New optimal trade-off point for coded caching systems with limited cache size

Yinbin Ma and Daniela Tuninetti
University of Illinois Chicago, Chicago, IL 60607, USA
Email:{yma52, danielat}@uic.edu

Abstract

This paper presents a new achievable scheme for coded caching systems with N files, $\mathsf{K} = \mathsf{N}$ users, and cache size $\mathsf{M} = 1/(\mathsf{N}-1)$. The scheme employs linear coding during the cache placement phase, and a three-stage transmissions designed to eliminate interference in the delivery phase. The achievable load meets a known converse bound, which impose no constraint on the cache placement, and is thus optimal. This new result, together with known inner and outer bounds, shows optimality of linear coding placement for $\mathsf{M} \leq 1/(\mathsf{N}-1)$ when $\mathsf{K} = \mathsf{N} \geq 3$. Interestingly and surprisingly, the proposed scheme is relatively simple but requires operations on a finite field of size at least 3.

I. Introduction

Coded caching, as introduced by Maddah-Ali and Niesen in [1], offers an opportunity to balance local cache storage against network bandwidth. The system model consists of an error-free broadcast channel with K users, each with local cache of size M files, and a central server with N files. The server proactively pushes content into the local caches when the network is underutilized and without knowing what the users will demand. Subsequently, once the server has received the file requests from the users, it initiates the transmission of coded multicast messages. with the goal of reducing the worst-case network communication load R.

a) Past Work: Coded caching under uncoded placement has been extensively investigated in prior research: [1] introduced an achievable scheme with uncoded placement; [2] showed that the scheme in [1] is optimal under the constraint of uncoded placement when $\mathsf{N} \geq \mathsf{K}$; [3] characterized the exact memory-load tradeoff under uncoded placement for $\mathsf{N} < \mathsf{K}$; [4] showed that uncoded placement is optimal to within a factor of 2. Uncoded placement is however

suboptimal: [1] showed that linear coding placement is optimal $\mathsf{K} = \mathsf{N} = 2$ in $\mathsf{M} \in (0, 1)$; [5] extends the optimal tradeoff with linear coding placement in $\mathsf{M} \in [0, 1/\mathsf{K}]$ to any $\mathsf{N} \leq \mathsf{K}$; [6] proposed a general scheme with linear coding placement, which combines rank-metric codes and maximum distance-separable codes and interference elimination for decoding, that achieves a gain over uncoded placement when $\mathsf{N} \leq \mathsf{K}$ but it is not optimal. In [7], leveraging [8], we derived the exact memory-load tradeoff under linear coding placement for $\mathsf{K} = 3$ users; we showed the existence of a new optimal tradeoff point for $\mathsf{M} = 1/2$.

In summary, the ultimate performance limit under linear coding placement remains an area of significant practical interest, with many aspects yet to be fully understood.

b) Contributions: In this paper, we consider a coded caching system with as many users as files, $\mathsf{N} = \mathsf{K}$. A new achievable scheme, that uses linear coding in the placement phase and interference elimination in the delivery phase, is proposed for cache size $\mathsf{M} = 1/(\mathsf{N} - 1)$. Interestingly and surprisingly, the proposed scheme is relatively simple but requires operations on a finite field of at least size three. The proposed scheme is shown to meet with equality a converse bound derived in [9] without any restriction for the placement phase, thus the proposed scheme is actually optimal. This implies that, together with the optimal scheme for $\mathsf{M} = 1/\mathsf{N}$ in [5], this work extends the memory regime over which optimality is known to $\mathsf{M} \leq 1/(\mathsf{N}-1)$ when $\mathsf{N} = \mathsf{K}$.

c) Paper Organization: This paper is organized as follows. Section II introduce the coded caching problem and relevant known results. Section III presents our main result, with proofs and an example. Section IV concludes the paper.

d) Notation Convention: In the rest of the paper we use the following notation convention. Calligraphic symbols denote sets, bold lowercase symbols vectors, bold uppercase symbols matrices, and sans-serif symbols system parameters. For a matrix $\mathbf{M}$, we let $\mathbf{M}[\mathcal{Q}, \mathcal{P}]$ denote the submatrix obtained by selection the rows indexed by $\mathcal{Q}$ and columns indexed by $\mathcal{P}$. For an integer b, we let $[b] := \{1, \ldots, b\}$. For sets $\mathcal{S}$ and $\mathcal{Q}$, we let $\mathcal{S} \setminus \mathcal{Q} := \{k : k \in \mathcal{S}, k \notin \mathcal{Q}\}$.

