{2,4\}}=-y_{5,2}=-\det([y_{5,2}]);$$

for $F_{3,\{4\}}$ we let

$$\beta_{\{4,5\},\{3,4\}}=-y_{5,3}=-\det([y_{5,3}]);$$

for $F_{1,\{5\}}$ we let

$$\beta_{\{4,5\},\{1,5\}}=y_{4,1}=\det([y_{4,1}]);$$

for $F_{2,\{5\}}$ we let

$$\beta_{\{4,5\},\{2,5\}}=y_{4,2}=\det([y_{4,2}]);$$

for $F_{3,\{5\}}$, we let

$$\beta_{\{4,5\},\{3,5\}}=y_{4,3}=\det([y_{4,3}]).$$

Next we focus $F_{1,\{1\}}$, which appears in $W_{\{1,4\}}$ and $W_{\{1,5\}}$. Since $\beta_{\{4,5\},\{1,4\}}=-y_{5,1}$ and $\beta_{\{4,5\},\{1,5\}}=y_{4,1}$, the coefficient of $F_{1,\{1\}}$ on the RHS of (20) is

$$y_{4,1}\beta_{\{4,5\},\{1,4\}}+y_{5,1}\beta_{\{4,5\},\{1,5\}}=0. \tag{22}$$

Similarly, the coefficient of $F_{2,\{2\}}$ on the RHS of (20), which appears in $W_{\{2,4\}}$ and $W_{\{2,5\}}$, is zero; and of $F_{3,\{3\}}$ on the RHS of (20), which appears in $W_{\{3,4\}}$ and $W_{\{3,5\}}$, is zero.

Now we focus on $F_{1,\{2\}}$, which appears in $W_{\{1,2\}}$, $W_{\{2,4\}}$, and $W_{\{2,5\}}$. Since $\beta_{\{4,5\},\{2,4\}}=-y_{5,2}$ and $\beta_{\{4,5\},\{2,5\}}=$

[1] In the YMA scheme for the MAN caching problem, each subfile in [3, eq. 6] is contained in two multicast messages in [3, eq. 6]. Hence, our proof is also a generalization of [3, Lemma 1] for the YMA scheme.

[2] Notice that (21) we write $y_{5,1}$ as the determinant of the 1×1 matrix $[y_{5,1}]$ because this will be generalized later on and that these decoding coefficients are always given by determinants of suitable matrices.

$y_{4,2}$, in order to let the coefficient of $F_{1,\{2\}}$ on the RHS of (20) be zero, we let

$$\beta_{\{4,5\},\{1,2\}} = y_{4,1}y_{5,2} - y_{5,1}y_{4,2} = \det([y_{4,1}, y_{4,2}; y_{5,1}, y_{5,2}]).$$

In addition, $F_{2,\{1\}}$ appears in $W_{\{1,2\}}$, $W_{\{1,4\}}$, and $W_{\{1,5\}}$. The coefficient of $F_{2,\{1\}}$ on the RHS of (20) is

$$-\beta_{\{4,5\},\{1,2\}} + y_{4,2}\beta_{\{4,5\},\{1,4\}} + y_{5,2}\beta_{\{4,5\},\{1,5\}} = 0. \quad (23)$$

Similarly, we let

$$\beta_{\{4,5\},\{1,3\}} = y_{4,1}y_{5,3} - y_{5,1}y_{4,3} = \det([y_{4,1}, y_{4,3}; y_{5,1}, y_{5,3}]),$$

such that the coefficients of $F_{1,\{3\}}$ and $F_{3,\{1\}}$ on the RHS of (20) are zero. We let

$$\beta_{\{4,5\},\{2,3\}} = y_{4,2}y_{5,3} - y_{5,2}y_{4,3} = \det([y_{4,2}, y_{4,3}; y_{5,2}, y_{5,3}]),$$

such that the coefficients of $F_{2,\{3\}}$ and $F_{3,\{2\}}$ on the RHS of (20) are 0.

With the above choice of decoding coefficients, we satisfies (20). In conclusion, each user can recover all multicast messages, and then recover its demanded function.

Performance. In total we transmit $\binom{\mathsf{K}}{t+1} - \binom{\mathsf{K}-\text{rank}_{\mathsf{q}}(\mathbb{D})}{t+1} = \binom{5}{2} - \binom{2}{2} = 9$ multicast messages, each of which contains $\frac{\mathsf{B}}{5}$ symbols. Hence, the transmitted load is $\frac{9}{5}$, which coincides with the optimal worst-case load for single-file retrieval under the constraint of undcoded cache placement and it is thus also optimal under the same conditions for function retrieval.

