Cache-Aided Matrix Multiplication Retrieval

Kai Wan, *Member, IEEE*, Hua Sun, *Member, IEEE*, Mingyue Ji, *Member, IEEE*,
Daniela Tuninetti, *Fellow, IEEE*, and Giuseppe Caire, *Fellow, IEEE*

***Abstract*—Coded caching is a promising technique to smooth out network traffic by storing part of the library content at the users' local caches. The seminal work on coded caching for single file retrieval by Maddah-Ali and Niesen (MAN) showed the existence of a global caching gain that scales with the total memory in the system, in addition to the known local caching gain in uncoded systems. This paper formulates a novel cache-aided matrix multiplication retrieval problem, relevant for data analytics and machine learning applications. In the considered problem, each cache-aided user requests the product of two matrices from the library. A structure-agnostic solution is to treat each possible matrix product as an independent file and use the MAN coded caching scheme for single file retrieval. This paper proposes two structure-aware schemes, which partition each matrix in the library by either rows or columns and let a subset of users cache some sub-matrices, that improve on the structure-agnostic scheme. For the case where the library matrices are "fat" matrices, the structure-aware row-partition scheme is shown to be order optimal under some constraint.**

***Index Terms*—Coded caching, matrix multiplication retrieval.**

I. Introduction

IT IS predicted that an order of magnitude increase in network throughput is needed to support the tremendous growth of data traffic expected for the near future [1]. Conventional technologies are severely limited towards the goal of achieving such a dramatic throughput gain. A clever usage of low-cost storage capacity on user devices to cache data plays a key role in the design of content distribution schemes. Coded caching is an effective way to smooth out network traffic during peak traffic hours by jointly designing cache placement and coded delivery schemes. The coded caching strategy originally proposed by Maddah-Ali and Niesen (MAN) in [2] has the potential to trade off relatively cheap memory for expensive bandwidth, i.e., the total traffic load on the network is inversely proportional to the aggregate cache memory in the network, a phenomenon referred to as *global coded caching gain.*

The MAN original model consists of a server, with access to the whole library, that is connected to several cache-aided users through an error-free shared-link. The MAN scheme contains two phases: (i) *placement phase* (peak-off hours): each cache-aided user stores some bits in its local cache without knowledge of later demands; (ii) *delivery phase* (peak-traffic hours): each user requests one file from the library and the server broadcasts coded packets to satisfy all users' requests simultaneously. The goal is to minimize the number of broadcasted bits for the worst-case demands, referred to as *worst-case load*. It was surprisingly shown in [2] that if each bit in the library can be cached by t users, the total load can potentially be reduced by $t+1$ times compared to the conventional uncoded caching scheme, in which the server simply broadcasts to each user the uncached part of the demanded file. The MAN shared-link coded caching problem for single file retrieval has been extended to a number of different network models (such as Device-to-Device networks [3], topological networks [4], multi-server networks [5], wireless interference channels [6], etc.) and different problems where reducing the communication cost is paramount (such as coded distributed computing [7], coded data shuffling [8]–[11], etc.).

A common point of the above problems is that users request whole files. Motivated by the fact that linear and multivariate polynomial operations are widely used fundamental primitives for building the complex queries that support many engineering problems, coded caching was introduced into the scalar linear function retrieval in [12]. Instead of letting each user download all the input files in the desired scalar linear function of files, an optimal coded caching scheme with uncoded cache placement was proposed in [12], which lets each user directly recover the desired function. In this paper, we turn our attention from scalar linear function to matrix multiplication. Matrix multiplication plays a key role in a wide variety of domains, such as for example data analytics, machine learning, and scientific computing [8], [13], [14]. Recently, information theoretic coding techniques have been proposed for the distributed matrix multiplication problem [13]–[19].

Manuscript received July 1, 2020; revised December 1, 2021; accepted February 28, 2022. Date of publication March 8, 2022; date of current version June 15, 2022. The work of Kai Wan and Giuseppe Caire was supported in part by the European Research Council under the ERC Advanced Grant N. 789190, CARENET. The work of Hua Sun was supported in part by NSF under Grant CCF-2007108 and Grant CCF-2045656. The work of Mingyue Ji was supported in part by NSF under Award 1817154 and Award 1824558. The work of Daniela Tuninetti was supported in part by NSF under Award 1910309. An earlier version of this paper was presented at the 2021 IEEE International Symposium on Information Theory, Melbourne, VIC, Australia [DOI: 10.1109/ISIT45174.2021.9518042]. *(Corresponding author: Kai Wan.)*

Kai Wan and Giuseppe Caire are with the Electrical Engineering and Computer Science Department, Technische Universität Berlin, 10587 Berlin, Germany (e-mail: kai.wan@tu-berlin.de; caire@tu-berlin.de).

Hua Sun is with the Department of Electrical Engineering, University of North Texas, Denton, TX 76203 USA (e-mail: hua.sun@unt.edu).

Mingyue Ji is with the Electrical and Computer Engineering Department, The University of Utah, Salt Lake City, UT 84112 USA (e-mail: mingyue.ji@utah.edu).

Daniela Tuninetti is with the Electrical and Computer Engineering Department, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: danielat@uic.edu).

Communicated by M. A. Maddah-Ali, Associate Editor for Communications.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3157835.

Digital Object Identifier 10.1109/TIT.2022.3157835

In a distributed computing system a master node aims to compute the multiplication of two large-scale matrices with the help of workers, where the workers can only store and compute on small parts of the matrices. Since workers may take different amounts of time to complete their assigned task, i.e., some are stragglers, the goal here is for the master node to recover the matrix product as soon as the number of responses received from the workers reaches the so-called recovery threshold. Different coding schemes have been proposed to mitigate the impact of stragglers on the completion time of a distributed computing task, such as polynomial codes [13], [20] and Matdot codes [14]. Recently, distributed matrix multiplication for resilience against stragglers was extended to wireless channels [21], where several users without local cache are connected to edge nodes with computation resources through a wireless link and where each user requests the product of a user-generated data matrix with a network-stored matrix. In this work, we are not interested in the problem of straggler mitigation, but rather in the problem of reducing the communication load across a shared-link network.

This paper formulates a novel shared-link cache-aided matrix multiplication retrieval problem, where we consider that each cache-aided user requests the product of two matrices in the library, instead of a single file. For example, each user aims to compute the linear correlation between each two vectors of two vector sets,[1] which can be seen as the multiplication of two matrices representing these two vector sets.

In our setting, the library contains N files that are thought of as matrices of dimension $\mathsf{s} \times \mathsf{r}$ on some finite field. In the placement phase, each of the K users can store up to Msr symbols from the library (corresponding to the size of up to M matrices). During the delivery phase, each user requests the product of two arbitrary matrices in the library, which are not known in advance at the time of cache placement. Different from existing information theoretic distributed matrix multiplication works for straggler mitigation, **we aim to apply coded caching strategies to the matrix multiplication retrieval problem with the goal of minimizing the load on the shared link between the server and the users by leveraging the cached contents and performing coded multicast delivery.**

A. Main Contributions

Our main contributions are as follows.

- We formulate an information theoretic shared-link coded caching problem for matrix multiplication retrieval, where each user requests the product of two matrices in the library.
- We propose a structure-agnostic scheme that treats each possible demanded matrix product as an independent file and attains the load corresponding to the MAN coded caching problem for single file retrieval.
- Then, we propose two coded caching schemes that leverage the specific structure of matrix multiplication. Different from the structure-agnostic matrix multiplication retrieval scheme, which lets the users directly cache some entries of the matrix products, the proposed structure-aware schemes let each user cache some entries of each matrix. One scheme partitions each library matrix into sub-matrices by rows and the other by columns. A subset of the users cache each sub-matrix, or some linear transformation of this sub-matrix. The delivery phase is designed so as to leverage the users' cached contents and the "correlation" among the elements of the demanded matrix products, i.e., the fact that some entries of a matrix product can be written as a function of the other entries of the same matrix product.
- When $\mathsf{s} \leq \mathsf{r}$ (i.e., the library matrices are "fat" matrices), we prove that the proposed row-partition scheme is order optimal within a factor of 2 under the constraint of uncoded cache placement (i.e., each user directly copies some entries of the matrices in the library into its local cache) and $\mathsf{N} \geq 2\mathsf{K}$. This is accomplished by proposing a novel genie-aided converse bound.

B. Paper Organization

The rest of this paper is organized as follows. Section II gives some results used later in the paper. Section III formulates the cache-aided matrix multiplication retrieval problem. Section IV summarizes the main results in this paper. Section V provides the details of the proposed coded cache-aided matrix multiplication retrieval schemes. Section VI concludes the paper. Some proofs can be found in the Appendix.

C. Notation Convention

Calligraphic symbols denote sets, bold symbols denote vectors and matrices, 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 $[n] := [1:n]$; $\oplus$ represents bit-wise XOR; $a! = a \times (a-1) \times \ldots \times 1$ represents the factorial of a; $\mathbb{F}_{\mathsf{q}}$ represents a finite field with order q; $\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{-1}$ represent the transpose and the inverse of matrix $\mathbf{A}$, respectively; $\mathrm{rank}(\mathbf{A})$ represents the rank of matrix $\mathbf{A}$; $\mathbf{I}_n$ represents the identity matrix of dimension $n \times n$; $(\mathbf{A})_{m \times n}$ explicitly indicates that the matrix $\mathbf{A}$ is of dimension $m \times n$; the matrix $[a; b]$ is written in a Matlab form, representing $\begin{bmatrix} a \\ b \end{bmatrix}$; we let $\binom{x}{y} = 0$ if $x < 0$ or $y < 0$ or $x < y$. In the rest of the paper entropies will be in base q, where q will be introduced later.

II. Preliminary Results on the Entropy of a Matrix Product

In this section we describe a procedure to "compress" matrix products that may not be full rank so as to reduce the load on the shared-link.

Consider a matrix $\mathbf{A} \in \mathbb{F}_{\mathsf{q}}^{M \times m}$ on a finite field $\mathbb{F}_{\mathsf{q}}$ of rank ρ with $M \geq m \geq \rho > 0$. We can choose ρ linearly independent rows of $\mathbf{A}$ and call the resulting matrix $\mathbf{A}_1 \in \mathbb{F}_{\mathsf{q}}^{\rho \times m}$, that is, $\mathbf{A}_1 \mathbf{A}_1^{\mathrm{T}} \in \mathbb{F}_{\mathsf{q}}^{\rho \times \rho}$ is full rank. We then can express each of the

[1] Linear correlation is used to find the linear relationship between two numerically expressed variables, which has wide applications in lots of areas, such as engineering research (including pattern recognition [22], signal detection [23], etc.) and medical science [24].

remaining $M-\rho$ rows of $\mathbf{A}$ as a linear combination of the rows of $\mathbf{A}_1$; let the matrix of the coefficients for the linear combinations be $\mathbf{A}_2 \in \mathbb{F}_{\mathsf{q}}^{(M-\rho)\times\rho}$. Finally, the original matrix $\mathbf{A}$ can be written as $\mathbf{A} = \mathbf{A}_3 \begin{bmatrix} \mathbf{I}_\rho \\ \mathbf{A}_2 \end{bmatrix} \mathbf{A}_1$, for some permutation matrix $\mathbf{A}_3 \in \{0,1\}^{M\times M}$ that only depends on the set of indices of the ρ chosen rows out of M rows. Thus we can write

$$\begin{aligned} H(\mathbf{A}) &= H(\mathbf{A}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3) && \text{(1a)}\\ &= H(\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_2) + H(\mathbf{A}|\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3) && \text{(1b)}\\ &= H(\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3) && \text{(1c)}\\ &\le H(\mathbf{A}_1) + H(\mathbf{A}_2) + H(\mathbf{A}_3) && \text{(1d)}\\ &\le \rho m + (M-\rho)\rho + \log_{\mathsf{q}}\left(\binom{M}{\rho}\right). && \text{(1e)} \end{aligned}$$

In other words, we need at most $(M+m)\rho - \rho^2$ symbols on $\mathbb{F}_{\mathsf{q}}$ to specify any $\mathbf{A} \in \mathbb{F}_{\mathsf{q}}^{M\times m}$ of rank ρ, up to a permutation matrix that contributes $\log_{\mathsf{q}}\left(\binom{M}{\rho}\right)$ to the entropy.

Next, for any two matrices $\mathbf{C} \in \mathbb{F}_{\mathsf{q}}^{m\times n}$ and $\mathbf{B} \in \mathbb{F}_{\mathsf{q}}^{n\times p}$, the entropy bound in (1), together with

$$\text{rank}[\mathbf{CB}] \le \min(\text{rank}[\mathbf{C}], \text{rank}[\mathbf{B}]) \le \min(n, m, p), \quad (2)$$

implies that we need, up to some symbols needed to describe a permutation, at most $f(m,n,p) = f(p,n,m)$ symbols on $\mathbb{F}_{\mathsf{q}}$ to specify the matrix product $\mathbf{CB} \in \mathbb{F}_{\mathsf{q}}^{m\times p}$ where the function $f(m,n,p)$ is defined as

$$\begin{aligned} f(m,n,p) &:= (m+p-\min(n,m,p))\min(n,m,p) && \text{(3a)}\\ &= \begin{cases} (m+p-n)n & \min(m,p) \ge n \\ mp & \min(m,p) \le n \end{cases}. && \text{(3b)} \end{aligned}$$

For later use, we express $f(m,n,p) = g\left(\frac{m}{n}, \frac{p}{n}\right)n^2$, where $g(\alpha,\beta)$ is a symmetric function in its arguments as is defined as

$$g(\alpha,\beta) := \begin{cases} \alpha+\beta-1 & \min(\alpha,\beta) \ge 1 \\ \alpha\beta & \min(\alpha,\beta) \le 1 \end{cases}. \quad (4)$$

Note that $\frac{g(\alpha,\alpha)}{\alpha} \le 2$.

In the rest of the paper, we will use $P(\mathbf{C},\mathbf{B})$ to denote the $f(m,n,p) + H(\mathbf{A}_3)$ symbols on $\mathbb{F}_{\mathsf{q}}$ that specify the matrix product $\mathbf{CB}$, where we set $\mathbf{A} = \mathbf{CB} \in \mathbb{F}_{\mathsf{q}}^{m\times p}$ in (1). Next, we will consider the following two cases:

- *n is large.* For each product $\mathbf{CB}$ considered in formulated cache-aided matrix multiplication problem (which will be clarified later), we assume that $m = a_1 n$ and $p = a_2 n$, where a_1, a_2 are fixed positive numbers and $n \gg \max\{a_1, a_2\}$. In this case of large matrices, for any field size q,

$$\begin{aligned} &\log_{\mathsf{q}}\left(\binom{\max(m,p)}{\min(n,m,p)}\right) \le \log_{\mathsf{q}}(\max(m,p)!) \\ &\le \underbrace{\frac{3}{2}\log_{\mathsf{q}}(e) + (\max(m,p)+\frac{1}{2})\log_{\mathsf{q}}(\frac{\max(m,p)}{e})}_{\text{by Stirling's approximation}}. \end{aligned}$$

Hence, we have (recall that $f(m,n,p)$ is with order $O(n^2)$)

$$\begin{aligned} &\frac{\log_{\mathsf{q}}\left(\binom{\max(m,p)}{\min(n,m,p)}\right)}{f(m,n,p)} \\ &\le \frac{\frac{3}{2}\log_{\mathsf{q}}(e) + (\max(a_1,a_2)n + \frac{1}{2})\log_{\mathsf{q}}(\frac{\max(a_1,a_2)n}{e})}{f(a_1\ n, n, a_2 n)} \\ &= \varepsilon_n, \end{aligned}$$

where $\lim_{n\to\infty}\varepsilon_n = 0$. So in this case of large enough matrices, we have $|P(\mathbf{C},\mathbf{B})| \le (1+\varepsilon_n)f(m,n,p)$.

- q *is large.* [25, Lemma 2] proved that for any two independent matrices $\mathbf{C} \in \mathbb{F}_{\mathsf{q}}^{m\times n}$ and $\mathbf{B} \in \mathbb{F}_{\mathsf{q}}^{n\times p}$ with uniformly i.i.d. entries on $\mathbb{F}_{\mathsf{q}}$, we have

$$\lim_{\mathsf{q}\to\infty} H(\mathbf{CB}) = f(m,n,p). \quad (5)$$

That is because, in this case we have

1) the matrices corresponding to $\mathbf{A}_1, \mathbf{A}_2$ in (1a) for the matrix $\mathbf{A} = \mathbf{CB} \in \mathbb{F}_{\mathsf{q}}^{m\times p}$ have uniformly i.i.d. entries, which leads $H(\mathbf{A}_1, \mathbf{A}_2) = H(\mathbf{A}_1) + H(\mathbf{A}_2) = f(m,n,p)$;
2) $H(\mathbf{A}_3) \le \log_{\mathsf{q}}\left(\binom{\max(m,p)}{\min(n,m,p)}\right) = \varepsilon_{\mathsf{q}}$, where $\lim_{\mathsf{q}\to\infty}\varepsilon_{\mathsf{q}} = 0$.

Thus we have $f(m,n,p) < H(\mathbf{CB}) \le |P(\mathbf{C},\mathbf{B})| = f(m,n,p) + \varepsilon_{\mathsf{q}}$, which leads to (5). Hence, we also have $\lim_{\mathsf{q}\to\infty}|P(\mathbf{C},\mathbf{B})| = f(m,n,p)$.

III. System Model

The $(\mathsf{K},\mathsf{N},\mathsf{a})$ shared-link cache-aided matrix multiplication retrieval problem is defined as follows. A server has access to a library of N matrices, denoted by $\mathbf{W}_1, \dots, \mathbf{W}_{\mathsf{N}}$, and each matrix is of dimension $\mathsf{s}\times\mathsf{r}$ on a finite field $\mathbb{F}_{\mathsf{q}}$, for some prime-power q. The column-row ratio of each matrix is denoted by $\mathsf{a} := \mathsf{r}/\mathsf{s} \in (0,\infty)$. We further assume that each element of each matrix is uniformly i.i.d. over $\mathbb{F}_{\mathsf{q}}$ and that q is sufficiently large so that the entropy of any matrix product $\mathbf{W}_i^T\mathbf{W}_j$ where $(i,j) \in [\mathsf{N}]^2$ is

$$\mathsf{B} := f(\mathsf{r},\mathsf{s},\mathsf{r}) = \mathsf{s}^2 g(\mathsf{a},\mathsf{a}) \le 2\mathsf{rs}, \quad (6)$$

i.e., B is the number of symbols on $\mathbb{F}_{\mathsf{q}}$ that suffices to specify any matrix product, as argued in Section II.[2] We also assume that s is finite and sufficiently large, such that any sub-matrix division is possible. The server is connected to K users through an error-free shared link. The system operates as follows.

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

$$\phi_k : \mathbb{F}_{\mathsf{q}}^{\mathsf{Nsr}} \to \mathbb{F}_{\mathsf{q}}^{\lfloor\mathsf{Msr}\rfloor}. \quad (7)$$

[2] Note that without the assumption that $\mathsf{q}\to\infty$, each proposed achievable scheme can still work to let each user retrieve its demanded matrix product. As showed in Section II, $\mathsf{q}\to\infty$ is needed for the converse of (5), which characterizes the entropy of matrix product. In addition, this assumption is also needed for the proposed converse bounds on the minimum worst-case load.