II. System Model and Known Results

A. Problem Formulation

A (N, K) coded caching system consists of a server, K users, and N files. Each file has B symbols, which are uniformly and independently distributed over $\mathbb{F}_{\mathsf{q}}$, where q is a prime-power number. Files are denoted as $F_n \in \mathbb{F}_{\mathsf{q}}^{\mathsf{B}}, n \in [\mathsf{N}]$. An error-free shared link connects the users to

the server. Each user has a local cache that can store no more than MB symbols in $\mathbb{F}_{\mathsf{q}}$. We refer to M as the *memory size*. The caches are populated by the server during the placement phase, without knowledge of future demands. During the delivery phase, each user communicates to the server its demanded file, and the server transmits a message X of no more than RB symbols in $\mathbb{F}_{\mathsf{q}}$. We refer to R as the *load*. Each user must recover the desired file from the transmit signal and the locally cached content. Mathematically, the system is described as follows.

a) Placement Phase: The cache content of user k is denoted by $Z_k \in \mathbb{F}_{\mathsf{q}}^{\mathsf{MB}}$ and satisfies

$$H(Z_k|F_1,\ldots,F_{\mathsf{N}}) = 0, \quad \forall k \in [\mathsf{K}]. \tag{1}$$

b) Delivery Phase: Let $d_k \in [\mathsf{N}]$ be the index of the file demanded by user $k \in [\mathsf{K}]$. After the demands are known, the server transmits a message $X \in \mathbb{F}_{\mathsf{q}}^{\mathsf{RB}}$ to the users, where

$$H(X|d_1,\ldots,d_{\mathsf{K}},F_1,\ldots,F_{\mathsf{N}}) = 0. \tag{2}$$

c) Decoding: Each user recovers its desired file from the local cached content and the transmitted signal, i.e.,

$$H(F_{d_k}|X,Z_k) = 0, \quad \forall k \in [\mathsf{K}]. \tag{3}$$

d) Load: The goal is to characterize

$$\mathsf{R}^\star(\mathsf{M}) = \limsup_{\mathsf{B},\mathsf{q}} \min_{X,Z_1,\ldots,Z_{\mathsf{K}}} \max_{d_1,\ldots,d_{\mathsf{K}}} \{\mathsf{R} : \text{is achievable with cache size } \mathsf{M}\}, \ \forall \mathsf{M} \in [0,\mathsf{N}]. \tag{4}$$

B. Linear Coded Placement

Denote by $F := [F_1;\ldots;F_{\mathsf{N}}] \in \mathbb{F}_{\mathsf{q}}^{\mathsf{NB}}$ the column vector that contains all the symbols of all the files. In this work we consider *linear coding placement* (LinP), that is, in (1) we restrict the cached contents to be of the form

$$Z_k = \mathbf{E}_k F \in \mathbb{F}_{\mathsf{q}}^{\mathsf{MB}}, \ \forall k \in [\mathsf{K}], \tag{5}$$

where $\mathbf{E}_k \in \mathbb{F}_{\mathsf{q}}^{\mathsf{MB}\times\mathsf{NB}}$ is the *cache encoding matrix* for user k. The optimal load under LinP is defined as in (4) but with the LinP constraint in (5) (instead of (1)), and is denoted as $\mathsf{R}^\star_{\text{LinP}}(\mathsf{M})$. For $\mathsf{M} \in [0,\mathsf{N}]$, we know

$$\frac{\mathsf{R}_{\text{YMA}}(\mathsf{M})}{2.00884} \le \mathsf{R}^\star(\mathsf{M}) \le \mathsf{R}^\star_{\text{LinP}}(\mathsf{M}) \le \mathsf{R}_{\text{YMA}}(\mathsf{M}), \tag{6}$$

where R_{YMA} in (6) is the optimal load under uncoded placement [2], [3], i.e., each row in each cache encoding matrix $\mathbf{E}_k$ in (5) has at most one non-zero entry, and where the first inequality in (6) was proved in [4].

Next we outline known achievable schemes with LinP that strictly improve on uncoded placement and that are actually optimal, i.e., cases for which $\mathsf{R}^\star(\mathsf{M}) = \mathsf{R}^\star_{\text{LinP}}(\mathsf{M}) < \mathsf{R}_{\text{YMA}}(\mathsf{M})$ for some M, usually is the small memory regime.