B. General Description

We use the file split in (9)-(10), resulting in the demand split in (12). In the delivery phase, after the demand matrix $\mathbb{D}$ is revealed, among the K users we first choose $\text{rank}_{\mathsf{q}}(\mathbb{D})$ leaders (assume the set of leaders is $\mathcal{L} = \{\mathcal{L}(1), \ldots, \mathcal{L}(|\mathcal{L}|)\}$), where

$$|\mathcal{L}| = \text{rank}_{\mathsf{q}}(\mathbb{D}_{\mathcal{L}}) = \text{rank}_{\mathsf{q}}(\mathbb{D}). \quad (24)$$

For each $i \in [|\mathcal{L}|]$, we also define that the *leader index* of leader $\mathcal{L}(i)$ is i. From (24), we can represent the demands of non-leaders by the linear combinations of the demands of leaders. More precisely, we define

$$F'_i := y_{\mathcal{L}(i),1}F_1 + \ldots + y_{\mathcal{L}(i),\mathsf{N}}F_{\mathsf{N}}, \ \forall i \in [|\mathcal{L}|], \quad (25)$$

and represent the demand of each user $k \in [\mathsf{K}]$ by

$$y_{k,1}F_1 + \ldots + y_{k,\mathsf{N}}F_{\mathsf{N}} = x_{k,1}F'_1 + \ldots + x_{k,|\mathcal{L}|}F'_{|\mathcal{L}|}. \quad (26)$$

Clearly, for each leader $\mathcal{L}(i)$ where $i \in [|\mathcal{L}|]$, $\mathbf{x}_{\mathcal{L}(i)}$ is an $|\mathcal{L}|$-dimension unit vector where the i^{th} element is 1. The transformed demand matrix $\mathbb{D}'$ is defined as follows,

$$\mathbb{D}' = [x_{1,1}, \ldots, x_{1,|\mathcal{L}|}; \ldots; x_{\mathsf{K},1}, \ldots, x_{\mathsf{K},|\mathcal{L}|}]. \quad (27)$$

In addition, for each $i \in [|\mathcal{L}|]$ and each $\mathcal{W} \subseteq [\mathsf{K}]$ where $|\mathcal{W}| = t$, we define

$$F'_{i,\mathcal{W}} := y_{\mathcal{L}(i),1}F_{1,\mathcal{W}} + \ldots + y_{\mathcal{L}(i),\mathsf{N}}F_{\mathsf{N},\mathcal{W}}, \quad (28)$$

refer $F'_{i,\mathcal{W}}$ to as a *transformed subfile*, and refer

$$B'_{k,\mathcal{W}} = x_{k,1}F'_{1,\mathcal{W}} + \ldots + x_{k,|\mathcal{L}|}F'_{|\mathcal{L}|,\mathcal{W}} \quad (29)$$

to as a *transformed block*.

Encoding. For $\mathcal{S} \subseteq [\mathsf{K}]$, denote the set of leaders in $\mathcal{S}$ by

$$\mathcal{L}_{\mathcal{S}} := \mathcal{S} \cap \mathcal{L}, \quad (30)$$

and the set of non-leaders in $\mathcal{S}$ by

$$\mathcal{N}_{\mathcal{S}} := \mathcal{S} \setminus \mathcal{L}. \quad (31)$$

We also denote the leader indices of leaders in $\mathcal{S}$ by

$$\text{Ind}_{\mathcal{S}} := \{i \in [|\mathcal{L}|] : \mathcal{L}(i) \in \mathcal{S}\}. \quad (32)$$

For example, if $\mathcal{L} = \{2, 4, 5\}$ and $\mathcal{S} = \{1, 2, 5\}$, we have $\mathcal{L}_{\mathcal{S}} = \{2, 5\}$, $\mathcal{N}_{\mathcal{S}} = \{1\}$, and $\text{Ind}_{\mathcal{S}} = \{1, 3\}$.

Now we focus on each set $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t + 1$, and generate the multicast message

$$W_{\mathcal{S}} = \sum_{i \in [|\mathcal{L}_{\mathcal{S}}|]} (-1)^{i-1} B'_{\mathcal{L}_{\mathcal{S}}(i), \mathcal{S}\setminus\{\mathcal{L}_{\mathcal{S}}(i)\}} + \sum_{j \in [|\mathcal{N}_{\mathcal{S}}|]} (-1)^{j-1} B'_{\mathcal{N}_{\mathcal{S}}(j), \mathcal{S}\setminus\{\mathcal{N}_{\mathcal{S}}(j)\}}. \quad (33)$$

Delivery. The server broadcasts $W_{\mathcal{S}}$ for each $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t + 1$ and $\mathcal{S} \cap \mathcal{L} \neq \emptyset$.