We denote the content in the cache of user $k \in [\mathsf{K}]$ by $Z_k = \phi_k(\mathbf{W}_1, \dots, \mathbf{W}_\mathsf{N})$. The non-negative parameter M is the *cache size*, measured in multiple of the size of each matrix in the library.

b) Delivery Phase: During the delivery phase, user $k \in [\mathsf{K}]$ sends its demand $\mathbf{d}_k = (d_{k,1}, d_{k,2})$ to the server, where $(d_{k,1}, d_{k,2}) \in [\mathsf{N}]^2$ means that user k requests the matrix product $\mathbf{W}^\mathsf{T}_{d_{k,1}} \mathbf{W}_{d_{k,2}} \in \mathbb{F}_\mathsf{q}^{\mathsf{r}\times\mathsf{r}}$. Given the demand $[\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_\mathsf{K}] \in [\mathsf{N}]^{2\times\mathsf{K}}$, the server broadcasts the message $X = \psi([\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_\mathsf{K}], \mathbf{W}_1, \dots, \mathbf{W}_\mathsf{N})$ to the users, where the encoding function ψ is such that

$$\psi : [\mathsf{N}]^{2\mathsf{K}} \times \mathbb{F}_\mathsf{q}^{\mathsf{Nsr}} \to \mathbb{F}_\mathsf{q}^{\lfloor \mathsf{RB} \rfloor}. \tag{8}$$

The non-negative parameter R is referred to as the *load* on the shared link, measured in multiple of the entropy of a matrix product B defined in (6).

c) Correctness: Each user $k \in [\mathsf{K}]$ decodes its desired matrix product from $([\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_\mathsf{K}], Z_k, X)$ through the decoding function ξ_k, defined as

$$\xi_k : [\mathsf{N}]^{2\mathsf{K}} \times \mathbb{F}_\mathsf{q}^{\lfloor \mathsf{Msr} \rfloor} \times \mathbb{F}_\mathsf{q}^{\lfloor \mathsf{RB} \rfloor} \to \mathbb{F}_\mathsf{q}^{\mathsf{B}}. \tag{9}$$

The worst-case probability of error is defined as

$$\varepsilon := \max_{[\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_\mathsf{K}]} \Pr\{\xi_k([\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_\mathsf{K}], Z_k, X) \neq \mathbf{W}^\mathsf{T}_{d_{k,1}} \mathbf{W}_{d_{k,2}}, \text{ for some } k \in [\mathsf{K}]\}. \tag{10}$$

d) Objective: In this paper, we assume that the computation power of the server and users is unlimited. Therefore, our focus is on the optimal tradeoff between communication cost and cache storage capacity. More precisely, a communication cost (a.k.a. load) R is achievable if there exists a caching scheme with placement, encoding, and decoding functions such that $\lim_{\mathsf{q}\to\infty} \varepsilon = 0$. We aim to determine the *minimum worst-case load* among all possible demands, defined for $\mathsf{M} \in [0, \mathsf{N}]$ as

$$\mathsf{R}^\star := \inf_{\substack{(\phi_k, k\in[\mathsf{K}]),\\ \psi,\ (\xi_k, k\in[\mathsf{K}])}} \{\mathsf{R} : \mathsf{R} \text{ is achievable}\}. \tag{11}$$

e) Uncoded Cache Placement: If each user directly copies some symbols of the N matrices 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 $\mathsf{R}^\star_\mathsf{u}$.

f) Isomorphic Demands: Since $W_i^\mathsf{T} W_j = \left(W_j^\mathsf{T} W_i\right)^\mathsf{T}$ for any $(i,j) \in [\mathsf{N}]^2$, we say that the demands $W_i^\mathsf{T} W_j$ and $W_j^\mathsf{T} W_i$ are *isomorphic*. The number of non-isomorphic demands is $\binom{\mathsf{N}}{2} + \mathsf{N} = \frac{\mathsf{N}(\mathsf{N}+1)}{2} = \binom{\mathsf{N}+1}{2}$. In this paper, without loss of generality, we thus can assume that $d_{k,1} \le d_{k,2}$ for each $k \in [\mathsf{K}]$.

Remark 1 (Range of M*):* Note that when $\mathsf{M} \ge \min\left(\mathsf{N}, \frac{\mathsf{N}(\mathsf{N}+1)}{2} \frac{g(\mathsf{a},\mathsf{a})}{\mathsf{a}}\right)$, we have $\mathsf{R}^\star = 0$. Indeed, the server does not need to send anything if each user can either store all possible matrices in the library (requiring Nrs symbols) or all possible non-isomorphic matrix products (requiring $\frac{\mathsf{N}(\mathsf{N}+1)}{2}\mathsf{B}$ symbols). Recall that $\frac{\mathsf{B}}{\mathsf{rs}} = \frac{g(\mathsf{a},\mathsf{a})}{\mathsf{a}} = \min(\mathsf{a}, 2 - 1/\mathsf{a})$. Hence, only for $\mathsf{M} < \min\left(\mathsf{N}, \frac{\mathsf{N}(\mathsf{N}+1)}{2} \frac{g(\mathsf{a},\mathsf{a})}{\mathsf{a}}\right)$ the load may be non-zero, in which case we have $\mathsf{R}^\star \le \min\left(\mathsf{K}, \frac{\mathsf{N}(\mathsf{N}+1)}{2}, \mathsf{N}\frac{\mathsf{a}}{g(\mathsf{a},\mathsf{a})}\right)$, as the server can satisfy all requests by either sending all demanded non-isomorphic matrix products (requiring $\min(\mathsf{K}, \frac{\mathsf{N}(\mathsf{N}+1)}{2})\mathsf{B}$ symbols), or all matrices in the library (requiring Nrs symbols). □

IV. Main Results and Discussions

This Section is organized as follows. We first summarize our main results in Section IV-A. We then provide two examples to illustrate the main ingredients of our novel achievable schemes in Section IV-B. We provide some numerical evaluations in Section IV-C. Finally, we discuss the difference between the proposed cache-aided matrix multiplication retrieval schemes and the existing works on distributed matrix multiplication for straggler mitigation in Section IV-D.

A. Main Results

For the $(\mathsf{K}, \mathsf{N}, \mathsf{a})$ shared-link cache-aided matrix multiplication retrieval problem, a simple solution is to treat each non-isomorphic product as an independent file, and thus the considered problem becomes a coded caching problem for single file retrieval with K users and $\frac{\mathsf{N}(\mathsf{N}+1)}{2}$ files, for which we can directly use the MAN coded caching scheme for single file retrieval. Such a scheme is agnostic of the structure of matrix multiplication, and thus we refer to it as *structure-agnostic scheme*. The achieved load by the structure-agnostic scheme is given as follows. The proof can be found in Appendix A.

Theorem 1 (Structure-Agnostic Scheme): For the $(\mathsf{K}, \mathsf{N}, \mathsf{a})$ shared-link cache-aided matrix multiplication retrieval problem, $\mathsf{R}^\star \le \mathsf{R}_\text{sa}$, where R_sa is the lower convex envelope of the following memory-load pairs

$$(\mathsf{M}, \mathsf{R}_\text{sa}) = \left(\frac{\mathsf{N}(\mathsf{N}+1)}{2} \frac{g(\mathsf{a},\mathsf{a})}{\mathsf{a}} \frac{t}{\mathsf{K}}, \frac{\mathsf{K}-t}{t+1} \right),\ t \in [0:\mathsf{K}]. \tag{12}$$

Note that when $\mathsf{M} = \frac{\mathsf{N}(\mathsf{N}+1)}{2}\frac{g(\mathsf{a},\mathsf{a})}{\mathsf{a}}$, i.e., $t = \mathsf{K}$, we have $\mathsf{R}_\text{sa} = 0$—see also Remark 1.

The structure-agnostic scheme does not perform well when N is large, because the number of non-isomorphic matrix products increases quadratically with N. We can improve on Theorem 1 by designing structure-aware caching schemes, which leverage the specific structure of matrix multiplication. In the structure-agnostic scheme, each user directly caches the elements in the matrix products; in the proposed structure-aware caching schemes, each user caches $\frac{\mathsf{M}}{\mathsf{N}}\mathsf{sr}$ symbols of each matrix in the library.

We first introduce two baseline structure-aware schemes. In the first baseline scheme, referred to as *uncoded caching baseline scheme*, each user caches $\frac{\mathsf{M}}{\mathsf{N}}\mathsf{r}$ columns of each matrix in the library; thus each user can reconstruct $\left(\frac{\mathsf{M}}{\mathsf{N}}\mathsf{r}\right)^2$ elements of each matrix product from its cached content. In the second baseline scheme, referred to as *mutli-request baseline scheme*, each user directly recovers the two library matrices instead of their product, akin to a coded caching scheme for multiple files retrieval [26]. The achieved loads by the baseline structure-aware schemes are given as follows. The proof details can be found in Sections V-A and V-B, respectively.

Theorem 2 (Baseline Structure-Aware Schemes): For the (K, N, a) shared-link cache-aided matrix multiplication retrieval problem, $R^\star \leq \min(R_1, R_2)$ where R_1 is defines as

$$R_1 := K\left(1 - \frac{M^2}{N^2}\right)\frac{a^2}{g(a,a)}, \tag{13}$$

and R_2 is the lower convex envelope of the following memory-load pairs

$$(M, R_2) = \left(N\frac{t}{K}, 2\frac{K-t}{t+1}\ \frac{a}{g(a,a)}\right),\ \forall t \in [0:K]. \tag{14}$$

The main limitation of the first baseline scheme in (13) is the use of uncoded caching (i.e., there is no multicasting gain). The main limitation of the second baseline scheme in (14) is that it directly recovers the two library matrices in order to recover their product, which is not necessary. In order to improve on the baseline structure-aware schemes, we next propose two schemes where we partition the matrices in the library into sub-matrices and then let a subset of the users cache (a linear transformation of) each sub-matrix. The achieved load of the row-partition scheme is given as follows. The proof details can be found in Section V-C.

Theorem 3 (Row-Partition Scheme): For the (K, N, a) shared-link cache-aided matrix multiplication retrieval problem, $R^\star \leq R_{\text{row}}$, where

$$\begin{aligned} R_{\text{row}} := \min_{\ell \in [K]} \frac{\lceil \frac{K}{\ell} \rceil}{g(a,a)} \Bigg(& g\left(\frac{a\binom{\ell}{t_\ell}}{\alpha_\ell}, \frac{a\binom{\ell}{t_\ell}}{\alpha_\ell}\right)\frac{\alpha_\ell^2}{\binom{\ell}{t_\ell}^2}\binom{\ell}{t_\ell+1} \\ & + g\left(\frac{a\binom{\ell}{t_\ell+1}}{1-\alpha_\ell}, \frac{a\binom{\ell}{t_\ell+1}}{1-\alpha_\ell}\right)\frac{(1-\alpha_\ell)^2}{\binom{\ell}{t_\ell+1}^2}\binom{\ell}{t_\ell+2}\Bigg), \end{aligned} \tag{15a}$$

$$\alpha_\ell := t_\ell + 1 - \frac{\ell M}{N},\ \ell \in [K], \tag{15b}$$

$$t_\ell := \left\lfloor \frac{\ell M}{N} \right\rfloor,\ \ell \in [K], \tag{15c}$$

with the convention that

$$R_{\text{row}} = \min_{\ell \in [K]} \left\lceil \frac{K}{\ell} \right\rceil \frac{g\left(a\binom{\ell}{t_\ell}, a\binom{\ell}{t_\ell}\right)\binom{\ell}{t_\ell+1}}{g(a,a)\ \binom{\ell}{t_\ell}^2} \text{ when } \alpha_\ell = 1, \text{ and} \tag{15d}$$

$$R_{\text{row}} = \min_{\ell \in [K]} \left\lceil \frac{K}{\ell} \right\rceil \frac{g\left(a\binom{\ell}{t_\ell+1}, a\binom{\ell}{t_\ell+1}\right)\binom{\ell}{t_\ell+2}}{g(a,a)\ \binom{\ell}{t_\ell+1}^2} \text{ when } \alpha_\ell = 0. \tag{15e}$$

In Remark 7 we shall argue that the row-partition strategy for Theorem 3 can be used with any known (for the shared-link caching problem for single file retrieval) caching scheme with uncoded cache placement.

The achieved load of the column-partition scheme is given as follows. The proof details can be found in Section V-D.

Theorem 4 (Column-Partition Scheme): For the (K, N, a) shared-link cache-aided matrix multiplication retrieval problem, $R^\star \leq R_{\text{col}}$, where

$$R_{\text{col}} := \begin{cases} y, & \text{if } a \leq 1; \\ \frac{y + 2(a-1)\left(\alpha_K \frac{K - t_K}{t_K+1} + (1-\alpha_K)\frac{K-t_K-1}{t_K+2}\right)}{2a-1} & \text{if } a > 1; \end{cases} \tag{16a}$$

$$\begin{aligned} y := \sum_{i \in [0:t_K+1]} \binom{K}{i+1} \Bigg(& \frac{\alpha_K^2}{\binom{K}{t_K}^2}\binom{K-i}{t_K-i}\binom{K-t_K}{t_K-i} \\ & + \frac{(1-\alpha_K)^2}{\binom{K}{t_K+1}^2}\binom{K-i}{t_K+1-i}\binom{K-t_K-1}{t_K+1-i} \\ & + 2\frac{\alpha_K(1-\alpha_K)}{\binom{K}{t_K}\binom{K}{t_K+1}}\binom{K-i}{t_K-i}\binom{K-t_K}{t_K+1-i}\Bigg), \end{aligned} \tag{16b}$$

where $t_K := \lfloor \frac{KM}{N} \rfloor \in [0:K]$ and $\alpha_K = \lfloor \frac{KM}{N} \rfloor + 1 - \frac{KM}{N} \in [0,1]$ were defined in (15c) and (15b), respectively.

In Remark 6 and Remark 9 we will show that the proposed row- and column-partition schemes outperform the two baseline schemes, respectively, and therefore we have the following Corollary.

Corollary 1: For the (K, N, a) shared-link cache-aided matrix multiplication retrieval problem, we have $R_{\text{row}} \leq R_2$ and $R_{\text{col}} \leq R_1$, for all $M \in [0, N]$.

Remark 2 (Structure-Agnostic vs Structure-Aware Schemes): We note that the proposed structure-aware schemes in this paper are not always better than the proposed structure-agnostic scheme. When a is very small, the structure-agnostic scheme outperforms the other schemes, because in this case the dimension of each matrix product is much less than the input matrices and thus it is more efficient to directly cache the matrix products. For example, if $\frac{N(N+1)}{2}\frac{g(a,a)}{a} < N$ (i.e., $a < \frac{2}{N+1}$) and $M = \frac{N(N+1)}{2}\frac{g(a,a)}{a}$, the achieved load of the structure-agnostic scheme is 0 (see also Remark 1), while the achieved loads of the structure-aware schemes are strictly larger than 0.

In general, see also Section IV-C for numerical evaluations, the row-partition scheme does not uniformly outperforms the column-partition scheme, or vice versa. Thus, for the proposed schemes, we cannot infer any uniform superiority of a certain placement strategy. □

Remark 3 (On Redundant Multicast Messages): In this paper's proposed coded caching schemes, after generating the coded symbols desired by the users, we use the MAN delivery scheme to generate multicast messages to deliver those coded symbols. Yu, Maddah-Ali and Avestimehr in [27] showed that some MAN multicast messages may be redundant when a file is requested by multiple users, and thus need not be transmitted. In our coded caching schemes, if there exist some products demanded by several users, we could use the approach in [27] to remove the redundant multicast messages. We do not report here this type of enhancement for sake of conciseness. □

Remark 4 (Extensions): Similarly to [12, Remark 3], we can extend the proposed schemes to Device-to-Device networks [3], where in the delivery phase each user broadcasts coded packets based on its cached content to all other users, and to the coded caching problem with private demands [28],

[29], where we aim to preserve the privacy of the demand of each user from other users. We do not report here this type of extensions for sake of conciseness. □

So far we looked at achievable schemes. We now turn to converse bounds. We can directly use the cut-set bounds in [2], [30] for the shared-link coded caching problem for single file retrieval into our problem, which leads to the following theorem.

Theorem 5 (Cut-Set Converse Bound): For the $(\mathsf{K}, \mathsf{N}, \mathsf{a})$ shared-link cache-aided matrix multiplication retrieval problem, we have

$$\mathsf{R}^\star \geq \max_{b\in[\min(\mathsf{N}',\mathsf{K})]} \left(b - b^2 \frac{\mathsf{M}}{\mathsf{N}'} \frac{\mathsf{a}}{g(\mathsf{a},\mathsf{a})} \right), \tag{17}$$

where $\mathsf{N}' = \left\lfloor \frac{\mathsf{N}}{2} \right\rfloor$.

Proof: For each $i \in [\mathsf{N}']$, define $W'_i = \mathbf{W}^T_{2(i-1)+1}\mathbf{W}_{2i}$. Consider a cut with $b \in [\min(\mathsf{N}', \mathsf{K})]$ users, and let each user demand one product W'_i where $i \in [\mathsf{N}']$. By using the cut-set bound in [2, Theorem 2], we have

$$\left\lfloor \frac{\mathsf{N}'}{b} \right\rfloor \mathsf{R}^\star \mathsf{B} + b\mathsf{Msr} \geq b \left\lfloor \frac{\mathsf{N}'}{b} \right\rfloor \mathsf{B}. \tag{18}$$

Then, by using the strategy in [30, By-product 1], we can remove the 'floor operator' in (18) and thus obtain (17). ■

When $\mathsf{a} \geq 1$ and $\mathsf{N} \geq 2\mathsf{K}$, we propose a novel genie-aided converse bound under the constraint of uncoded cache placement (proved in Appendix B), which smartly bounds the load by the converse bound in [27], [31] for the original MAN coded caching problem for single file retrieval. By using this novel converse bound, we have the following order optimality results.

Theorem 6 (Converse Bound and Order Optimality Result Under Uncoded Cache Placement): For the $(\mathsf{K}, \mathsf{N}, \mathsf{a})$ shared-link cache-aided matrix multiplication retrieval problem where $\mathsf{a} \geq 1$ and $\mathsf{N} \geq 2\mathsf{K}$, the worst-case load under the constraint of uncoded cache placement $\mathsf{R}^\star_{\mathrm{u}}$ is lower bounded by the lower convex envelop of

$$\left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{\mathsf{K}-t}{t+1} \frac{\mathsf{sr}}{f(\mathsf{r},\mathsf{s},\mathsf{r})} \right) = \left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{\mathsf{K}-t}{t+1} \frac{\mathsf{a}}{2\mathsf{a}-1} \right), \ \forall t \in [0 : \mathsf{K}]. \tag{19}$$

In addition, we have

$$\mathsf{R}^\star_{\mathrm{u}} \geq \frac{\mathsf{R}_2}{2} \geq \frac{\mathsf{R}_{\text{row}}}{2} \quad \text{when } \mathsf{a} \geq 1 \text{ and } \mathsf{N} \geq 2\mathsf{K}. \tag{20}$$

Note that the multiplicative gap between the converse bounds in Theorems 5 and 6 could be unbounded. For example, when $2\mathsf{a}$ divides K and $\mathsf{M} = \frac{2\mathsf{a}-1}{2\mathsf{a}}\mathsf{N}$, from Theorem 5 we have $\mathsf{R}^\star \geq 0$ and from Theorem 6 we have $\mathsf{R}^\star_{\mathrm{u}} \geq \frac{\mathsf{aK}/(2\mathsf{a}-1)}{\mathsf{K}(2\mathsf{a}-1)+2\mathsf{a}} > 0$. Hence, we cannot obtain the order optimality results in Theorem 6 from the cut-set converse bound in Theorem 5.