C. Optimal LinP Scheme for $\mathsf{N} = \mathsf{K}$ and $\mathsf{M} = 1/\mathsf{K}$ [5]

Partition each file into K equal-length subfiles as $F_n = [F_{n,1}, \ldots, F_{n,\mathsf{K}}],\ \forall n \in [\mathsf{N}]$. In the placement phase, the cache content of user $k \in [\mathsf{K}]$ is $Z_k = \sum_{n\in[\mathsf{N}]} F_{n,k}$. For the delivery phase, the worst case demand is when all files are requested; we thus consider without loss of generality (up to a permutation of the indices of the users) the demand $d_k = k, \forall k \in [\mathsf{K}]$, for which the server sends $X = \cup_{k\in[\mathsf{K}]}(F_{d_k,j} : j \in [\mathsf{K}] \setminus \{k\})$ that is, the server sends to each user the subfiles that 'interfere' with the subfiles of its demands file within it cache. The way we interpret the delivery phase is that, for every user, the server sends subfiles that simultaneously 'unlock' its cache while also being useful for other users. The load is

$$\mathsf{R}_{\text{CFL}}(1/\mathsf{K}) = \mathsf{N}(\mathsf{N}-1)/\mathsf{N} = \mathsf{N} - 1, \tag{7}$$

which is optimal as it meets the cut-set bound [5].

We note that the scheme described in this subsection can be extended to any $\mathsf{N} \leq \mathsf{K}$ and $\mathsf{M} = 1/\mathsf{K}$ [5]; we shall not describe there this generalization (which requires a second stage in the delivery, in addition to what we described for $\mathsf{N} = \mathsf{K}$) due to space limitation and because in this work we focus on $\mathsf{N} = \mathsf{K}$ only.

D. LinP scheme for $\mathsf{N} = \mathsf{K}$ from [6]

In [6], Tian and Chen introduced a framework with LinP for $\mathsf{N} \leq \mathsf{K}$ that uses a combination of rank metric codes and maximum distance separable codes for the placement phase, and a delivery phase designed to serve the purpose of content delivery to intended users and interference elimination for unintended users. This scheme improves on existing literature in some memory regime but has a drawback: codes are from a large finite field, rather than the binary filed for schemes with uncoded placements.

For the scheme we propose in this paper, the delivery is also inspired by interference elimination; our scheme however only needs a finite filed of size three and it is provably optimal for $\mathsf{N} = \mathsf{K}$ at $\mathsf{M} = 1/(\mathsf{N}-1)$ and outperforms [6]. Indeed, for every $t \in [\mathsf{K}]$ and $\mathsf{N} = \mathsf{K}$, the lower convex envelope of the following points is achievable [6]

$$(\mathsf{M}_t, \mathsf{R}_t)_{\text{TC}} = \left(\frac{t^2}{\mathsf{N}}, \mathsf{N} - t\right). \tag{8}$$

When $t = 1$, the tradeoff in (8) is equivalent to (7). When $t = 2$ and $\mathsf{N} \geq 4$, it can be easily verified that the tradeoff in (8) is worse than what can be achieved with uncoded placement, that is, $\mathsf{R}_{\text{YMA}}(4/\mathsf{N}) < \mathsf{N} - 2 = \mathsf{R}_{\text{TC}}(4/\mathsf{N})$, where $\mathsf{R}_{\text{YMA}}(\mathsf{M})$ is the load from [3]. For $\mathsf{M} = 1/(\mathsf{N}-1)$ we have

$$\begin{aligned}
\mathsf{R}_{\text{TC}}\left(\frac{1}{\mathsf{N}-1}\right) &= \mathsf{R}_{\text{TC}}\left((1 - \frac{1}{3(\mathsf{N}-1)})\frac{1}{\mathsf{N}} + \frac{1}{3(\mathsf{N}-1)}\frac{4}{\mathsf{N}}\right) \\
&= (1 - \frac{1}{3(\mathsf{N}-1)})\,(\mathsf{N}-1) + \frac{1}{3(\mathsf{N}-1)}\,(\mathsf{N}-2) \\
&= \mathsf{N} - 1 + \frac{1}{3(\mathsf{N}-1)} \leq \mathsf{N} - 1 - \frac{1}{\mathsf{N}} = \mathsf{R}^{\star}\left(\frac{1}{\mathsf{N}-1}\right),
\end{aligned} \tag{9}$$

where the last equality is shown in Section III-A and is attained by the scheme we propose in the next section.

E. Optimal LinP Scheme for $\mathsf{N} = \mathsf{K} = 3$ *and* $\mathsf{M} = 1/2$ *[7]*

In our Allerton 2023 paper, we determined $\mathsf{R}^{\star}_{\text{LinP}}(\mathsf{M})$ for $\mathsf{N} = \mathsf{K} = 3$. We showed the existence of an optimal corner point for $\mathsf{M} = 1/2$, which was unknown before. The new discovered point $(\mathsf{M}, \mathsf{R}) = (1/2, 5/3)$ meets with equality the converse bound $8 \leq 6\mathsf{M} + 3\mathsf{R}$ derived in [10] for $\mathsf{N} = \mathsf{K} = 3$. In Fig. 1, the region in green can be achieved by LinP [7], while in the gray region memory sharing between our new optimal corner point [7] and the YMA scheme [3] does not meet the converse bound from [10] (which is the best known converse bound for $\mathsf{N} = \mathsf{K} = 3$). For the gray region, non-linear coding placement may be needed; it is actually known that the corner point on the converse bound given by $(\mathsf{M}, \mathsf{R}) = (2/3, 4/3)$ cannot be achieved by LinP [10].