Decoding. We consider each set $\mathcal{A} \subseteq [\mathsf{K}]$ where $|\mathcal{A}| = t+1$ and $\mathcal{A} \cap \mathcal{L} = \emptyset$. We define that the *non-leader index* of non-leader $\mathcal{A}(i)$ is i, where $i \in [t + 1]$. For each $\mathcal{S} \subseteq \mathcal{A} \cup \mathcal{L}$, we have $\mathcal{N}_{\mathcal{S}} \subseteq \mathcal{A}$. In addition, with a slight abuse of notation we denote the non-leader indices of non-leaders in $\mathcal{A} \setminus \mathcal{S}$ by

$$\overline{\text{Ind}}_{\mathcal{S}} = \{i \in [t + 1] : \mathcal{A}(i) \notin \mathcal{S}\}. \quad (34)$$

For example, if $\mathcal{A} = \{4, 5, 6\}$ and $\mathcal{S} = \{1, 2, 5\}$, we have $\overline{\text{Ind}}_{\mathcal{S}} = \{1, 3\}$. For any set $\mathcal{X}$ and any number y, we define $\text{Tot}(\mathcal{X})$ as the sum of the elements in $\mathcal{X}$, i.e.,

$$\text{Tot}(\mathcal{X}) := \sum_{i \in [|\mathcal{X}|]} \mathcal{X}(i);. \quad (35)$$

It is proved in [10, Appendix A] that (16) holds with

$$\beta_{\mathcal{A},\mathcal{S}} = (-1)^{1+\text{Tot}(\overline{\text{Ind}}_{\mathcal{S}})} \det(\mathbb{D}'_{\mathcal{A}\setminus\mathcal{S},\text{Ind}_{\mathcal{S}}}). \quad (36)$$

In other words, each user $k \in [\mathsf{K}]$ can recover all messages $W_{\mathcal{S}}$ where $\mathcal{S} \subseteq [\mathsf{K}]$ and $|\mathcal{S}| = t + 1$. For each desired transformed block $B'_{k,\mathcal{W}}$, where $\mathcal{W} \subseteq ([\mathsf{K}] \setminus \{k\})$ and $|\mathcal{W}| = t$, user k can recover it in $W_{\mathcal{W}\cup\{k\}}$, because it knows all the other transformed blocks in $W_{\mathcal{W}\cup\{k\}}$. Hence, user k can recover $x_{k,1}F'_1 + \ldots + x_{k,|\mathcal{L}|}F'_{|\mathcal{L}|}$, which is identical to its demand.

Performance. We transmit $\binom{\mathsf{K}}{t+1} - \binom{\mathsf{K}-\text{rank}_{\mathsf{q}}(\mathbb{D})}{t+1}$ multicast messages, each of $\mathsf{B}/\binom{\mathsf{K}}{t}$ symbols, thus giving (7).

VI. Conclusion

This paper shows that there is no load penalty in retrieving linear functions of the library files in cache-aided shared-link systems with uncoded cache placement.

Acknowledgement: The work of K. Wan and G. Caire was partially funded by the European Research Council under the ERC Advanced Grant N. 789190, CARENET. The work of M. Ji was supported in part by NSF Awards 1817154 and 1824558. The work of D. Tuninetti was supported in part by NSF Award 1910309.

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," *IEEE Trans. Infor. Theory*, vol. 64, pp. 1281 – 1296, Feb. 2018.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Trans. Infor. Theory*, vol. 65, no. 1, pp. 647–663, Jan. 2019.

[5] 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.

[6] C. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of d2d coded caching with uncoded cache placement and one-shot delivery," *in IEEE Int. Symp. Inf. Theory*, Jul. 2019.

[7] K. Wan and G. Caire, "On coded caching with private demands," *arXiv:1908.10821*, Aug. 2019.

[8] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "Fundamental limits of device-to-device private caching with trusted server," *arXiv:1912.09985*, Dec. 2019.

[9] H. Sun and S. A. Jafar, "The capacity of private computation," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 3880–3897, Jun. 2019.

[10] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "On optimal load-memory tradeoff of cache-aided scalar linear function retrieval," *arXiv:2001.03577*, Jan. 2020.