B. High-Level Strategies for Theorems 3 and 4

In this section we provide one simple example to highlight the key ideas in Theorems 3 and 4, in which we partition each matrix in the library by columns and by rows, respectively.

Example 1 (Case $\mathsf{a} \leq 1$): In this example, there are $\mathsf{K} = 2$ users and $\mathsf{N} = 4$ matrices of dimension $\mathsf{s} \times \mathsf{r} = 2 \times 2$ (i.e., $\mathsf{a} = 1$), where each user can store up to 8 symbols (i.e., $\mathsf{M} = 2$). Denote the four matrices as

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} d_1 & d_2 \\ d_3 & d_4 \end{bmatrix}.$$

For the delivery phase, assume that user 1 demands $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ and user 2 demands $\mathbf{C}^{\mathrm{T}}\mathbf{D}$, where

$$\mathbf{A}^{\mathrm{T}}\mathbf{B} = \begin{bmatrix} a_1b_1 + a_3b_3 & a_1b_2 + a_3b_4 \\ a_2b_1 + a_4b_3 & a_2b_2 + a_4b_4 \end{bmatrix} =: \begin{bmatrix} p_1 & p_2 \\ p_3 & p_4 \end{bmatrix}, \tag{21a}$$

$$\mathbf{C}^{\mathrm{T}}\mathbf{D} = \begin{bmatrix} c_1d_1 + c_3d_3 & c_1d_2 + c_3d_4 \\ c_2d_1 + c_4d_3 & c_2d_2 + c_4d_4 \end{bmatrix} =: \begin{bmatrix} q_1 & q_2 \\ q_3 & q_4 \end{bmatrix}. \tag{21b}$$

Next we compare the performances of our schemes.

1) Structure-agnostic scheme: In Theorem 1, we treat each matrix product as an independent file and use the MAN coded caching scheme for single file retrieval (for the case of $\mathsf{K} = 2$ users, $\frac{\mathsf{N}(\mathsf{N}+1)}{2} = 10$ files and cache size $\mathsf{M} = 2$ files) to transmit $28/5 = 5.6$ symbols.
2) Column-partition scheme: here we let user 1 cache the first column of each matrix (e.g., a_1 and a_3 for the first file and similarly for the other files), and let user 2 cache the second column of each matrix (e.g., a_2 and a_4 for the first file and similarly for the other files).

   Based on the cached content, p_1 in (21a) can be reconstructed by user 1 and q_4 in (21b) can be reconstructed by user 2. By (13) of Theorem 2, the server transmits the remaining three symbols in the matrix product desired by each user, for a total of 6 symbols.

   Based on the cached content, we further note that user 1 requests p_4 in (21a) that can be reconstructed by user 2, while user 2 requests q_1 in (21b) that can be reconstructed by user 1. Thus the server can transmit the coded symbol $p_4 + q_1$. Hence, the server only needs to totally transmit 5 symbols (i.e., the server transmits $(p_2, p_3, p_4 + q_1, q_2, q_3)$) as in Theorem 4.

   Note that in this scheme, each user directly recovers the desired "sum of products" symbols (e.g., $p_4 = a_2b_2 + a_4b_4$).
3) Row-partition scheme: here we use $\ell = \mathsf{K} = 2$, in which case the cache replication placement for Theorem 3 reduces to the MAN cache placement—the role of ℓ will be clarified further in Example 2 and Remark 5.

   By (14) of Theorem 2, each user directly recovers the two matrices that make up its desired matrix product. In other words, during the delivery phase user 1 recovers a_3, a_4, b_3, b_4, which are cached by user 2, and user 2 recovers c_1, c_2, d_1, d_2 which are cached by user 1. Hence, the servers transmits $a_3 + c_1, a_4 + c_2, b_3 + d_1, b_4 + d_2$, totally 4 symbols.

   To improve on the above, we let user 1 cache the first row of each matrix (e.g., a_1 and a_2 for the first file and similarly for the other files), and let user 2 cache the second row of each matrix (e.g., a_3 and a_4 for the first file and similarly for the other files). The server transmits

$$(a_3b_3+c_1d_1,\ a_3b_4+c_1d_2,\ a_4b_3+c_2d_1,\ a_4b_4+c_2d_2), \tag{22}$$

such that user 1 can recover $(a_3b_3, a_3b_4, a_4b_3, a_4b_4)$ and user 2 can recover $(c_1d_1, c_1d_2, c_2d_1, c_2d_2)$.

By leveraging the correlation of the elements in the products, we can further reduce the number of transmissions. Upon observing that

$$a_4b_4 = (a_3b_3)^{-1}(a_3b_4)(a_4b_3),$$
$$c_2d_2 = (c_1d_1)^{-1}(c_1d_2)(c_2d_1),$$

we do not need to transmit $a_4b_4 + c_2d_2$ in (22). Hence, we only need to transmit 3 symbols as in Theorem 3.

Note that in this scheme, each user recovers each individual term (e.g., a_3b_3 for user 1) in the "sum of products" symbols (e.g., $a_1b_1 + a_3b_3$).

□

To conclude, the high-level ideas for the row-partition and the column-partition schemes, as well as, their main advantages and limitations, are as follows:

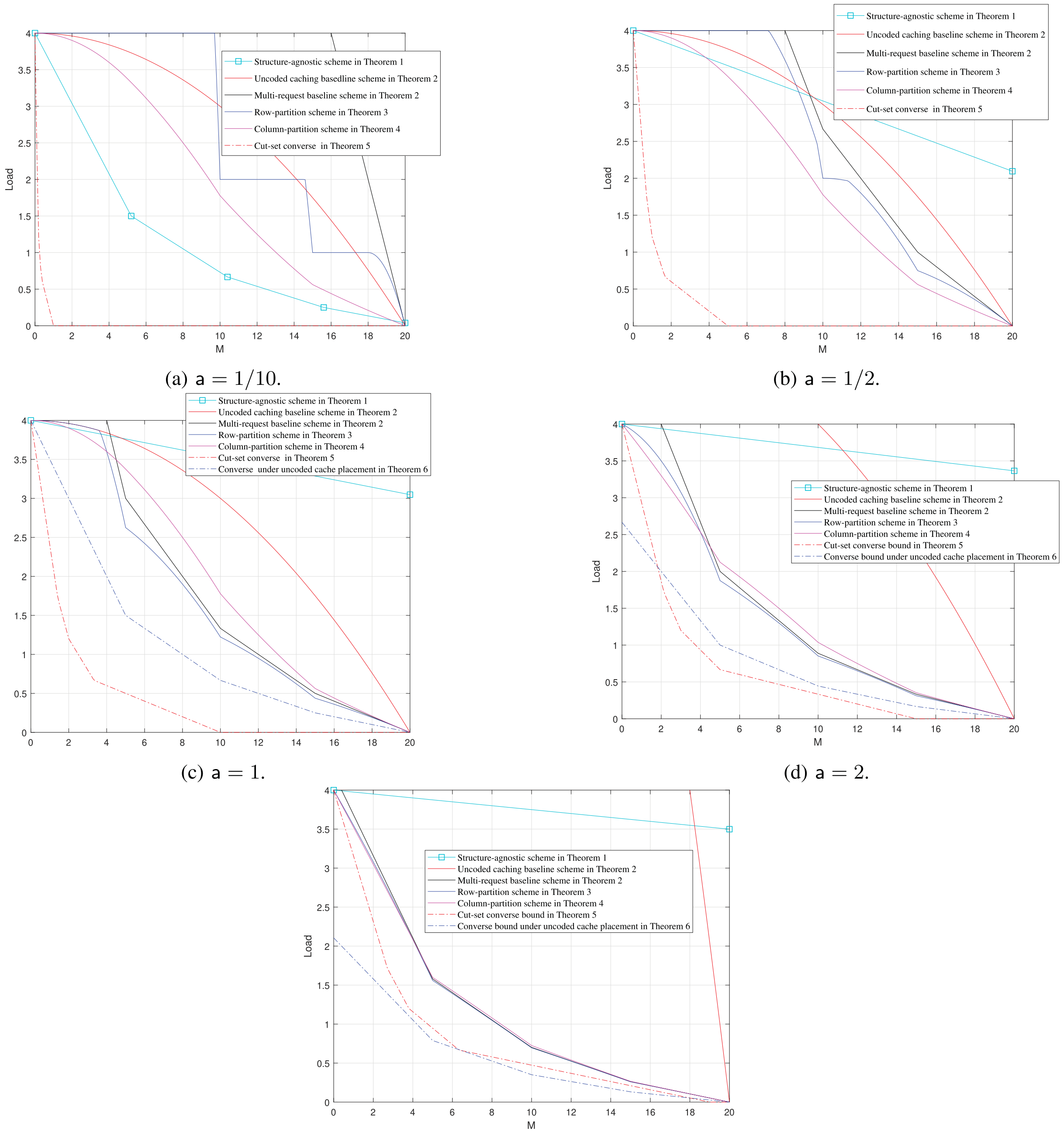
1) *Row-partition scheme.* The first approach partitions each matrix by rows and use the cache replication strategy in [32]. It will be explained in Remark 5 that, the cache replication strategy in the shared-link caching problem for single file retrieval aims to reduce the sub-packetization level compared to the MAN scheme. In our context, the proposed cache replication strategy with row-partition can reduce both the load and the sub-packetization level simultaneously.

   The matrix product desired by each user can be expressed by a sum of products of sub-matrices. By further encoding each term in the sum into a coded packet with length equal to its entropy, we then use the MAN delivery scheme to transmit the coded packets.
2) *Column-partition scheme.* The second approach partitions each matrix by columns. We separately consider the case $\mathsf{a} \leq 1$ and the case $\mathsf{a} > 1$. When $\mathsf{a} \leq 1$ (see the above example), we use the MAN cache placement strategy in [2] and propose a multi-round delivery scheme to transmit the coded packets. When $\mathsf{a} > 1$ (see Example 4), each demanded matrix product is not full rank; thus the entropy of each product is $(2\mathsf{a}-1)\mathsf{s}^2$ which is strictly less than the number of its elements $\mathsf{a}^2\mathsf{s}^2$, i.e., there exist some redundant elements in each product. Hence, we partition each matrix in the library into two blocks, where the cache placement of the first block is as in the MAN scheme and we propose to use a coded cache placement for the second block. In the delivery phase, each product is also partitioned into blocks and the correlation among blocks is taken into consideration during the encoding procedure.
3) *On types of placement.* We also remark that the structure-agnostic scheme uses an inter-file coded placement, where coding occurs across the symbols of all files (i.e., matrices). The row-partition scheme and the column-partition scheme for $\mathsf{a} \leq 1$ use uncoded cache placement. Finally, the column-partition scheme for $\mathsf{a} > 1$ uses an intra-file coded placement, where coding only occurs within the symbols of the same file.
4) *Advantages and limitations.* The main advantages and limitations of the proposed schemes are (see also Remarks 5 and 8):
   - *Row-partition scheme.* Its main advantage is that multicast opportunities are fully leveraged. In other words, if we need to transmit a requested symbol to a user and this symbol is cached by t other users, it is encoded in a multicast message with $t+1$ symbols, where each symbol is cached by t users and demanded by one user. However, each element in a desired matrix product is the sum of some products of the elements in the library matrices. The main limitation of the row-partition scheme is that each user recovers each individual product in the sum.
   - *Column-partition scheme.* Its main advantage is to let each user directly recover each element in the desired matrix product. Its main limitation is that multicast opportunities are not fully leveraged.
5) *Open problems.* In Theorem 6, we show that the proposed schemes are order optimal under uncoded cache placement for the case where $\mathsf{a} \geq 1$ and $\mathsf{N} \geq 2\mathsf{K}$. For the remaining cases, in particular for the case $\mathsf{a} < 1$, it is part of our on-going works to improve the proposed row-partition and column-partition schemes. This may be attained by using inter-file coded placements and by a new partition approach that has both the advantages of the row-partition and of the column-partition schemes, and overcomes their limitations. The derivation of a non-trivial converse bound for this case is also part of on-going works.

C. Numerical Evaluations

We now provide some numerical evaluations for the proposed schemes and converse bounds. In Fig. 1, we consider the case of $\mathsf{K} = 4$ users, $\mathsf{N} = 20$ files, and ratio $\mathsf{a} \in \left\{\frac{1}{10}, \frac{1}{2}, 1, 2, 10\right\}$. We observe the following from Fig. 1.

1) The row-partition scheme is always better than the multi-request baseline scheme, and the column-partition scheme is always better than the uncoded caching baseline scheme, as Corollary 1 shows.
2) When a is small, the performance of the multi-request baseline scheme is much worse than the proposed row-partition and column-partition schemes. This is because in the multi-request baseline scheme each user recovers the two library matrices of its desired matrix product, which has $2\mathsf{rs}$ symbols while the desired matrix product only has r^2 symbols, which is much lower than $2\mathsf{rs}$ when a is small.
3) When a is large, the performance of the uncoded caching baseline scheme is much worse than the proposed row-partition and column-partition schemes. This is because in the uncoded caching baseline scheme each user recovers all the r^2 symbols in the desired matrix



(a) $\mathsf{a} = 1/10$.

(b) $\mathsf{a} = 1/2$.

(c) $\mathsf{a} = 1$.

(d) $\mathsf{a} = 2$.

(e) $\mathsf{a} = 10$.

Fig. 1. Performance of various schemes for the shared-link cache-aided matrix multiplication retrieval problem with $\mathsf{K} = 4$ users and $\mathsf{N} = 20$ files for various values of the ratio a.

product. However, when a is large, r^2 is much larger than $f(\mathsf{r},\mathsf{s},\mathsf{r}) = 2\mathsf{sr} - \mathsf{s}^2$, which is the entropy of the matrix product.

4) The structure-agnostic scheme performs well when a is very small, since in this regime the entropy of each matrix product is much less than the entropy of each library matrix, and thus it is better to let the users directly cache the products.

5) The load v.s. cache size curves may not be convex. This is because in our setting we cannot memory-share between any two memory-load tradeoff points. For example, if we partition each matrix in the library into two parts and use a different cache placement strategy on each part, in the product of any two matrices there may exist some elements computed from both parts. In this case the computation of the matrix multiplication cannot be divided into two separate parts, each of which is based on one cache placement strategy.

D. Comparison to Existing Distributed Matrix Multiplication Computation Schemes for Straggler Mitigation

The distributed matrix multiplication problem has received much attention in the recent years. The problem is as follows. There are two uniformly i.i.d. matrices $\mathbf{A}$ of dimension $s \times r$ and $\mathbf{B}$ of dimension $s \times t$, where $s \geq \min(r, t)$. The matrix product $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ must be computed distributidely by a group of workers. There are mainly three strategies proposed in the literature, which partition each matrix into sub-matrices by rows [15], or by columns [14], or by blocks [13]. Each worker stores a linear combination of the sub-matrices in each matrix, and then computes the product of the two stored matrices, which is then sent to the master. From the transmissions of any T workers, the master must be able to correctly recover the matrix product. The objective is to characterize the minimum T, referred to as recovery threshold.

There are two main differences between our problem and the distributed matrix multiplication problem:

1) In our problem, there are multiple users receiving packets from the server, each of which caches some contents from the library and desires a product of two matrices. Hence, our problem is a *broadcast problem with side information*. By careful design, we aim to maximize the local caching gain (i.e., if some elements in the desired matrix product have already been cached, we need not transmit them in the delivery phase) and the coded caching multicasting gain. In contrast, in the distributed matrix multiplication computation problem, only the master wants to retrieve a product (no multicasting gain) and this master should recover the product only from the receiving packets (no local caching gain).
2) In the distributed matrix multiplication computation problem, it is usually assumed that $s \geq \min(r,t)$ (i.e., $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ is full rank). Hence, each element in the product $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ is also uniformly i.i.d. over $\mathbb{F}_{\mathsf{q}}$. The existing schemes let the master recover each element in the product individually (without leveraging the correlation among the elements in the product). Instead, our proposed schemes for this case (i.e., $\mathsf{a} \leq 1$) still leverage the correlation among the elements in each product (see Example 1). This is possible because each user cached some elements of each library matrix, and with this side information its desired product could be further compressed.

V. Novel Structure-Aware Achievable Schemes

A. Uncoded Caching Baseline Scheme: Proof of (13)

Placement phase: Each user caches the first $\frac{\mathsf{M}}{\mathsf{N}}\mathsf{r}$ columns of each of the N matrices in the library.

Delivery phase: User $k \in [\mathsf{K}]$ demands $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}\mathbf{W}_{d_{k,2}}$. Note that the first $\frac{\mathsf{M}}{\mathsf{N}}\mathsf{r}$ rows of $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}$ and the first $\frac{\mathsf{M}}{\mathsf{N}}\mathsf{r}$ columns of $\mathbf{W}_{d_{k,2}}$ are cached by user $k \in [\mathsf{K}]$. Hence, user $k \in [\mathsf{K}]$ can directly recover $\frac{\mathsf{M}^2\mathsf{r}^2}{\mathsf{N}^2}$ elements of $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}\mathbf{W}_{d_{k,2}}$. Then we let the server directly transmit the remaining $\left(1-\frac{\mathsf{M}^2}{\mathsf{N}^2}\right)\mathsf{r}^2$ elements of $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}\mathbf{W}_{d_{k,2}}$. Hence, the total load is

$$\mathsf{K}\left(1-\frac{\mathsf{M}^2}{\mathsf{N}^2}\right)\frac{\mathsf{r}^2}{f(\mathsf{r},\mathsf{s},\mathsf{r})} = \mathsf{K}\left(1-\frac{\mathsf{M}^2}{\mathsf{N}^2}\right)\frac{\mathsf{a}^2}{g(\mathsf{a},\mathsf{a})},$$

which coincides with (13).

B. Multi-Request Baseline Scheme: Proof of (14)

We treat each matrix in the library as a file with sr symbols, and use the coded caching scheme for multiple files retrieval in [26]. We focus on each cache size $\mathsf{M} = \frac{\mathsf{N}t}{\mathsf{K}}$, where $t \in [0:\mathsf{K}]$.

Placement phase: We divide the sr symbols of each matrix $\mathbf{W}_i$ into $\binom{\mathsf{K}}{t}$ non-overlapping and equal-length subfiles, $\mathbf{W}_i = \{W_{i,\mathcal{T}} : \mathcal{T} \subseteq [\mathsf{K}], |\mathcal{T}| = t\}$. Each subfile $W_{i,\mathcal{T}}$ contains $\frac{\mathsf{sr}}{\binom{\mathsf{K}}{t}}$ symbols and is cached exclusively by the users in $\mathcal{T}$.

Delivery phase: User $k \in [\mathsf{K}]$ demands $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}\mathbf{W}_{d_{k,2}}$. We let user $k \in [\mathsf{K}]$ recover $\mathbf{W}_{d_{k,1}}$ and $\mathbf{W}_{d_{k,2}}$. For each set $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t+1$, we let the server broadcast the pair of multicast messages

$$\sum_{k\in\mathcal{S}} W_{d_{k,1},\mathcal{S}\setminus\{k\}}, \quad \sum_{k\in\mathcal{S}} W_{d_{k,2},\mathcal{S}\setminus\{k\}}. \tag{23}$$

In $\sum_{k\in\mathcal{S}} W_{d_{k,1},\mathcal{S}\setminus\{k\}}$, user k stores all subfiles except $W_{d_{k,1},\mathcal{S}\setminus\{k\}}$ and thus it can recover this subfile. Similarly, user k can recover $W_{d_{k,2},\mathcal{S}\setminus\{k\}}$ from (23).

After considering all sets of users with cardinality $t+1$, each user can recover the two library matrices of its desired matrix product. The total load is

$$2\binom{\mathsf{K}}{t+1}\frac{\mathsf{sr}}{\binom{\mathsf{K}}{t}}\frac{1}{f(\mathsf{r},\mathsf{s},\mathsf{r})} = \frac{2(\mathsf{K}-t)\mathsf{a}}{(t+1)g(\mathsf{a},\mathsf{a})},$$

which coincides with (14).

C. Row-Partition Scheme: Proof of Theorem 3

We will start with a more detailed example than the one in Section IV-B to introduce the row-partition scheme in Theorem 3. Here, we partition each matrix in the library by rows and let each sub-matrix be cached by a set of users.

Example 2: Consider the $(\mathsf{K},\mathsf{N},\mathsf{a}) = (4, 20, 1/2)$ shared-link cache-aided matrix multiplication retrieval problem, with cache size $\mathsf{M} = 10$. We use the cache replication strategy in [32]. More precisely, we divide the 4 users into $\ell \in [4]$ groups and let the users in the same group cache the same content.

Case $\ell = 4$. First we consider the case $\ell = 4$, in which case the cache replication strategy in [32] is the same as the MAN cache placement strategy in [2]. By computing $t_4 = \left\lfloor \frac{4\mathsf{M}}{\mathsf{N}} \right\rfloor = 2$, we partition each matrix $\mathbf{W}_i$ where $i \in [20]$ into $\binom{\ell}{t_\ell} = 6$ sub-matrices as follows (the dimension of a matrix is shown in the subscript of its parenthesis)

$$(\mathbf{W}_i)_{\mathsf{s}\times\mathsf{r}} = \begin{bmatrix} (\mathbf{W}_{i,\{1,2\}})_{\mathsf{s}/6\times\mathsf{r}} \\ \hdashline (\mathbf{W}_{i,\{1,3\}})_{\mathsf{s}/6\times\mathsf{r}} \\ \hdashline (\mathbf{W}_{i,\{1,4\}})_{\mathsf{s}/6\times\mathsf{r}} \\ \hdashline (\mathbf{W}_{i,\{2,3\}})_{\mathsf{s}/6\times\mathsf{r}} \\ \hdashline (\mathbf{W}_{i,\{2,4\}})_{\mathsf{s}/6\times\mathsf{r}} \\ \hdashline (\mathbf{W}_{i,\{3,4\}})_{\mathsf{s}/6\times\mathsf{r}} \end{bmatrix}.$$

Each sub-matrix $\mathbf{W}_{i,\mathcal{T}}$ where $\mathcal{T} \subseteq [4]$ and $|\mathcal{T}| = 2$, is cached by users in $\mathcal{T}$. Thus, each user caches $20\times3\times\frac{\mathsf{sr}}{6} = 10\mathsf{sr} = \mathsf{Msr}$ symbols in total, thus satisfying the cache size constraint.

Assume that

$$[\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_4] = [1,2;3,4;5,6;7,8]. \tag{24}$$

The matrix product demanded by user 1 is

$$\begin{aligned}\mathbf{W}_1^{\mathrm{T}}\mathbf{W}_2 &= \sum_{\mathcal{T}\subseteq [4]:|\mathcal{T}|=2} \mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}\mathbf{W}_{2,\mathcal{T}} \\ &= \sum_{\mathcal{T}'\subseteq [4]:|\mathcal{T}'|=2,1\in\mathcal{T}'} \mathbf{W}_{1,\mathcal{T}'}^{\mathrm{T}}\mathbf{W}_{2,\mathcal{T}'} \\ &\quad + \sum_{\mathcal{T}\subseteq [4]:|\mathcal{T}|=2,1\notin\mathcal{T}} \mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}\mathbf{W}_{2,\mathcal{T}}. \end{aligned} \tag{25}$$

Note that the first term on the RHS of (25) is known by user 1 from its cache. Thus user 1 only needs to recover the second term. For each $\mathcal{T} \subseteq [4]$ where $|\mathcal{T}| = 2$ and $1 \notin \mathcal{T}$, $\mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}\mathbf{W}_{2,\mathcal{T}}$ is cached by the users in $\mathcal{T}$. In addition, $\mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}\mathbf{W}_{2,\mathcal{T}}$ can be encoded into $P(\mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}, \mathbf{W}_{2,\mathcal{T}})$ of size $f(\mathsf{r}, \mathsf{s}/6, \mathsf{r})$ symbols. Since $\mathsf{a} = \frac{\mathsf{r}}{\mathsf{s}} = \frac{1}{2}$, we have

$$f\left(\mathsf{r}, \frac{\mathsf{s}}{6}, \mathsf{r}\right) = f\left(\frac{\mathsf{s}}{2}, \frac{\mathsf{s}}{6}, \frac{\mathsf{s}}{2}\right) = \frac{\mathsf{s}}{6}\left(\frac{\mathsf{s}}{2} + \frac{\mathsf{s}}{2}\right) - \left(\frac{\mathsf{s}}{6}\right)^2 = \frac{5\mathsf{s}^2}{36}. \tag{26}$$

We will let user 1 recover $P(\mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}, \mathbf{W}_{2,\mathcal{T}})$ during the delivery phase.

After generating the coded symbols for each user, the server broadcasts

$$\sum_{k\in\mathcal{S}} P(\mathbf{W}_{d_{k,1},\mathcal{S}\setminus\{k\}}^{\mathrm{T}}, \mathbf{W}_{d_{k,2},\mathcal{S}\setminus\{k\}}), \tag{27}$$

for each set $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t_4 + 1 = 3$. Each user $k \in \mathcal{S}$ knows all the coded symbols in the sum (27) from its cache except $P(\mathbf{W}_{d_{k,1},\mathcal{S}\setminus\{k\}}^{\mathrm{T}}, \mathbf{W}_{d_{k,2},\mathcal{S}\setminus\{k\}})$, such that it can recover $P(\mathbf{W}_{d_{k,1},\mathcal{S}\setminus\{k\}}^{\mathrm{T}}, \mathbf{W}_{d_{k,2},\mathcal{S}\setminus\{k\}})$ and then recover $\mathbf{W}_{d_{k,1},\mathcal{S}\setminus\{k\}}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{S}\setminus\{k\}}$. For example, for $\mathcal{S} = \{1, 2, 3\}$, the server broadcasts

$$\begin{aligned} &P(\mathbf{W}_{1,\{2,3\}}^{\mathrm{T}}, \mathbf{W}_{2,\{2,3\}}) + P(\mathbf{W}_{3,\{1,3\}}^{\mathrm{T}}, \mathbf{W}_{4,\{1,3\}}) \\ &+ P(\mathbf{W}_{5,\{1,2\}}^{\mathrm{T}}, \mathbf{W}_{6,\{1,2\}}), \end{aligned} \tag{28}$$

and similarly for the remaining multicast messages. Hence, the server broadcasts $4f\left(\mathsf{r}, \frac{\mathsf{s}}{6}, \mathsf{r}\right) = \frac{5\mathsf{s}^2}{9}$ symbols in total, thus the achieved load is

$$\frac{5\mathsf{s}^2}{9f(\mathsf{r},\mathsf{s},\mathsf{r})} = \frac{5\mathsf{s}^2}{9f(\mathsf{s}/2,\mathsf{s},\mathsf{s}/2)} = \frac{20}{9}. \tag{29}$$

Case $\ell = 2$**.** Then, we consider the case $\ell = 2$. By computing $t_2 = \left\lfloor \frac{2\mathsf{M}}{\mathsf{N}} \right\rfloor = 1$, we partition each matrix $\mathbf{W}_i$ where $i \in [20]$ into $\binom{\ell}{t_\ell} = 2$ sub-matrices as

$$(\mathbf{W}_i)_{\mathsf{s}\times\mathsf{r}} = \left[\begin{array}{c} (\mathbf{W}_{i,\{1\}})_{\mathsf{s}/2\times\mathsf{r}} \\ \hdashline (\mathbf{W}_{i,\{2\}})_{\mathsf{s}/2\times\mathsf{r}} \end{array}\right].$$

We let users 1 and 3 cache $\mathbf{W}_{i,\{1\}}$, and let users 2 and 4 cache $\mathbf{W}_{i,\{2\}}$. In other words, we divide the users into two placement groups, where the first group contains users 1 and 3, and the second group contains users 2 and 4. The users in the same group have the same cache content. So each user caches $20 \times \frac{\mathsf{sr}}{2} = 10\mathsf{sr} = \mathsf{Msr}$ symbols, satisfying the cache size constraint.

During the delivery phase, we assume that the users' demands are given as in (24). The matrix product demanded by user 1 is

$$\mathbf{W}_1^{\mathrm{T}}\mathbf{W}_2 = \mathbf{W}_{1,\{1\}}^{\mathrm{T}}\mathbf{W}_{2,\{1\}} + \mathbf{W}_{1,\{2\}}^{\mathrm{T}}\mathbf{W}_{2,\{2\}}, \tag{30}$$

for which user 1 only needs to recover $\mathbf{W}_{1,\{2\}}^{\mathrm{T}}\mathbf{W}_{2,\{2\}}$. In addition, $\mathbf{W}_{1,\{2\}}^{\mathrm{T}}\mathbf{W}_{2,\{2\}}$ can be encoded into $P(\mathbf{W}_{1,\{2\}}^{\mathrm{T}}, \mathbf{W}_{2,\{2\}})$ of size $f\left(\mathsf{r}, \frac{\mathsf{s}}{2}, \mathsf{r}\right) = \frac{\mathsf{s}^2}{4}$ symbols.

After generating the coded symbols for each user, we divide the users into two transmission groups. In the first transmission group, we let the server satisfy the demands of users 1 and 2 by broadcasting

$$P(\mathbf{W}_{1,\{2\}}^{\mathrm{T}}, \mathbf{W}_{2,\{2\}}) + P(\mathbf{W}_{3,\{1\}}^{\mathrm{T}}, \mathbf{W}_{4,\{1\}}). \tag{31}$$

In the second transmission group, we let the server satisfy the demands of users 3 and 4 by broadcasting

$$P(\mathbf{W}_{5,\{2\}}^{\mathrm{T}}, \mathbf{W}_{6,\{2\}}) + P(\mathbf{W}_{7,\{1\}}^{\mathrm{T}}, \mathbf{W}_{8,\{1\}}). \tag{32}$$

Hence, the server broadcasts $2f\left(\mathsf{r}, \frac{\mathsf{s}}{2}, \mathsf{r}\right) = \frac{\mathsf{s}^2}{2}$ symbols in total, thus the achieved load is

$$\frac{\mathsf{s}^2}{2f(\mathsf{r},\mathsf{s},\mathsf{r})} = \frac{\mathsf{s}^2}{2\ f(\mathsf{s}/2,\mathsf{s},\mathsf{s}/2)} = 2.$$

Case $\ell = 1$**.** Similarly, when $\ell = 1$ (i.e., one single placement group) the achieved load is 4.

Case $\ell = 3$**.** When $\ell = 3$ the achieved load is $\frac{40}{9}$ (i.e., three placement groups).

All Cases Together. Hence, the minimum load achieved by the proposed row-partition scheme is 2 with $\ell = 2$, which is less than $64/21$, 3, and $8/3$ achieved by the structure-agnostic scheme in Theorem 1 and the two baseline structure-aware schemes in Theorem 2, respectively. □

Remark 5 (Row-Partition: $\ell = 4$ v.s. $\ell = 2$): In Example 2, when $\ell = 4$, each transmitted packet is a sum of $t_4 + 1 = 3$ coded symbols, while when $\ell = 2$, it is a sum of $t_2 + 1 = 2$ coded symbols. However, the latter attains the lowest load. This is because when $\ell = 4$, in order to recover $\sum_{\mathcal{T}\subseteq [4]:|\mathcal{T}|=t_4=2,1\notin\mathcal{T}} \mathbf{W}_{1,\mathcal{T}}^{\mathrm{T}}\mathbf{W}_{2,\mathcal{T}}$ in (25), we let user 1 recover each term in this sum, which increases the communication load. However, when $\ell = 2$, there is one set $\mathcal{T} \subseteq [2]$ where $|\mathcal{T}| = t_2 = 1$ and $1 \notin \mathcal{T}$, and this set is $\mathcal{T} = \{2\}$; thus we directly let user 1 recover $\mathbf{W}_{1,\{2\}}^{\mathrm{T}}\mathbf{W}_{2,\{2\}}$ in (30).

In other words, as already mentioned, the proposed row-partition scheme uses the cache replication placement in [32], which was proposed for the MAN shared-link caching problem for single file retrieval in order to reduce the sub-packetization at the expense of a higher load compared to the MAN scheme. However, in our row-partition approach for the considered cache-aided matrix multiplication retrieval problem, such a placement can simultaneously reduce the sub-packetization level and the load compared to the MAN cache placement.

□

We now generalize the proposed row-partition scheme in Example 2. We focus on each $\ell \in [\mathsf{K}]$.

Placement phase: We first compute $t_\ell = \left\lfloor \frac{\ell\mathsf{M}}{\mathsf{N}} \right\rfloor$ and $\alpha_\ell = t_\ell + 1 - \frac{\ell\mathsf{M}}{\mathsf{N}}$ as defined in (15c) and (15b), respectively. Among all the s rows of each matrix in the library, there are $\alpha_\ell\mathsf{s}$ rows cached by t_ℓ users, and $(1-\alpha_\ell)\mathsf{s}$ rows cached by $t_\ell + 1$ users, such that the average number of users caching each row is $\frac{\ell\mathsf{M}}{\mathsf{N}}$. More precisely, the first $\alpha_\ell\mathsf{s}$ rows of $\mathbf{W}_i$ where $i \in [\mathsf{N}]$ are partitioned into $\binom{\ell}{t_\ell}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,\mathcal{T}_1}$ where $\mathcal{T}_1 \subseteq [\ell]$ and $|\mathcal{T}_1| = t_\ell$. $\mathbf{W}_{i,\mathcal{T}_1}$ has dimension $\frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}} \times \mathsf{r}$. The remaining $(1-\alpha_\ell)\mathsf{s}$ rows of $\mathbf{W}_i$ are partitioned into $\binom{\ell}{t_\ell+1}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,\mathcal{T}_2}$ where $\mathcal{T}_2 \subseteq [\ell]$ and $|\mathcal{T}_2| = t_\ell + 1$. $\mathbf{W}_{i,\mathcal{T}_2}$ has dimension $\frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}} \times \mathsf{r}$. Each user $k \in [\mathsf{K}]$ caches $\mathbf{W}_{i,\mathcal{T}}$ where $i \in [\mathsf{N}]$, $\mathcal{T} \subseteq [\ell]$, $|\mathcal{T}| \in \{t_\ell, t_\ell+1\}$, and $\mathrm{Mod}(k,\ell) \in \mathcal{T}$.[3] Hence, user k caches (recall that $\mathsf{M} = \frac{t_\ell+1-\alpha_\ell}{\ell}\mathsf{N}$)

$$\mathsf{N}\left(\binom{\ell-1}{t_{\ell-1}}\frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}}\cdot \mathsf{r} + \binom{\ell-1}{t_\ell}\frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}}\cdot \mathsf{r}\right)$$

$$= \mathsf{Nsr}\left(\frac{t_\ell}{\ell}\alpha_\ell + \frac{t_\ell+1}{\ell}(1-\alpha_\ell)\right) \tag{33a}$$

$$= \mathsf{Nsr}\frac{t_\ell+1-\alpha_\ell}{\ell} = \mathsf{Msr} \text{ symbols,} \tag{33b}$$

satisfying the cache size constraint.

Note that if $\mathrm{Mod}(k_1,\ell) = \mathrm{Mod}(k_2,\ell)$ where $k_1, k_2 \in [\mathsf{K}]$, users k_1 and k_2 have the same cache content.

Delivery phase: For each $\ell \in [\mathsf{K}]$, we define

$$\mathcal{N}_\ell := \big\{\mathcal{T} \subseteq [\ell] : |\mathcal{T}| \in \{t_\ell, t_\ell+1\}\big\}, \tag{34}$$

and sort the sets in $\mathcal{N}_\ell$ in a lexicographic order. $\mathcal{N}_\ell(j)$ represents the j^{th} set in $\mathcal{N}_\ell$, where $j \in \left[\binom{\ell+1}{t_\ell+1}\right]$.[4]

We divide the users into $\left\lceil \frac{\mathsf{K}}{\ell} \right\rceil$ groups. More precisely, we let

$$\mathcal{G}_i = [(i-1)\ell+1 : i\ell], \quad \forall i \in \left[\left\lceil \frac{\mathsf{K}}{\ell} - 1 \right\rceil\right]; \tag{35a}$$

$$\mathcal{G}_{\lceil \frac{\mathsf{K}}{\ell} \rceil} = \left[\ell\left\lceil \frac{\mathsf{K}}{\ell} - 1\right\rceil + 1 : \mathsf{K}\right], \tag{35b}$$

where the first $\left\lceil \frac{\mathsf{K}}{\ell} - 1 \right\rceil$ groups contains ℓ users with different caches, and the last group contains $\mathsf{K} - \ell\left\lceil \frac{\mathsf{K}}{\ell} - 1\right\rceil$ users with different caches.

Let us focus on the transmission for group $\mathcal{G}_i$ where $i \in \left[\left\lceil \frac{\mathsf{K}}{\ell} \right\rceil\right]$. We sort the users in $\mathcal{G}_i$ in an increasing order and let $\mathcal{G}_i(j)$ be the j^{th} user.[5] For each user $k \in \mathcal{G}_i$, its desired matrix product can be expressed as

$$\mathbf{W}^{\mathrm{T}}_{d_{k,1}}\mathbf{W}_{d_{k,2}} = \begin{bmatrix} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{N}_\ell(1)} \\ \hdashline \vdots \\ \hdashline \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{N}_\ell\left(\binom{\ell+1}{t_\ell+1}\right)} \end{bmatrix} \left[\begin{array}{c;{2pt/2pt}c;{2pt/2pt}c} \mathbf{W}_{d_{k,2},\mathcal{N}_\ell(1)} & \cdots & \mathbf{W}_{d_{k,2},\mathcal{N}_\ell\left(\binom{\ell+1}{t_\ell+1}\right)} \end{array}\right] \tag{36a}$$

$$= \sum_{\mathcal{T}_1 \subseteq [\ell] : |\mathcal{T}_1| = t_\ell} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}\mathbf{W}_{d_{k,2},\mathcal{T}_1} + \sum_{\mathcal{T}_2 \subseteq [\ell] : |\mathcal{T}_2| = t_\ell+1} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}\mathbf{W}_{d_{k,2},\mathcal{T}_2} \tag{36b}$$

$$= \sum_{\mathcal{T}'_1 \subseteq [\ell] : |\mathcal{T}'_1| = t_\ell, \mathrm{Mod}(k,\ell) \in \mathcal{T}'_1} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}'_1}\mathbf{W}_{d_{k,2},\mathcal{T}'_1} + \sum_{\mathcal{T}_1 \subseteq [\ell] : |\mathcal{T}_1| = t_\ell, \mathrm{Mod}(k,\ell) \notin \mathcal{T}_1} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}\mathbf{W}_{d_{k,2},\mathcal{T}_1}$$
$$+ \sum_{\mathcal{T}'_2 \subseteq [\ell] : |\mathcal{T}'_2| = t_\ell+1, \mathrm{Mod}(k,\ell) \in \mathcal{T}'_2} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}'_2}\mathbf{W}_{d_{k,2},\mathcal{T}'_2} + \sum_{\mathcal{T}_2 \subseteq [\ell] : |\mathcal{T}_2| = t_\ell+1, \mathrm{Mod}(k,\ell) \notin \mathcal{T}_2} \mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}\mathbf{W}_{d_{k,2},\mathcal{T}_2}. \tag{36c}$$

We note that the first and third term on the RHS of (36c) can be re-constructed by the cached content of user k. Hence, user k only needs to recover the second and fourth terms in (36c) during the delivery phase. For each $\mathcal{T}_1 \subseteq [\ell]$ where $|\mathcal{T}_1| = t_\ell$ and $\mathrm{Mod}(k,\ell) \notin \mathcal{T}_1$, we can encode $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}\mathbf{W}_{d_{k,2},\mathcal{T}_1}$ into $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}, \mathbf{W}_{d_{k,2},\mathcal{T}_1}\right)$ of size

$$f\left(\mathsf{r}, \frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}}, \mathsf{r}\right) = g\left(\frac{\mathsf{r}\binom{\ell}{t_\ell}}{\alpha_\ell \mathsf{s}}, \frac{\mathsf{r}\binom{\ell}{t_\ell}}{\alpha_\ell \mathsf{s}}\right)\left(\frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}}\right)^2 \tag{37a}$$

$$= g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}, \frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}\right)\left(\frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}}\right)^2 \text{ symbols,} \tag{37b}$$

where $\mathsf{a} = \mathsf{r}/\mathsf{s}$, and $f(\cdot)$ and $g(\cdot)$ are defined in Section II. We will let user k recover $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}, \mathbf{W}_{d_{k,2},\mathcal{T}_1}\right)$ during the delivery phase. For each $\mathcal{T}_2 \subseteq [\ell]$ where $|\mathcal{T}_2| = t_\ell + 1$ and $\mathrm{Mod}(k,\ell) \notin \mathcal{T}_2$, we can encode $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}\mathbf{W}_{d_{k,2},\mathcal{T}_2}$ into $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}, \mathbf{W}_{d_{k,2},\mathcal{T}_2}\right)$ of size

$$f\left(\mathsf{r}, \frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}}, \mathsf{r}\right)$$
$$= g\left(\frac{\mathsf{r}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)\mathsf{s}}, \frac{\mathsf{r}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)\mathsf{s}}\right)\left(\frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}}\right)^2 \tag{38a}$$

$$= g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}, \frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}\right)\left(\frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}}\right)^2 \text{ symbols.} \tag{38b}$$

We will also let user k recover $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}, \mathbf{W}_{d_{k,2},\mathcal{T}_2}\right)$ during the delivery phase.

After generating the desired coded symbols for all users in $\mathcal{G}_i$, the server broadcasts

$$X_{i,\mathcal{S}_1} := \sum_{j \in \mathcal{S}_1} P\left(\mathbf{W}^{\mathrm{T}}_{d_{(i-1)\ell+j,1},\mathcal{S}_1 \setminus \{j\}}, \mathbf{W}_{d_{(i-1)\ell+j,2},\mathcal{S}_1 \setminus \{j\}}\right), \tag{39}$$

for each set $\mathcal{S}_1 \subseteq [\ell]$ where $\mathcal{S}_1 = t_\ell + 1$, such that user $(i-1)\ell + j$ can recover $P\left(\mathbf{W}^{\mathrm{T}}_{d_{(i-1)\ell+j,1},\mathcal{S}_1 \setminus \{j\}}, \mathbf{W}_{d_{(i-1)\ell+j,2},\mathcal{S}_1 \setminus \{j\}}\right)$ from $X_{i,\mathcal{S}_1}$, where $j \in \mathcal{S}_1$. Similarly, for each set $\mathcal{S}_2 \subseteq [\ell]$ where $\mathcal{S}_2 = t_\ell + 2$, the server broadcasts

$$X_{i,\mathcal{S}_2} := \sum_{j \in \mathcal{S}_2} P\left(\mathbf{W}^{\mathrm{T}}_{d_{(i-1)\ell+j,1},\mathcal{S}_2 \setminus \{j\}}, \mathbf{W}_{d_{(i-1)\ell+j,2},\mathcal{S}_2 \setminus \{j\}}\right), \tag{40}$$

such that user $(i-1)\ell + j$ can recover $P\left(\mathbf{W}^{\mathrm{T}}_{d_{(i-1)\ell+j,1},\mathcal{S}_2 \setminus \{j\}}, \mathbf{W}_{d_{(i-1)\ell+j,2},\mathcal{S}_2 \setminus \{j\}}\right)$. Hence, for the users in $\mathcal{G}_i$, the total number of symbols transmitted

[3] $\mathrm{Mod}(a,b)$ represents the modulo operation on a with integer quotient b. In this paper, if b divides a, we let $\mathrm{Mod}(a,b) = b$.

[4] From the Pascal's Triangle, we have $\binom{\ell}{t_\ell} + \binom{\ell}{t_\ell+1} = \binom{\ell+1}{t_\ell+1}$.

[5] If $j > |\mathcal{G}_i|$, we let $\mathcal{G}_i(j) = \emptyset$.

by the server is $\binom{\ell}{t_\ell+1} g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}, \frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}\right)\left(\frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}}\right)^2 + \binom{\ell}{t_\ell+2} g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}, \frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}\right)\left(\frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}}\right)^2$.

Considering all the $\lceil \frac{\mathsf{K}}{\ell} \rceil$ transmission groups, the total load is

$$\begin{aligned}
&\frac{\lceil \frac{\mathsf{K}}{\ell} \rceil}{f(\mathsf{r},\mathsf{s},\mathsf{r})}\left(\binom{\ell}{t_\ell+1} g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}, \frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}\right)\left(\frac{\alpha_\ell \mathsf{s}}{\binom{\ell}{t_\ell}}\right)^2\right.\\
&\left.+\binom{\ell}{t_\ell+2} g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}, \frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}\right)\left(\frac{(1-\alpha_\ell)\mathsf{s}}{\binom{\ell}{t_\ell+1}}\right)^2\right)\\
&=\frac{\lceil \frac{\mathsf{K}}{\ell} \rceil}{g(\mathsf{a},\mathsf{a})}\left(\binom{\ell}{t_\ell+1} g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}, \frac{\mathsf{a}\binom{\ell}{t_\ell}}{\alpha_\ell}\right)\left(\frac{\alpha_\ell}{\binom{\ell}{t_\ell}}\right)^2\right.\\
&\left.+\binom{\ell}{t_\ell+2} g\left(\frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}, \frac{\mathsf{a}\binom{\ell}{t_\ell+1}}{(1-\alpha_\ell)}\right)\left(\frac{(1-\alpha_\ell)}{\binom{\ell}{t_\ell+1}}\right)^2\right),
\end{aligned} \tag{41}$$

which coincides with (15a).

Remark 6 (Row-Partition Scheme v.s. Multi-Request Baseline Scheme): If we let $\ell = \mathsf{K}$, encode $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}\mathbf{W}_{d_{k,2},\mathcal{T}_1}$ into the concatenation of all the symbols in $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}$ and $\mathbf{W}_{d_{k,2},\mathcal{T}_1}$ whose length is strictly larger than the length of $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_1}, \mathbf{W}_{d_{k,2},\mathcal{T}_1}\right)$, and encode $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}\mathbf{W}_{d_{k,2},\mathcal{T}_2}$ into the concatenation of all the symbols in $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}$ and $\mathbf{W}_{d_{k,2},\mathcal{T}_2}$ whose length is strictly larger than the length of $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}_2}, \mathbf{W}_{d_{k,2},\mathcal{T}_2}\right)$, then it is equivalent to let each user recover the two library matrices of its desired matrix product; thus the proposed row-partition scheme becomes the multi-request baseline scheme for Theorem 2. Therefore, the proposed row-partition scheme is strictly better than the multi-request baseline scheme when $\mathsf{M} < \mathsf{N}$. □

Remark 7 (Application of Other Shared-Link Coded Caching Schemes): Obviously, with the proposed row-partition strategy, we can apply any coded caching scheme with uncoded cache placement for the original MAN coded caching problem for single file retrieval to the considered cache-aided matrix multiplication retrieval problem. More precisely, for any existing scheme with uncoded cache placement for the single file retrieval problem, each file W'_i where $i \in [\mathsf{N}]$ is divided into non-overlapping subfiles, $W'_i = \{W'_{i,\mathcal{T}} : \mathcal{T} \subseteq [\mathsf{K}]\}$. In the considered matrix multiplication retrieval problem, we can partition each matrix $\mathbf{W}_i$ into 2^{K} sub-matrix by rows, each sub-matrix denoted by $\mathbf{W}_{i,\mathcal{T}}$ of dimension $\frac{\mathsf{s}|W'_{i,\mathcal{T}}|}{|W'_i|} \times \mathsf{r}$. We then encode $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}}\mathbf{W}_{d_{k,2},\mathcal{T}}$ into $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}}\mathbf{W}_{d_{k,2},\mathcal{T}}\right)$ symbols. Finally, we use the delivery phase of this existing scheme to deliver $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{T}}\mathbf{W}_{d_{k,2},\mathcal{T}}\right)$ as delivering $W'_{d_k,\mathcal{T}}$ in the original file retrieval problem, where d_k represents the desired file of user k. □

D. Column-Partition Scheme: Proof of Theorem 4

We continue Example 2 to introduce the column-partition scheme in Theorem 4. Here we partition each matrix in the library by columns and let each sub-matrix be cached by a set of users.

Example 3: Recall that we consider the $(\mathsf{K},\mathsf{N},\mathsf{a}) = (4, 20, 1/2)$ shared-link cache-aided matrix multiplication retrieval problem with cache size $\mathsf{M} = 10$.

Placement phase: We use the MAN cache placement in [2]. With $t_{\mathsf{K}} = \lfloor \frac{\mathsf{KM}}{\mathsf{N}} \rfloor = 2$, we partition each matrix $\mathbf{W}_i$ where $i \in [20]$ into $\binom{\mathsf{K}}{t_{\mathsf{K}}} = 6$ sub-matrices as follows, $\mathbf{W}_i = \left[\mathbf{W}_{i,\{1,2\}}, \mathbf{W}_{i,\{1,3\}}, \mathbf{W}_{i,\{1,4\}}, \mathbf{W}_{i,\{2,3\}}, \mathbf{W}_{i,\{2,4\}}, \mathbf{W}_{i,\{3,4\}}\right]$, where sub-matrix $\mathbf{W}_{i,\mathcal{T}}$ of dimension $\mathsf{s} \times \frac{\mathsf{r}}{6}$, for $\mathcal{T} \subseteq [4]$ and $|\mathcal{T}| = 2$, is cached by users in $\mathcal{T}$. Each user thus caches $20 \times 3 \times \frac{\mathsf{sr}}{6} = 10\mathsf{sr} = \mathsf{Msr}$ symbols, satisfying the cache size constraint.

Delivery phase: Assume that the users' demands are as in (24). The matrix product demanded by user 1 can be expressed as

$$(\mathbf{W}^{\mathrm{T}}_1\mathbf{W}_2)_{\mathsf{r}\times\mathsf{r}} = \left[\begin{array}{c:c:c}
(\mathbf{W}^{\mathrm{T}}_{1,\{1,2\}}\mathbf{W}_{2,\{1,2\}})_{\frac{\mathsf{r}}{6}\times\frac{\mathsf{r}}{6}} & \cdots & (\mathbf{W}^{\mathrm{T}}_{1,\{1,2\}}\mathbf{W}_{2,\{3,4\}})_{\frac{\mathsf{r}}{6}\times\frac{\mathsf{r}}{6}} \\ \hdashline
(\mathbf{W}^{\mathrm{T}}_{1,\{1,3\}}\mathbf{W}_{2,\{1,2\}})_{\frac{\mathsf{r}}{6}\times\frac{\mathsf{r}}{6}} & \cdots & (\mathbf{W}^{\mathrm{T}}_{1,\{1,3\}}\mathbf{W}_{2,\{3,4\}})_{\frac{\mathsf{r}}{6}\times\frac{\mathsf{r}}{6}} \\ \hdashline
\vdots & \ddots & \vdots \\ \hdashline
(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}\mathbf{W}_{2,\{1,2\}})_{\frac{\mathsf{r}}{6}\times\frac{\mathsf{r}}{6}} & \cdots & (\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}\mathbf{W}_{2,\{3,4\}})_{\frac{\mathsf{r}}{6}\times\frac{\mathsf{r}}{6}}
\end{array}\right]. \tag{42}$$

Each sub-matrix $\mathbf{W}^{\mathrm{T}}_{1,\mathcal{T}_1}\mathbf{W}_{2,\mathcal{T}_2}$ in (42) where $\mathcal{T}_1, \mathcal{T}_2 \subseteq [4]$ and $|\mathcal{T}_1| = |\mathcal{T}_2| = 2$, is then encoded into $P(\mathbf{W}^{\mathrm{T}}_{1,\mathcal{T}_1}, \mathbf{W}_{2,\mathcal{T}_2})$ of $f\left(\frac{\mathsf{r}}{6}, \mathsf{s}, \frac{\mathsf{r}}{6}\right) = f\left(\frac{\mathsf{s}}{12}, \mathsf{s}, \frac{\mathsf{s}}{12}\right) = \frac{\mathsf{s}^2}{144}$ symbols. Note that $P(\mathbf{W}^{\mathrm{T}}_{1,\mathcal{T}_1}, \mathbf{W}_{2,\mathcal{T}_2})$ can be directly re-constructed by each user in $\mathcal{T}_1 \cap \mathcal{T}_2$ from their cached content. Hence, during the delivery phase user 1 needs to recover $P(\mathbf{W}^{\mathrm{T}}_{1,\mathcal{T}_1}, \mathbf{W}_{2,\mathcal{T}_2})$ where $1 \notin (\mathcal{T}_1 \cap \mathcal{T}_2)$. We divide the coded symbols desired by user 1 into groups, such that $F_{1,\mathcal{V}}$ represents the set of coded symbols desired by user 1 and uniquely known by users in $\mathcal{V}$. More precisely, we have

$$\begin{aligned}
F_{1,\emptyset} = \Big\{&P(\mathbf{W}^{\mathrm{T}}_{1,\{1,2\}}, \mathbf{W}_{2,\{3,4\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{1,3\}}, \mathbf{W}_{2,\{2,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{1,4\}}, \mathbf{W}_{2,\{2,3\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{2,3\}}, \mathbf{W}_{2,\{1,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{2,4\}}, \mathbf{W}_{2,\{1,3\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}, \mathbf{W}_{2,\{1,2\}})\Big\};
\end{aligned} \tag{43a}$$

$$\begin{aligned}
F_{1,\{2\}} = \Big\{&P(\mathbf{W}^{\mathrm{T}}_{1,\{1,2\}}, \mathbf{W}_{2,\{2,3\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{1,2\}}, \mathbf{W}_{2,\{2,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{2,3\}}, \mathbf{W}_{2,\{1,2\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{2,3\}}, \mathbf{W}_{2,\{2,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{2,4\}}, \mathbf{W}_{2,\{1,2\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{2,4\}}, \mathbf{W}_{2,\{2,3\}})\Big\};
\end{aligned} \tag{43b}$$

$$\begin{aligned}
F_{1,\{3\}} = \Big\{&P(\mathbf{W}^{\mathrm{T}}_{1,\{1,3\}}, \mathbf{W}_{2,\{2,3\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{1,3\}}, \mathbf{W}_{2,\{3,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{2,3\}}, \mathbf{W}_{2,\{1,3\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{2,3\}}, \mathbf{W}_{2,\{3,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}, \mathbf{W}_{2,\{1,3\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}, \mathbf{W}_{2,\{2,3\}})\Big\};
\end{aligned} \tag{43c}$$

$$\begin{aligned}
F_{1,\{4\}} = \Big\{&P(\mathbf{W}^{\mathrm{T}}_{1,\{1,4\}}, \mathbf{W}_{2,\{2,4\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{1,4\}}, \mathbf{W}_{2,\{3,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{2,4\}}, \mathbf{W}_{2,\{1,4\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{2,4\}}, \mathbf{W}_{2,\{3,4\}}),\\
&P(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}, \mathbf{W}_{2,\{1,4\}}),\ P(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}, \mathbf{W}_{2,\{2,4\}})\Big\};
\end{aligned} \tag{43d}$$

$$F_{1,\{2,3\}} = \Big\{P(\mathbf{W}^{\mathrm{T}}_{1,\{2,3\}}, \mathbf{W}_{2,\{2,3\}})\Big\}; \tag{43e}$$

$$F_{1,\{2,4\}} = \Big\{P(\mathbf{W}^{\mathrm{T}}_{1,\{2,4\}}, \mathbf{W}_{2,\{2,4\}})\Big\}; \tag{43f}$$

$$F_{1,\{3,4\}} = \Big\{P(\mathbf{W}^{\mathrm{T}}_{1,\{3,4\}}, \mathbf{W}_{2,\{3,4\}})\Big\}. \tag{43g}$$

From (43), and similarly for the other users, we have

$$|F_{2,\emptyset}| = |F_{3,\emptyset}| = |F_{4,\emptyset}| = \frac{\mathsf{s}^2}{24}; \tag{44a}$$

$$|F_{2,\{1\}}| = |F_{2,\{3\}}| = |F_{2,\{4\}}| = |F_{3,\{1\}}| = |F_{3,\{2\}}| \\ = |F_{3,\{4\}}| = |F_{4,\{1\}}| = |F_{4,\{2\}}| = |F_{4,\{3\}}| = \frac{\mathsf{s}^2}{24}; \tag{44b}$$

$$|F_{2,\{1,3\}}| = |F_{2,\{1,4\}}| = |F_{2,\{3,4\}}| = |F_{3,\{1,2\}}| = |F_{3,\{1,4\}}| \\ = |F_{3,\{2,4\}}| = |F_{4,\{1,2\}}| = |F_{4,\{1,3\}}| = |F_{4,\{2,3\}}| = \frac{\mathsf{s}^2}{144}. \tag{44c}$$

Next we divide the transmission into three rounds. In the first round, the server broadcasts

$$F_{1,\emptyset}, \quad F_{2,\emptyset}, \quad F_{3,\emptyset}, \quad F_{4,\emptyset}, \tag{45}$$

for a total of $\frac{4\mathsf{s}^2}{24} = \frac{\mathsf{s}^2}{6}$ symbols. In the second round, the server broadcasts

$$F_{1,\{2\}} + F_{2,\{1\}}, \quad F_{1,\{3\}} + F_{3,\{1\}}, \quad F_{1,\{4\}} + F_{4,\{1\}}, \\ F_{2,\{3\}} + F_{3,\{2\}}, \quad F_{2,\{4\}} + F_{4,\{2\}}, \quad F_{3,\{4\}} + F_{4,\{3\}}, \tag{46}$$

for a total of $\frac{6\mathsf{s}^2}{24} = \frac{\mathsf{s}^2}{4}$ symbols. In the third round, the server broadcasts

$$F_{1,\{2,3\}} + F_{2,\{1,3\}} + F_{3,\{1,2\}}, \quad F_{1,\{2,4\}} + F_{2,\{1,4\}} + F_{4,\{1,2\}}, \\ F_{1,\{3,4\}} + F_{3,\{1,4\}} + F_{4,\{1,3\}}, \quad F_{2,\{3,4\}} + F_{3,\{2,4\}} + F_{4,\{2,3\}}, \tag{47}$$

for a total of $\frac{4\mathsf{s}^2}{144} = \frac{\mathsf{s}^2}{36}$ symbols. Hence, the achieved load is

$$\frac{\frac{\mathsf{s}^2}{6} + \frac{\mathsf{s}^2}{4} + \frac{\mathsf{s}^2}{36}}{f(\mathsf{r}, \mathsf{s}, \mathsf{r})} = \frac{\frac{\mathsf{s}^2}{6} + \frac{\mathsf{s}^2}{4} + \frac{\mathsf{s}^2}{36}}{f(\mathsf{s}/2, \mathsf{s}, \mathsf{s}/2)} = \frac{16}{9},$$

which is less than all other schemes. □

Remark 8 (Row-Partition With $\ell = \mathsf{K}$ v.s. Column-Partition): We now compare the row-partition scheme with $\ell = \mathsf{K} = 4$ and the column-partition scheme through the above example. In both schemes, each sub-matrix in the library matrices is cached by $t_4 = 2$ users. The main advantage of the row-partition scheme with $\ell = 4$ is that each transmitted packet is a sum of $t_4 + 1 = 3$ coded symbols, while most packets transmitted by the column-partition scheme are the sums of $t_4 = 2$ coded symbols. However, each element in the desired matrix product by each user is a sum of some products of the elements in the library matrices. Instead of letting the user recover each individual product in the sum as in the row-partition scheme (e.g., we let user 1 recover each individual product in the sum (25)), the column-partition scheme directly lets the user recover this sum (e.g., we let user 1 recover each term in the product matrix (42)).

To conclude, as mentioned already in Section IV-B, the main advantage of the row-partition scheme is to fully leverage the multicast opportunities, while the main advantage of the column-partition scheme is to let each user directly recover each element in the product. □

We then generalize the column-partition scheme in Example 3.

1) $\mathsf{a} \leq 1$: Let us first consider the case where $\mathsf{a} \leq 1$ (i.e., $\mathsf{r} \leq \mathsf{s}$).

Placement phase: Let $t_{\mathsf{K}} = \left\lfloor \frac{\mathsf{KM}}{\mathsf{N}} \right\rfloor$ and $\alpha_{\mathsf{K}} = t_{\mathsf{K}} + 1 - \frac{\mathsf{KM}}{\mathsf{N}}$. Among all the r columns of each matrix in the library, there $\alpha_{\mathsf{K}}\mathsf{r}$ columns cached by t_{K} users, and $(1 - \alpha_{\mathsf{K}})\mathsf{r}$ columns cached by $t_{\mathsf{K}} + 1$ users, such that the average number of users caching each column is $\frac{\mathsf{KM}}{\mathsf{N}}$. More precisely, the first $\alpha_{\mathsf{K}}\mathsf{r}$ columns of $\mathbf{W}_i$ where $i \in [\mathsf{N}]$ are partitioned into $\binom{\mathsf{K}}{t_{\mathsf{K}}}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,\mathcal{T}_1}$ where $\mathcal{T}_1 \subseteq [\mathsf{K}]$ and $|\mathcal{T}_1| = t_{\mathsf{K}}$. $\mathbf{W}_{i,\mathcal{T}_1}$ has dimension $\mathsf{s} \times \frac{\alpha_{\mathsf{K}}\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}$. The remaining $(1 - \alpha_{\mathsf{K}})\mathsf{r}$ columns of $\mathbf{W}_i$ are partitioned into $\binom{\mathsf{K}}{t_{\mathsf{K}}+1}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,\mathcal{T}_2}$ where $\mathcal{T}_2 \subseteq [\mathsf{K}]$ and $|\mathcal{T}_2| = t_{\mathsf{K}} + 1$. $\mathbf{W}_{i,\mathcal{T}_2}$ has dimension $\mathsf{s} \times \frac{(1-\alpha_{\mathsf{K}})\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}$.

Each user $k \in [\mathsf{K}]$ caches $\mathbf{W}_{i,\mathcal{T}}$ where $i \in [\mathsf{N}]$, $\mathcal{T} \subseteq [\mathsf{K}]$, $|\mathcal{T}| \in \{t_{\mathsf{K}}, t_{\mathsf{K}} + 1\}$, and $k \in \mathcal{T}$. Hence, user $k \in [\mathsf{K}]$ caches

$$\mathsf{N}\left(\binom{\mathsf{K}-1}{t_{\mathsf{K}-1}}\mathsf{s} \cdot \frac{\alpha_{\mathsf{K}}\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}} + \binom{\mathsf{K}-1}{t_{\mathsf{K}}}\mathsf{s} \cdot \frac{(1-\alpha_{\mathsf{K}})\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right) \\ = \mathsf{Nsr}\left(\frac{t_{\mathsf{K}}}{\mathsf{K}}\alpha_{\mathsf{K}} + \frac{t_{\mathsf{K}}+1}{\mathsf{K}}(1-\alpha_{\mathsf{K}})\right) \tag{48a}$$

$$= \mathsf{Nsr}\frac{t_{\mathsf{K}} + 1 - \alpha_{\mathsf{K}}}{\mathsf{K}} = \mathsf{Msr} \ \ \text{symbols}, \tag{48b}$$

thus satisfying the cache size constraint.

Delivery phase: Recall from (34) that $\mathcal{N}_{\mathsf{K}} := \big\{\mathcal{T} \subseteq [\mathsf{K}] : |\mathcal{T}| \in \{t_{\mathsf{K}}, t_{\mathsf{K}} + 1\}\big\}$, where $|\mathcal{N}_{\mathsf{K}}| = \binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}$. Let $\mathcal{N}_{\mathsf{K}}(j)$ represents the j^{th} set in $\mathcal{N}_{\mathsf{K}}$, where $j \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$.

The matrix product desired by user $k \in [\mathsf{K}]$, $\mathbf{W}_{d_{k,1}}^{\mathrm{T}}\mathbf{W}_{d_{k,2}}$, can be expressed in (49), shown at the bottom of the next page.

For any pair (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$,

- if $k \in \mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2)$, $\mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(j_1)}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(j_2)}$ can be reconstructed by user k from its cached content;
- otherwise, we encode $\mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(j_1)}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(j_2)}$ into $P\left(\mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(j_1)}^{\mathrm{T}}, \mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(j_2)}\right)$. We then add $P\left(\mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(j_1)}^{\mathrm{T}}, \mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(j_2)}\right)$ into $F_{k,\mathcal{N}_{\mathsf{K}}(j_1)\cap\mathcal{N}_{\mathsf{K}}(j_2)}$, which represents the set of coded symbols desired by user k that can be reconstructed by users in $\mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2)$.

The following lemma is proved in Appendix C.

Lemma 1: For each $i \in [0 : t_{\mathsf{K}} + 1]$ and $k \in [\mathsf{K}]$, we have

$$|F_{k,\mathcal{V}}| = \left(\left(\frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}\right)^2 \binom{\mathsf{K}-i}{t_{\mathsf{K}}-i}\binom{\mathsf{K}-t_{\mathsf{K}}}{t_{\mathsf{K}}-i} \right. \\ + \left(\frac{(1-\alpha_{\mathsf{K}})\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right)^2 \binom{\mathsf{K}-i}{t_{\mathsf{K}}+1-i}\binom{\mathsf{K}-t_{\mathsf{K}}-1}{t_{\mathsf{K}}+1-i} + \\ \left. +2\frac{\alpha_{\mathsf{K}}(1-\alpha_{\mathsf{K}})\mathsf{a}^2}{\binom{\mathsf{K}}{t_{\mathsf{K}}}\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\binom{\mathsf{K}-i}{t_{\mathsf{K}}-i}\binom{\mathsf{K}-t_{\mathsf{K}}}{t_{\mathsf{K}}+1-i}\right)\mathsf{s}^2, \tag{50}$$

for all $\mathcal{V} \subseteq ([\mathsf{K}] \setminus \{k\})$ where $|\mathcal{V}| = i$.

In other words, the length of $F_{k,\mathcal{V}}$ only depends on $|\mathcal{V}|$. Hence, we define $f_{i,\mathsf{a}}$ as the RHS of (50), representing the length of each $F_{k,\mathcal{V}}$ where $|\mathcal{V}| = i$.

The transmission is divided into $t_{\mathsf{K}}+2$ rounds. In round $i \in [0 : t_{\mathsf{K}}+1]$, for each $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = i+1$, the server broadcasts

$$X_{\mathcal{S}} = \sum_{k\in\mathcal{S}} F_{k,\mathcal{S}\setminus\{k\}}, \tag{51}$$

such that each user $k \in \mathcal{S}$ can recover $F_{k,\mathcal{S}\setminus\{k\}}$.

Considering all the $t_{\mathsf{K}}+2$ rounds, the total load is

$$\frac{\sum_{i\in[0:t_{\mathsf{K}}+1]} \binom{\mathsf{K}}{i+1} f_{i,\mathsf{a}}}{f(\mathsf{r},\mathsf{s},\mathsf{r})} = \frac{\sum_{i\in[0:t_{\mathsf{K}}+1]} \binom{\mathsf{K}}{i+1} f_{i,\mathsf{a}}}{\mathsf{a}^2\mathsf{s}^2}, \tag{52}$$

where (52) follows from $\mathsf{a} \leq 1$. From (52), we prove (16a) for the case where $\mathsf{a} \leq 1$.

2) $\mathsf{a} > 1$: We then consider the case where $\mathsf{a} > 1$ (i.e., $\mathsf{r} > \mathsf{s}$). In this case, each demanded matrix product is not full-rank. So compared to the proposed column-partition scheme for $\mathsf{a} \leq 1$, we will use a novel coded cache placement and some additional steps in the delivery phase to deal with the rank deficiency. We first use the following example to illustrate the key ideas.

Example 4: Consider the case of $\mathsf{K} = 2$ users, $\mathsf{N} = 4$ matrices of dimension $\mathsf{s} \times \mathsf{r} = 2 \times 4$ (i.e., $\mathsf{a} = 2$), and that each user can store up to 16 symbols (i.e., $\mathsf{M} = 2$). Assume that the four matrices are $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$. We express the matrix $\mathbf{A}$ as follows

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \end{bmatrix} = \mathbf{A}_1 \left[\mathbf{I}_2, \mathbf{A}_1^{-1}\mathbf{A}_2\right],$$
$$\mathbf{A}_1 := \begin{bmatrix} a_1 & a_2 \\ a_5 & a_6 \end{bmatrix}, \mathbf{A}_2 := \begin{bmatrix} a_3 & a_4 \\ a_7 & a_8 \end{bmatrix},$$

where we assumed that block $\mathbf{A}_1$ is full rank (this is true with high probability when the filed size is large); same for the remaining matrices. Note that the general column-partition scheme described later also works for the case where $\mathbf{A}_1$ is not full rank; thus for arbitrary finite field, the proposed scheme also works.

Placement phase: user 1 caches $\begin{bmatrix} a_1 \\ a_5 \end{bmatrix}$ and $\mathbf{A}_1^{-1}\begin{bmatrix} a_3 \\ a_7 \end{bmatrix}$, and user 2 caches $\begin{bmatrix} a_2 \\ a_6 \end{bmatrix}$ and $\mathbf{A}_1^{-1}\begin{bmatrix} a_4 \\ a_8 \end{bmatrix}$; similarly for the other matrices. Hence, each user caches 4 symbols from each matrix; thus each user caches 16 symbols in total.

Delivery phase: Assume that the users 1 and 2 demand

$$\mathbf{A}^{\mathrm{T}}\mathbf{B} = \left[\begin{array}{c:c} \mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1 & \mathbf{A}_1^{\mathrm{T}}\mathbf{B}_2 \\ \hdashline \mathbf{A}_2^{\mathrm{T}}\mathbf{B}_1 & \mathbf{A}_2^{\mathrm{T}}\mathbf{B}_2 \end{array}\right], \mathbf{C}^{\mathrm{T}}\mathbf{D} = \left[\begin{array}{c:c} \mathbf{C}_1^{\mathrm{T}}\mathbf{D}_1 & \mathbf{C}_1^{\mathrm{T}}\mathbf{D}_2 \\ \hdashline \mathbf{C}_2^{\mathrm{T}}\mathbf{D}_1 & \mathbf{C}_2^{\mathrm{T}}\mathbf{D}_2 \end{array}\right],$$

respectively, where each matrix product contains 4 blocks. The delivery phase of the column-partition scheme contains three steps:

- In the first step, we let user 1 recover $\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1$ and let user 2 recover $\mathbf{C}_1^{\mathrm{T}}\mathbf{D}_1$. The delivery is exactly the same as the column-partition scheme in the previous example for $\mathsf{a} \leq 1$. Thus, we need to transmit 5 symbols.
- In the second step, we let user 1 and user 2 recover
$$\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_2 = \left[\begin{array}{c:c} \mathbf{A}_1^{\mathrm{T}}\begin{bmatrix} b_3 \\ b_7 \end{bmatrix} & \mathbf{A}_1^{\mathrm{T}}\begin{bmatrix} b_4 \\ b_8 \end{bmatrix} \end{array}\right],$$
$$\mathbf{C}_1^{\mathrm{T}}\mathbf{D}_2 = \left[\begin{array}{c:c} \mathbf{C}_1^{\mathrm{T}}\begin{bmatrix} d_3 \\ d_7 \end{bmatrix} & \mathbf{C}_1^{\mathrm{T}}\begin{bmatrix} d_4 \\ d_8 \end{bmatrix} \end{array}\right],$$
respectively. Since user 1 has recovered $\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1$ in the first step and cached $\mathbf{B}_1^{-1}\begin{bmatrix} b_3 \\ b_7 \end{bmatrix}$, it can recover $\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1\mathbf{B}_1^{-1}\begin{bmatrix} b_3 \\ b_7 \end{bmatrix} = \mathbf{A}_1^{\mathrm{T}}\begin{bmatrix} b_3 \\ b_7 \end{bmatrix}$. Similarly, user 2 can recover $\mathbf{C}_1^{\mathrm{T}}\begin{bmatrix} d_3 \\ d_7 \end{bmatrix}$. In addition, $\mathbf{A}_1^{\mathrm{T}}\begin{bmatrix} b_4 \\ b_8 \end{bmatrix} = \mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1\mathbf{B}_1^{-1}\begin{bmatrix} b_4 \\ b_8 \end{bmatrix}$, where $F'_{1,\{2\}} = \mathbf{B}_1^{-1}\begin{bmatrix} b_4 \\ b_8 \end{bmatrix}$ is requested by user 1 and cached by user 2. Similarly, $F'_{2,\{1\}} = \mathbf{D}_1^{-1}\begin{bmatrix} d_3 \\ d_7 \end{bmatrix}$ is requested by user 2 and cached by user 1. We let the server transmit $F'_{1,\{2\}} + F'_{2,\{1\}}$ for a total of 2 symbols.
- In the third step, we let user 1 recover
$$\mathbf{A}_2^{\mathrm{T}}\mathbf{B}_1 = \left(\mathbf{A}_1^{-1}\mathbf{A}_2\right)^{\mathrm{T}}\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1,$$
$$\mathbf{A}_2^{\mathrm{T}}\mathbf{B}_2 = \left(\mathbf{A}_1^{-1}\mathbf{A}_2\right)^{\mathrm{T}}\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_2,$$
and let user 2 recover
$$\mathbf{C}_2^{\mathrm{T}}\mathbf{D}_1 = \left(\mathbf{C}_1^{-1}\mathbf{C}_2\right)^{\mathrm{T}}\mathbf{C}_1^{\mathrm{T}}\mathbf{D}_1,$$
$$\mathbf{C}_2^{\mathrm{T}}\mathbf{D}_2 = \left(\mathbf{C}_1^{-1}\mathbf{C}_2\right)^{\mathrm{T}}\mathbf{C}_1^{\mathrm{T}}\mathbf{D}_2.$$
Note that $\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_1$ and $\mathbf{A}_1^{\mathrm{T}}\mathbf{B}_2$ have been recovered by user 1; in addition, we have
$$\mathbf{A}_1^{-1}\mathbf{A}_2 = \left[\begin{array}{c:c} \mathbf{A}_1^{-1}\begin{bmatrix} a_3 \\ a_7 \end{bmatrix} & \mathbf{A}_1^{-1}\begin{bmatrix} a_4 \\ a_8 \end{bmatrix} \end{array}\right],$$
where $\mathbf{A}_1^{-1}\begin{bmatrix} a_3 \\ a_7 \end{bmatrix}$ is cached by user 1, and $F''_{1,\{2\}} = \mathbf{A}_1^{-1}\begin{bmatrix} a_4 \\ a_8 \end{bmatrix}$ is requested by user 1 and cached by user 2. Similarly, user 2 only needs to recover $F''_{2,\{1\}} = \mathbf{C}_1^{-1}\begin{bmatrix} c_3 \\ c_7 \end{bmatrix}$, which is cached by user 1. We let the server transmit $F''_{1,\{2\}} + F''_{2,\{1\}}$ for a total of 2 symbols.

Thus, the server transmits $5+2+2 = 9$ symbols in total. Had we directly used the column-partition scheme for the case $\mathsf{a} \leq 1$, the server would have sent 5 symbols for each block, for a total of 20 symbols. □

We are now ready to generalize Example 4.

$$\mathbf{W}_{d_{k,1}}^{\mathrm{T}}\mathbf{W}_{d_{k,2}} = \left[\begin{array}{c:c:c} \mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(1)}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(1)} & \cdots & \mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(1)}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}\left(\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right)} \\ \hdashline \vdots & \ddots & \vdots \\ \hdashline \mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}\left(\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right)}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(1)} & \cdots & \mathbf{W}_{d_{k,1},\mathcal{N}_{\mathsf{K}}\left(\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right)}^{\mathrm{T}}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}\left(\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right)} \end{array}\right]. \tag{49}$$

Placement phase: We partition each matrix $\mathbf{W}_i$ where $i \in [\mathsf{N}]$ into two blocks

$$(\mathbf{W}_i)_{\mathsf{s}\times\mathsf{r}} = \left[\begin{array}{c:c} (\mathbf{W}_{i,1})_{\mathsf{s}\times\mathsf{s}} & (\mathbf{W}_{i,2})_{\mathsf{s}\times(\mathsf{r}-\mathsf{s})} \end{array}\right].$$

Up to a column permutation, the rank of $\mathbf{W}_{i,1}$ is equal to the rank of $\mathbf{W}_i$.[6]

The cache placement for $\mathbf{W}_{i,1}$ is the same as the case where $\mathsf{a} < 1$. Recall that $t_\mathsf{K} = \lfloor \frac{\mathsf{KM}}{\mathsf{N}} \rfloor$ and $\alpha_\mathsf{K} = t_\mathsf{K} + 1 - \frac{\mathsf{KM}}{\mathsf{N}}$. The first $\alpha_\mathsf{K}\mathsf{s}$ columns of $\mathbf{W}_{i,1}$ are partitioned into $\binom{\mathsf{K}}{t_\mathsf{K}}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,1,\mathcal{T}_1}$ and cached by users in $\mathcal{T}_1$, where $\mathcal{T}_1 \subseteq [\mathsf{K}]$ and $|\mathcal{T}_1| = t_\mathsf{K}$. The remaining $(1-\alpha_\mathsf{K})\mathsf{s}$ columns of $\mathbf{W}_{i,1}$ are partitioned into $\binom{\mathsf{K}}{t_\mathsf{K}+1}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,1,\mathcal{T}_2}$ and cached by users in $\mathcal{T}_2$, where $\mathcal{T}_2 \subseteq [\mathsf{K}]$ and $|\mathcal{T}_2| = t_\mathsf{K} + 1$.

The cache placement for $\mathbf{W}_{i,2}$ is as follows.

- We partition the first $\alpha_\mathsf{K}(\mathsf{r}-\mathsf{s})$ columns of $\mathbf{W}_{i,2}$ into $\binom{\mathsf{K}}{t_\mathsf{K}}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,2,\mathcal{T}_1}$, where $\mathcal{T}_1 \subseteq [\mathsf{K}]$ and $|\mathcal{T}_1| = t_\mathsf{K}$. $\mathbf{W}_{i,2,\mathcal{T}_1}$ has dimension $\mathsf{s} \times \frac{\alpha_\mathsf{K}(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_\mathsf{K}}}$. We let each user in $\mathcal{T}_1$ cache $\mathbf{Q}(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1})$, where
$$\mathbf{W}_{i,1}\mathbf{Q}(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1}) = \mathbf{W}_{i,2,\mathcal{T}_1},$$
and the dimension of $\mathbf{Q}(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1})$ is the same as $\mathbf{W}_{i,2,\mathcal{T}_1}$. More precisely, since the rank of $\mathbf{W}_{i,1}$ is equal to the rank of $\mathbf{W}_i$, each column of $\mathbf{W}_{i,2,\mathcal{T}_1}$ can be expressed by a linear combination of the columns of $\mathbf{W}_{i,1}$. For example, the j^{th} column of $\mathbf{W}_{i,2,\mathcal{T}_1}$ is equal to
$$\mathbf{W}_{i,1}\mathbf{Q}_j(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1}),$$
where $\mathbf{Q}_j(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1})$ represents the j^{th} column of $\mathbf{Q}(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1})$. Note that if $\mathbf{W}_{i,1}$ is full-rank, $\mathbf{Q}(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}_1})$ becomes $\mathbf{W}_{i,1}^{-1}\mathbf{W}_{i,2,\mathcal{T}_1}$.
- Similarly, the remaining $(1-\alpha_\mathsf{K})(\mathsf{r}-\mathsf{s})$ columns of $\mathbf{W}_{i,2}$ are partitioned into $\binom{\mathsf{K}}{t_\mathsf{K}+1}$ sub-matrices, each of which is denoted by $\mathbf{W}_{i,2,\mathcal{T}_2}$, where $\mathcal{T}_2 \subseteq [\mathsf{K}]$ and $|\mathcal{T}_2| = t_\mathsf{K} + 1$. $\mathbf{W}_{i,2,\mathcal{T}_2}$ has dimension $\mathsf{s} \times \frac{(1-\alpha_\mathsf{K})(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_\mathsf{K}+1}}$. We let each user in $\mathcal{T}_2$ cache $\mathbf{Q}(\mathbf{W}_{i,1}\mathbf{W}_{i,2,\mathcal{T}_2})$.

Since the dimension of $\mathbf{Q}(\mathbf{W}_{i,1}, \mathbf{W}_{i,2,\mathcal{T}})$ is the same as $\mathbf{W}_{i,2,\mathcal{T}}$ for any $\mathcal{T} \subseteq [\mathsf{K}]$ where $|\mathcal{T}| \in \{t_\mathsf{K}, t_\mathsf{K}+1\}$, the total number of symbols cached by each user is the same as for the case where $\mathsf{a} \le 1$ (which is Msr). Hence the cache size constraint is satisfied.

Delivery phase: The matrix product desired by user $k \in [\mathsf{K}]$ can be expressed as

$$(\mathbf{W}^{\mathrm{T}}_{d_{k,1}}\mathbf{W}_{d_{k,2}})_{\mathsf{r}\times\mathsf{r}} = \left[\begin{array}{c} (\mathbf{W}^{\mathrm{T}}_{d_{k,1},1})_{\mathsf{s}\times\mathsf{s}} \\ \hdashline (\mathbf{W}^{\mathrm{T}}_{d_{k,1},2})_{(\mathsf{r}-\mathsf{s})\times\mathsf{s}} \end{array}\right] \left[\begin{array}{c:c} (\mathbf{W}_{d_{k,2},1})_{\mathsf{s}\times\mathsf{s}} & (\mathbf{W}_{d_{k,2},2})_{\mathsf{s}\times(\mathsf{r}-\mathsf{s})} \end{array}\right] \tag{54a}$$

$$= \left[\begin{array}{c:c} (\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1})_{\mathsf{s}\times\mathsf{s}} & (\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2})_{\mathsf{s}\times(\mathsf{r}-\mathsf{s})} \\ \hdashline (\mathbf{W}^{\mathrm{T}}_{d_{k,1},2}\mathbf{W}_{d_{k,2},1})_{(\mathsf{r}-\mathsf{s})\times\mathsf{s}} & (\mathbf{W}^{\mathrm{T}}_{d_{k,1},2}\mathbf{W}_{d_{k,2},2})_{(\mathsf{r}-\mathsf{s})\times(\mathsf{r}-\mathsf{s})} \end{array}\right]. \tag{54b}$$

In the following, we divide the transmission into three steps.

First step: we deliver packets for $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1}$, for all $k \in [\mathsf{K}]$. The transmission for $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1}$ is the same as the proposed column-partition scheme with $\mathsf{a} = 1$ as described earlier in this subsection. Thus with the same derivation that led to (52), the total number of symbols transmitted in the first step is

$$\sum_{i\in[0:t_\mathsf{K}+1]} \binom{\mathsf{K}}{i+1} f_{i,1} = y\mathsf{s}^2, \tag{55}$$

where y is defined in (16b).

Second step: we then focus on $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2}$. We partition $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2}$ into $\binom{\mathsf{K}}{t_\mathsf{K}} + \binom{\mathsf{K}}{t_\mathsf{K}+1} = \binom{\mathsf{K}+1}{t_\mathsf{K}+1}$ sub-matrices as

$$\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2} = \left[\begin{array}{c:c:c} \mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2,\mathcal{N}_\mathsf{K}(1)} & \cdots & \mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2,\mathcal{N}_\mathsf{K}\left(\binom{\mathsf{K}+1}{t_\mathsf{K}+1}\right)} \end{array}\right].$$

For each $j \in \left[\binom{\mathsf{K}+1}{t_\mathsf{K}+1}\right]$, we have

$$\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2,\mathcal{N}_\mathsf{K}(j)} = \mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1}\mathbf{Q}\left(\mathbf{W}_{d_{k,2},1}, \mathbf{W}_{d_{k,2},2,\mathcal{N}_\mathsf{K}(j)}\ \right). \tag{56}$$

Note that $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1}$ has been recovered by user k in the first delivery step. Hence, in this step user k needs to recover $\mathbf{Q}(\mathbf{W}_{d_{k,2},1}\mathbf{W}_{d_{k,2},2,\mathcal{N}_\mathsf{K}(j)}\ )$, which is cached by users in $\mathcal{N}_\mathsf{K}(j)$. We let

$$F'_{k,\mathcal{N}_\mathsf{K}(j)} = \mathbf{Q}(\mathbf{W}_{d_{k,2},1}, \mathbf{W}_{d_{k,2},2,\mathcal{N}_\mathsf{K}(j)}\ ),$$

which contains $\mathsf{s}\frac{\alpha_\mathsf{K}(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_\mathsf{K}}}$ symbols if $|\mathcal{N}_\mathsf{K}(j)| = t_\mathsf{K}$, and contains $\mathsf{s}\frac{(1-\alpha_\mathsf{K})(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_\mathsf{K}+1}}$ symbols if $|\mathcal{N}_\mathsf{K}(j)| = t_\mathsf{K}+1$.

For each set $\mathcal{S}_1 \subseteq [\mathsf{K}]$ where $\mathcal{S}_1 = t_\mathsf{K}+1$, the server broadcasts

$$\sum_{j\in\mathcal{S}_1} F'_{k,\mathcal{S}_1\setminus\{k\}}. \tag{57}$$

For each set $\mathcal{S}_2 \subseteq [\mathsf{K}]$ where $\mathcal{S}_2 = t_\mathsf{K}+2$, the server broadcasts

$$\sum_{j\in\mathcal{S}_2} F'_{k,\mathcal{S}_2\setminus\{k\}}. \tag{58}$$

Hence, the total number of symbols transmitted in the second step is

$$\binom{\mathsf{K}}{t_\mathsf{K}+1}\mathsf{s}\frac{\alpha_\mathsf{K}(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_\mathsf{K}}} + \binom{\mathsf{K}}{t_\mathsf{K}+2}\mathsf{s}\frac{(1-\alpha_\mathsf{K})(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_\mathsf{K}+1}}. \tag{59}$$

Third step: we let each user $k \in [\mathsf{K}]$ recover the remaining parts of its desired matrix product, shown in (60b) shown at the bottom of the next page. Note that $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1}$ and $\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2}$ have been recovered by user k in the first and second steps, respectively. Now it only needs to recover $\mathbf{Q}(\mathbf{W}_{d_{k,1},1}\mathbf{W}_{d_{k,1},2})$, which can be expressed as

$$\mathbf{Q}(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2}) = \Bigg[\mathbf{Q}\left(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2,\mathcal{N}_\mathsf{K}(1)}\right), \ldots, \mathbf{Q}\left(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2,\mathcal{N}_\mathsf{K}\left(\binom{\mathsf{K}+1}{t_\mathsf{K}+1}\right)}\right)\Bigg].$$

[6] The information of permutation is also cached by each user, which is negligible compared to the field size q and the cache size of each user.

For each $j \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$, we let

$$F''_{k,\mathcal{N}_{\mathsf{K}}(j)} = \mathbf{Q}\left(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2,\mathcal{N}_{\mathsf{K}}(j)}\ \right),$$

which contains $\mathsf{s}\frac{\alpha_{\mathsf{K}}(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}$ symbols if $|\mathcal{N}_{\mathsf{K}}(j)| = t_{\mathsf{K}}$, and contains $\mathsf{s}\frac{(1-\alpha_{\mathsf{K}})(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}$ symbols if $|\mathcal{N}_{\mathsf{K}}(j)| = t_{\mathsf{K}} + 1$.

For each set $\mathcal{S}_1 \subseteq [\mathsf{K}]$ where $\mathcal{S}_1 = t_{\mathsf{K}} + 1$, the server broadcasts

$$\sum_{j \in \mathcal{S}_1} F''_{k,\mathcal{S}_1 \setminus \{k\}}. \tag{61}$$

For each set $\mathcal{S}_2 \subseteq [\mathsf{K}]$ where $\mathcal{S}_2 = t_{\mathsf{K}}+2$, the server broadcasts

$$\sum_{j \in \mathcal{S}_2} F''_{k,\mathcal{S}_2 \setminus \{k\}}. \tag{62}$$

Hence, the total number of symbols transmitted in the third step is

$$\binom{\mathsf{K}}{t_{\mathsf{K}}+1}\mathsf{s}\frac{\alpha_{\mathsf{K}}(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_{\mathsf{K}}}} + \binom{\mathsf{K}}{t_{\mathsf{K}}+2}\mathsf{s}\frac{(1-\alpha_{\mathsf{K}})(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}. \tag{63}$$

Considering all the three steps, from (55), (59), and (63), the total load is

$$\begin{aligned} &\frac{y\mathsf{s}^2 + 2\binom{\mathsf{K}}{t_{\mathsf{K}}+1}\mathsf{s}\frac{\alpha_{\mathsf{K}}(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_{\mathsf{K}}}} + 2\binom{\mathsf{K}}{t_{\mathsf{K}}+2}\mathsf{s}\frac{(1-\alpha_{\mathsf{K}})(\mathsf{r}-\mathsf{s})}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}}{f(\mathsf{r},\mathsf{s},\mathsf{r})} \\ &= \frac{y + 2\binom{\mathsf{K}}{t_{\mathsf{K}}+1}\frac{\alpha_{\mathsf{K}}(\mathsf{a}-1)}{\binom{\mathsf{K}}{t_{\mathsf{K}}}} + 2\binom{\mathsf{K}}{t_{\mathsf{K}}+2}\frac{(1-\alpha_{\mathsf{K}})(\mathsf{a}-1)}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}}{2\mathsf{a}-1}, \end{aligned} \tag{64}$$

where (64) follows from that $\mathsf{a} > 1$. From (64), we prove (16a) for the case where $\mathsf{a} > 1$.

Remark 9 (Column-Partition Scheme v.s. Uncoded Caching Baseline Scheme): When $\mathsf{a} \leq 1$, for any pair (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$ and $k \notin \mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2)$, if the server directly broadcasts $\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(j_1)}\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(j_2)}$, then our column-partition scheme reduces to the uncoded caching baseline scheme for Theorem 2. Hence, in this case our column-partition scheme is strictly better than the uncoded caching baseline scheme if $0 < \mathsf{M} < \mathsf{N}$. When $\mathsf{a} > 1$, besides the above improvement which also appears in the first delivery step, in the second and third steps we further compress the desired matrix products of the users by leveraging the correlation among the elements in the product and the users' caches. Hence, in this case our column-partition scheme is strictly better than the uncoded caching baseline scheme if $\mathsf{M} < \mathsf{N}$.

□

VI. Conclusion

This paper introduced a novel coded caching problem for matrix multiplication retrieval, where each cache-aided user requests the product of two matrices in the library. We first proposed a structure-agnostic scheme which treats each product as an independent file. In order to leverage the structure of matrix multiplication, we proposed two schemes (by row-partition and column-partition, respectively) to attain coded caching gain for the matrix multiplication retrieval problem, by leveraging the correlation among the elements in each product. The proposed schemes outperform the baseline schemes. For "fat" matrices, the proposed row-partition scheme is proved to be order optimal within a factor of 2 under the constraint of uncoded cache placement and $\mathsf{N} \geq 2\mathsf{K}$.

Appendix A
Structure-Agnostic Scheme: Proof of Theorem 1

For each pair (i, j) where $1 \leq i \leq j \leq \mathsf{N}$, we define

$$W_{(i,j)} := P\left(\mathbf{W}_i^{\mathrm{T}}, \mathbf{W}_j\right) \tag{65}$$

and treat $W_{(i,j)}$ as an independent file with B symbols, where we can recover $\mathbf{W}_i^{\mathrm{T}}\mathbf{W}_j$ from $W_{(i,j)}$. We then use the MAN coded caching scheme as follows.

Placement phase: We focus on each $t \in [0 : \mathsf{K}]$. For each pair (i, j) where $1 \leq i \leq j \leq \mathsf{N}$, we divide $W_{(i,j)}$ into $\binom{\mathsf{K}}{t}$ non-overlapping and equal-length subfiles, $W_{(i,j)} = \{W_{(i,j),\mathcal{T}} : \mathcal{T} \subseteq [\mathsf{K}], |\mathcal{T}| = t\}$, where each subfile $W_{(i,j),\mathcal{T}}$ contains $\frac{\mathsf{B}}{\binom{\mathsf{K}}{t}}$ symbols and is cached by users in $\mathcal{T}$. As there are $\binom{\mathsf{N}}{2} + \mathsf{N} = \frac{\mathsf{N}(\mathsf{N}+1)}{2}$ pairs (i, j) where $1 \leq i \leq j \leq \mathsf{N}$, the total number of symbols cached by each user is

$$\frac{\mathsf{N}(\mathsf{N}+1)}{2}\frac{\mathsf{B}\binom{\mathsf{K}-1}{t-1}}{\binom{\mathsf{K}}{t}} = \frac{\mathsf{N}(\mathsf{N}+1)\mathsf{B}t}{2\mathsf{K}} = \frac{\mathsf{N}(\mathsf{N}+1)\mathsf{B}t}{2\mathsf{a}\mathsf{K}\mathsf{s}^2}\mathsf{sr} = \mathsf{Msr},$$

satisfying the cache size constraint.

Delivery phase: Each user $k \in [\mathsf{K}]$ demands $W_{\mathbf{d}_k}$. For each set $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| = t + 1$, the server transmits

$$\sum_{k \in \mathcal{S}} W_{\mathbf{d}_k,\mathcal{S}\setminus\{k\}}, \tag{66}$$

where each user $k \in \mathcal{S}$ caches all subfiles except $W_{\mathbf{d}_k,\mathcal{S}\setminus\{k\}}$ such that it can recover $W_{\mathbf{d}_k,\mathcal{S}\setminus\{k\}}$.

After considering all sets of users with cardinality $t + 1$, each user can recover its demanded file and thus recover its demanded product. Hence, the total load is

$$\frac{\binom{\mathsf{K}}{t+1}}{\binom{\mathsf{K}}{t}} = \frac{\mathsf{K} - t}{t + 1},$$

which coincides with (12).

$$\begin{aligned} &\left[\ \mathbf{W}^{\mathrm{T}}_{d_{k,1},2}\mathbf{W}_{d_{k,2},1} \ \middle|\ \mathbf{W}^{\mathrm{T}}_{d_{k,1},2}\mathbf{W}_{d_{k,2},2}\ \right] \\ &= \left[\ \left(\mathbf{W}_{d_{k,1},1}\mathbf{Q}(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2})\right)^{\mathrm{T}}\mathbf{W}_{d_{k,2},1} \ \middle|\ \left(\mathbf{W}_{d_{k,1},1}\mathbf{Q}(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2})\right)^{\mathrm{T}}\mathbf{W}_{d_{k,2},2}\ \right] &\text{(60a)} \\ &= \left[\ \left(\mathbf{Q}(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2})\right)^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},1} \ \middle|\ \left(\mathbf{Q}(\mathbf{W}_{d_{k,1},1}, \mathbf{W}_{d_{k,1},2})\right)^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}_{d_{k,1},1}\mathbf{W}_{d_{k,2},2}\ \right]. &\text{(60b)} \end{aligned}$$

APPENDIX B
PROOF OF THEOREM 6

We consider the case where $\mathsf{a} \geq 1$ and $\mathsf{N} \geq 2\mathsf{K}$.

A. Converse

We consider the worse-case demands, where $[\mathbf{d}_1; \ldots; \mathbf{d}_{\mathsf{K}}]$ contains $2\mathsf{K}$ different indices of matrices.

We use a genie-aided converse bound. We assume that during the delivery phase there is a private link from the server to each user $k \in [\mathsf{K}]$ through which the server transmits $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}$ to user k. In this case, the minimum worst-case number of broadcasted symbols by the server under uncoded cache placement is denoted by $\mathsf{L}^{\star}_{\text{genie, u}}$. Obviously, we have

$$\mathsf{R}^{\star}_{\mathrm{u}} f(\mathsf{r}, \mathsf{s}, \mathsf{r}) \geq \mathsf{L}^{\star}_{\text{genie, u}}. \tag{67}$$

Recall that $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}$ is of dimension $\mathsf{r} \times \mathsf{s}$ where $\mathsf{r} \geq \mathsf{s}$ and its elements are uniformly i.i.d. Hence, if user k can recover $\mathbf{W}^{\mathrm{T}}_{d_{k,1}} \mathbf{W}_{d_{k,2}}$, with the knowledge of $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}$ this user can also recover $\mathbf{W}_{d_{k,2}}$. On the other hand, if user k can recover $\mathbf{W}_{d_{k,2}}$, with the knowledge of $\mathbf{W}^{\mathrm{T}}_{d_{k,1}}$ this user can also recover $\mathbf{W}^{\mathrm{T}}_{d_{k,1}} \mathbf{W}_{d_{k,2}}$. Hence, when $\mathsf{a} \geq 1$ we have

$$H(\mathbf{W}^{\mathrm{T}}_{d_{k,1}} \mathbf{W}_{d_{k,2}} | \mathbf{W}^{\mathrm{T}}_{d_{k,1}}) = H(\mathbf{W}_{d_{k,2}} | \mathbf{W}^{\mathrm{T}}_{d_{k,1}}) = H(\mathbf{W}_{d_{k,2}}).$$

Under the constraint of uncoded cache placement, it is equivalent to the problem with the same network but each user aims to retrieve a whole file (each file has sr symbols).

In addition, since the cache placement is uncoded and $[\mathbf{d}_1; \ldots; \mathbf{d}_{\mathsf{K}}]$ contains $2\mathsf{K}$ different indices of matrices, the matrix transmitted through the private link cannot help each user $k \in [\mathsf{K}]$ to decode its desired file (i.e., $\mathbf{W}_{d_{k,2}}$). Thus we can use the converse bound in [27], [31] for the original MAN coded caching problem for single file retrieval to lower bound $\mathsf{L}^{\star}_{\text{genie, u}}$. In other words, $(\mathsf{M}, \mathsf{L}^{\star}_{\text{genie, u}})$ is lower bounded by the lower convex envelop of $\left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{\mathsf{K}-t}{t+1}\mathsf{sr}/\mathsf{B}\right)$, for all $t \in [0 : \mathsf{K}]$. In conclusion, from (67), $(\mathsf{M}, \mathsf{R}^{\star}_{\mathrm{u}})$ is lower bounded by the lower convex envelop of

$$\left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{\mathsf{K}-t}{t+1} \frac{\mathsf{sr}}{f(\mathsf{r}, \mathsf{s}, \mathsf{r})}\right) = \left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{\mathsf{K}-t}{t+1} \frac{\mathsf{a}}{2\mathsf{a}-1}\right), \ \forall t \in [0 : \mathsf{K}]. \tag{68}$$

B. Achievability

From (14), the multi-request baseline scheme can achieve the lower convex envelop of $(\mathsf{M}, \mathsf{R}_2) = \left(\frac{\mathsf{N}t}{\mathsf{K}}, \frac{2(\mathsf{K}-t)\mathsf{a}}{(t+1)g(\mathsf{a}, \mathsf{a})}\right)$, for all $t \in [0 : \mathsf{K}]$. Compared with the converse bound in (68), the multi-request baseline scheme is order optimal within a factor of 2 under the constraint of uncoded cache placement and $\mathsf{a} \geq 1$. In addition, from Corollary 1, the proposed row-partition scheme outperforms the multi-request baseline scheme. Hence, we prove Theorem 6.

APPENDIX C
PROOF OF LEMMA 1

We fix one $k \in [\mathsf{K}]$ and one $i \in [0 : t_{\mathsf{K}} + 1]$. Now we want to compute the length of $F_{k,\mathcal{V}}$ where $\mathcal{V} \subseteq ([\mathsf{K}] \setminus \{k\})$ and $|\mathcal{V}| = i$. If one pair (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$ satisfies that, $\mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2) = \mathcal{V}$, we have that $F_{k,\mathcal{V}}$ contains $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1}, \mathcal{N}_{\mathsf{K}}(j_1)}, \mathbf{W}_{d_{k,2}, \mathcal{N}_{\mathsf{K}}(j_2)}\right)$.

Now we divide all the pairs (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$ and $\mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2) = \mathcal{V}$ into the following four cases.

Case 1: $|\mathcal{N}_{\mathsf{K}}(j_1)| = |\mathcal{N}_{\mathsf{K}}(j_2)| = t_{\mathsf{K}}$: In this case, the length of $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1}, \mathcal{N}_{\mathsf{K}}(j_1)}, \mathbf{W}_{d_{k,2}, \mathcal{N}_{\mathsf{K}}(j_2)}\right)$ is

$$f\left(\frac{\alpha_{\mathsf{K}}\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}, \mathsf{s}, \frac{\alpha_{\mathsf{K}}\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}\right) = g\left(\frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}, \frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}\right) \mathsf{s}^2 \tag{69a}$$

$$= \left(\frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}\right)^2 \mathsf{s}^2, \tag{69b}$$

where (69b) comes from that $\mathsf{a} \leq 1$ and thus $\frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}} \leq 1$. The number of pairs (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$, $\mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2) = \mathcal{V}$, and $|\mathcal{N}_{\mathsf{K}}(j_1)| = |\mathcal{N}_{\mathsf{K}}(j_2)| = t_{\mathsf{K}}$ is

$$\binom{\mathsf{K} - |\mathcal{V}|}{t_{\mathsf{K}} - |\mathcal{V}|} \binom{\mathsf{K} - t_{\mathsf{K}}}{t_{\mathsf{K}} - |\mathcal{V}|} = \binom{\mathsf{K} - i}{t_{\mathsf{K}} - i} \binom{\mathsf{K} - t_{\mathsf{K}}}{t_{\mathsf{K}} - i}. \tag{70}$$

Case 2: $|\mathcal{N}_{\mathsf{K}}(j_1)| = |\mathcal{N}_{\mathsf{K}}(j_2)| = t_{\mathsf{K}} + 1$: In this case, the length of $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1}, \mathcal{N}_{\mathsf{K}}(j_1)}, \mathbf{W}_{d_{k,2}, \mathcal{N}_{\mathsf{K}}(j_2)}\right)$ is

$$f\left(\frac{(1-\alpha_{\mathsf{K}})\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}, \mathsf{s}, \frac{(1-\alpha_{\mathsf{K}})\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right)$$

$$= g\left(\frac{(1-\alpha_{\mathsf{K}})\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}, \frac{(1-\alpha_{\mathsf{K}})\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right) \mathsf{s}^2 \tag{71a}$$

$$= \left(\frac{(1-\alpha_{\mathsf{K}})\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right)^2 \mathsf{s}^2. \tag{71b}$$

The number of pairs (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$, $\mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2) = \mathcal{V}$, and $|\mathcal{N}_{\mathsf{K}}(j_1)| = |\mathcal{N}_{\mathsf{K}}(j_2)| = t_{\mathsf{K}} + 1$ is

$$\binom{\mathsf{K} - |\mathcal{V}|}{t_{\mathsf{K}} + 1 - |\mathcal{V}|} \binom{\mathsf{K} - t_{\mathsf{K}} - 1}{t_{\mathsf{K}} + 1 - |\mathcal{V}|} = \binom{\mathsf{K} - i}{t_{\mathsf{K}} + 1 - i} \binom{\mathsf{K} - t_{\mathsf{K}} - 1}{t_{\mathsf{K}} + 1 - i}. \tag{72}$$

Case 3: $|\mathcal{N}_{\mathsf{K}}(j_1)| = t_{\mathsf{K}}$ and $|\mathcal{N}_{\mathsf{K}}(j_2)| = t_{\mathsf{K}} + 1$: In this case, the length of $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1}, \mathcal{N}_{\mathsf{K}}(j_1)}, \mathbf{W}_{d_{k,2}, \mathcal{N}_{\mathsf{K}}(j_2)}\right)$ is

$$f\left(\frac{\alpha_{\mathsf{K}}\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}, \mathsf{s}, \frac{(1-\alpha_{\mathsf{K}})\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right) = g\left(\frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}, \frac{(1-\alpha_{\mathsf{K}})\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\right) \mathsf{s}^2 \tag{73a}$$

$$= \frac{\alpha_{\mathsf{K}}(1-\alpha_{\mathsf{K}})\mathsf{a}^2}{\binom{\mathsf{K}}{t_{\mathsf{K}}}\binom{\mathsf{K}}{t_{\mathsf{K}}+1}} \mathsf{s}^2. \tag{73b}$$

The number of pairs (j_1, j_2) where $j_1, j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$, $\mathcal{N}_{\mathsf{K}}(j_1) \cap \mathcal{N}_{\mathsf{K}}(j_2) = \mathcal{V}$, $|\mathcal{N}_{\mathsf{K}}(j_1)| = t_{\mathsf{K}}$, and $|\mathcal{N}_{\mathsf{K}}(j_2)| =$

$t_{\mathsf{K}}+1$ is

$$\binom{\mathsf{K}-|\mathcal{V}|}{t_{\mathsf{K}}-|\mathcal{V}|}\binom{\mathsf{K}-t_{\mathsf{K}}}{t_{\mathsf{K}}+1-|\mathcal{V}|}=\binom{\mathsf{K}-i}{t_{\mathsf{K}}-i}\binom{\mathsf{K}-t_{\mathsf{K}}}{t_{\mathsf{K}}+1-i}. \tag{74}$$

Case 4: $|\mathcal{N}_{\mathsf{K}}(j_{1)}| = t_{\mathsf{K}}+1$ and $|\mathcal{N}_{\mathsf{K}}(j_{2)}| = t_{\mathsf{K}}$: In this case, the length of $P\left(\mathbf{W}^{\mathrm{T}}_{d_{k,1},\mathcal{N}_{\mathsf{K}}(j_1)},\mathbf{W}_{d_{k,2},\mathcal{N}_{\mathsf{K}}(j_2)}\right)$ is

$$f\left(\frac{(1-\alpha_{\mathsf{K}})\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}},\mathsf{s},\frac{\alpha_{\mathsf{K}}\mathsf{r}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}\right)=g\left(\frac{(1-\alpha_{\mathsf{K}})\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}+1}},\frac{\alpha_{\mathsf{K}}\mathsf{a}}{\binom{\mathsf{K}}{t_{\mathsf{K}}}}\right)\mathsf{s}^2 \tag{75a}$$

$$=\frac{\alpha_{\mathsf{K}}(1-\alpha_{\mathsf{K}})\mathsf{a}^2}{\binom{\mathsf{K}}{t_{\mathsf{K}}}\binom{\mathsf{K}}{t_{\mathsf{K}}+1}}\mathsf{s}^2. \tag{75b}$$

The number of pairs (j_1,j_2) where $j_1,j_2 \in \left[\binom{\mathsf{K}+1}{t_{\mathsf{K}}+1}\right]$, $\mathcal{N}_{\mathsf{K}}(j_1)\cap\mathcal{N}_{\mathsf{K}}(j_2)=\mathcal{V}$, $|\mathcal{N}_{\mathsf{K}}(j_1)|=t_{\mathsf{K}}+1$, and $|\mathcal{N}_{\mathsf{K}}(j_2)|=t_{\mathsf{K}}$ is

$$\binom{\mathsf{K}-|\mathcal{V}|}{t_{\mathsf{K}}+1-|\mathcal{V}|}\binom{\mathsf{K}-t_{\mathsf{K}}-1}{t_{\mathsf{K}}-|\mathcal{V}|}$$

$$=\binom{\mathsf{K}-i}{t_{\mathsf{K}}+1-i}\binom{\mathsf{K}-t_{\mathsf{K}}-1}{t_{\mathsf{K}}-i} \tag{76a}$$

$$=\binom{\mathsf{K}-i}{t_{\mathsf{K}}-i}\binom{\mathsf{K}-t_{\mathsf{K}}}{t_{\mathsf{K}}+1-i}. \tag{76b}$$

Considering all the above four cases, we can prove Lemma 1.

References

[1] (2014). Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018*. White Paper. [Online]. Available: http://goo.gl/l77HAJ

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[3] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[4] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3212–3229, Jun. 2016.

[5] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 7253–7271, Dec. 2016.

[6] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3092–3107, May 2017.

[7] 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. 64, no. 1, pp. 109–128, Jan. 2018.

[8] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[9] M. Adel Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 7325–7349, Nov. 2019.

[10] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling for distributed machine learning," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 3098–3131, May 2020.

[11] K. Wan, D. Tuninetti, M. Ji, G. Caire, and P. Piantanida, "Fundamental limits of decentralized data shuffling," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3616–3637, Jun. 2020.

[12] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "On optimal load-memory tradeoff of cache-aided scalar linear function retrieval," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 4001–4018, Jun. 2021.

[13] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.

[14] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[15] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 4406–4416.

[16] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2722–2734, 2020.

[17] W.-T. Chang and R. Tandon, "On the upload versus download cost for secure and private matrix multiplication," 2019, *arXiv:1906.10684*.

[18] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2821–2846, May 2021.

[19] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the 'cubic' barrier," 2020, *arXiv:2001.05101*.

[20] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1215–1225.

[21] K. Li, M. Tao, J. Zhang, and O. Simeone, "Coded computing and cooperative transmission for wireless distributed matrix multiplication," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2224–2239, Apr. 2021.

[22] B. Kumar and A. Mahalanobis, *Correlation Pattern Recognition*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[23] G. B. Giannakis and M. K. Tsatsanis, "Signal detection and classification using matched filtering and higher order statistics," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 38, no. 7, pp. 1284–1296, Jul. 1990.

[24] M. Mukaka, "Statistics corner: A guide to appropriate use of correlation coefficient in medical research," *Malawi Med. J.*, vol. 24, no. 3, pp. 69–71, 2012.

[25] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 67, no. 11, pp. 7420–7437, Nov. 2021.

[26] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching and coded multicasting: Multiple groupcast index coding," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, 2014, pp. 881–885.

[27] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 1281–1296, Feb. 2018.

[28] K. Wan and G. Caire, "On coded caching with private demands," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 358–372, Jan. 2021.

[29] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "Fundamental limits of device-to-device private caching with a trusted server under uncoded cache placement and user collusion," 2019, *arXiv:1912.09985*.

[30] K. Wan, D. Tuninetti, M. Ji, and G. Caire, "On the fundamental limits of Fog-RAN cache-aided networks with downlink and sidelink communications," *IEEE Trans. Inf. Theory*, vol. 67, no. 4, pp. 2353–2378, Apr. 2021.

[31] K. Wan, D. Tuninetti, and P. Piantanida, "An index coding approach to caching with uncoded cache placement," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1318–1332, Mar. 2020.

[32] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite length analysis of caching-aided coded multicasting," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5524–5537, Oct. 2016.

Kai Wan (Member, IEEE) received the B.E. degree in optoelectronics from the Huazhong University of Science and Technology, China, in 2012, and the M.Sc. and Ph.D. degrees in communications from the Université Paris-Saclay, France, in 2014 and 2018, respectively. He is currently a Post-Doctoral Researcher with the Communications and Information Theory Chair (CommIT), Technische Universität Berlin, Berlin, Germany. His research interests include information theory, coding techniques, and their applications on coded caching, index coding, distributed storage, distributed computing, wireless communications, and privacy and security. He has been serving as an Associate Editor for the IEEE COMMUNICATIONS LETTERS since August 2021.

Hua Sun (Member, IEEE) received the B.E. degree in communications engineering from the Beijing University of Posts and Telecommunications, China, in 2011, and the M.S. degree in electrical and computer engineering and the Ph.D. degree in electrical engineering from the University of California at Irvine, Irvine, CA, USA, in 2013 and 2017, respectively.

He is currently an Assistant Professor with the Department of Electrical Engineering, University of North Texas, USA. His research interests include information theory and its applications to communications, privacy, security, and storage. He was a recipient of the NSF CAREER Award in 2021, and the UNT College of Engineering Distinguished Faculty Fellowship in 2021. His coauthored articles received the IEEE Jack Keil Wolf ISIT Student Paper Award in 2016, and the IEEE GLOBECOM Best Paper Award in 2016.

Mingyue Ji (Member, IEEE) received the B.E. degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2006, the M.Sc. degree in electrical engineering from the Royal Institute of Technology, Sweden, in 2008, the M.Sc. degree in electrical engineering from the University of California at Santa Cruz, Santa Cruz, CA, USA, in 2010, and the Ph.D. degree from the Ming Hsieh Department of Electrical Engineering, University of Southern California, in 2015. He subsequently was a Staff II System Design Scientist with Broadcom Corporation (Broadcom Ltd.) from 2015 to 2016. He is currently an Assistant Professor with the Electrical and Computer Engineering Department and an Adjunct Assistant Professor with the School of Computing, The University of Utah. His research interests include the broad area of information theory, coding theory, concentration of measure and statistics with the applications of caching networks, wireless communications, distributed storage and computing systems, distributed machine learning, and (statistical) signal processing. He received the NSF CAREER Award in 2022, the IEEE Communications Society Leonard G. Abraham Prize for the Best IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Paper in 2019, the Best Paper Awards at the 2021 IEEE GLOBECOM Conference and at the 2015 IEEE ICC Conference, the Best Student Paper Award at the 2010 IEEE European Wireless Conference and USC Annenberg Fellowship from 2010 to 2014. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS since 2020.

Daniela Tuninetti (Fellow, IEEE) received the Ph.D. degree in electrical engineering from ENST/Télécom ParisTech, Paris, France, with work done at the Eurecom Institute, Sophia Antipolis, France, in 2002. She was a Post-Doctoral Research Associate at the School of Communication and Computer Science, Swiss Federal Institute of Technology in Lausanne (EPFL, Lausanne, Switzerland), from 2002 to 2004. She is currently a Professor with the Department of Electrical and Computer Engineering, University of Illinois at Chicago (UIC), which she joined in 2005. Her research interests include ultimate performance limits of wireless interference networks (with special emphasis on cognition and user cooperation), coexistence between radar and communication systems, multi-relay networks, content-type coding, cache-aided systems, and distributed private coded computing. She was a recipient of the Best Paper Award at the European Wireless Conference in 2002, the NSF CAREER Award in 2007, and named as a University of Illinois Scholar in 2015. She was the Editor-in-Chief of the IEEE Information Theory Society Newsletter from 2006 to 2008, and an Editor of the IEEE COMMUNICATION LETTERS from 2006 to 2009, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2011 to 2014, and the IEEE TRANSACTIONS ON INFORMATION THEORY from 2014 to 2017. She is also an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS. She is also a Distinguished Lecturer with the Information Theory Society.

Giuseppe Caire (Fellow, IEEE) was born in Torino, in 1965. He received the B.Sc. degree in electrical engineering from the Politecnico di Torino in 1990, the M.Sc. degree in electrical engineering from Princeton University in 1992, and the Ph.D. degree from the Politecnico di Torino in 1994.

He has been a Post-Doctoral Research Fellow with the European Space Agency (ESTEC), Noordwijk, The Netherlands, from 1994 to 1995 an Assistant Professor of telecommunications at the Politecnico di Torino; an Associate Professor at the University of Parma, Italy; a Professor with the Department of Mobile Communications, Eurecom Institute, Sophia-Antipolis, France; and a Professor of electrical engineering with the Viterbi School of Engineering, University of Southern California, Los Angeles, CA, USA. He is currently an Alexander von Humboldt Professor with the Faculty of Electrical Engineering and Computer Science, Technical University of Berlin, Germany. His main research interests include the field of communications theory, information theory, and channel and source coding with particular focus on wireless communications. He received the Jack Neubauer Best System Paper Award from the IEEE Vehicular Technology Society in 2003, the IEEE Communications Society and Information Theory Society Joint Paper Award in 2004 and 2011, the Okawa Research Award in 2006, the Alexander von Humboldt Professorship in 2014, the Vodafone Innovation Prize in 2015, an ERC Advanced Grant in 2018, the Leonard G. Abraham Prize for the Best IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Paper in 2019, and the IEEE Communications Society Edwin Howard Armstrong Achievement Award in 2020. He was a recipient of the 2021 Leibinz Prize of the German National Science Foundation (DFG). He has served in the Board of Governors for the IEEE Information Theory Society from 2004 to 2007, and as an Officer from 2008 to 2013. He was the President of the IEEE Information Theory Society in 2011.