We next describe the scheme in [7] for $\mathsf{N} = \mathsf{K} = 3$ that achieves the optimal point $(\mathsf{M}, \mathsf{R}) = (1/2, 5/3)$.

a) Placement Phase: We partition each file into 6 equal-length subfiles as follows

$$F_1 = [A_1^{(1)}, A_2^{(1)}, A_1^{(2)}, A_2^{(2)}, A_1^{(3)}, A_2^{(3)}], \tag{10a}$$

$$F_2 = [B_1^{(1)}, B_2^{(1)}, B_1^{(2)}, B_2^{(2)}, B_1^{(3)}, B_2^{(3)}], \tag{10b}$$

$$F_3 = [C_1^{(1)}, C_2^{(1)}, C_1^{(2)}, C_2^{(2)}, C_1^{(3)}, C_2^{(3)}]. \tag{10c}$$

We place *coded content* into each cache as follows

$$Z_1 = [A_1^{(1)} + B_1^{(1)},\ A_2^{(1)} + C_1^{(1)},\ B_2^{(1)} + C_2^{(1)}] \tag{11a}$$

$$Z_2 = [A_1^{(2)} + B_1^{(2)},\ A_2^{(2)} + C_1^{(2)},\ B_2^{(2)} + C_2^{(2)}], \tag{11b}$$

$$Z_3 = [A_1^{(3)} + B_1^{(3)},\ A_2^{(3)} + C_1^{(3)},\ B_2^{(3)} + C_2^{(3)}]. \tag{11c}$$

b) Delivery Phase: If every user demands the same file, we send all the subfiles of the demanded file, for a load of $6 \times 1/6 = 1 < 5/3$.

If two distinct files are demanded, we cannot send both of them uncoded as the load would be $2 \times 6 \times 1/6 = 2 > 5/3$. The delivery in this case is as follows. Assume demand vector $[1, 2, 2]$, that is, user 1 wants A and the other two users want B. We start by sending $(B_1^{(1)}, A_1^{(2)}, A_1^{(3)})$, then we send $(A_2^{(1)}, A_2^{(2)}, A_2^{(3)},\ B_2^{(1)}, B_2^{(2)}, B_2^{(3)},\ B_1^{(2)} + B_1^{(3)})$. One can readily verify that all users are able to recover the demanded file. The load is $10 \times 1/6 = 5/3$, as claimed.

Let us now focus on the case where all three files are demanded. Without loss of generality (i.e., up to a permutation of the user indices) let us consider the demand vector $[1, 2, 3]$, that is, user 1 wants A, user 2 wants B, and user 3 wants C. The delivery phase has *three stages*. For this example, the server overall transmits $X = (X_1, X_2, X_3)$ with X_i the transmitted signal in delivery stage i, $i \in [3]$, where

$$X_1 = (B_1^{(1)}, C_1^{(1)},\ A_1^{(2)}, C_2^{(2)},\ A_2^{(3)}, B_2^{(3)}), \tag{12a}$$

$$X_2 = (A_1^{(3)} - A_2^{(2)},\ B_2^{(1)} - B_1^{(3)},\ C_2^{(1)} - C_1^{(2)}), \tag{12b}$$

$$X_3 = (A_1^{(3)} + A_2^{(2)} + B_2^{(1)} + B_1^{(3)} + C_2^{(1)} + C_1^{(2)}). \tag{12c}$$

c) Decoding: We now explain how the users decode their desired file from (12) and the cached contents in (11).

The messages in stage 1 in (12a) are meant to 'unlock' the coded cache contents in (11). Take user 1 as an example: he wants $A_1^{(1)}$ and $A_2^{(1)}$, which are in its cache but are 'interfered' by $B_2^{(1)}$ and $C_1^{(1)}$, respectively; so we send $B_2^{(1)}$ and $C_1^{(1)}$, to allow him to recover $A_1^{(1)}$ and $A_2^{(1)}$. Note that $B_2^{(1)}$ and $C_1^{(1)}$ are also simultaneously useful for user 2 and user 3, respectively. We do the same for the other users.

After this first 'uncoded' stage, we have an equivalent coded caching problem where

$$\text{user 1 wants } \mathbf{f}_1 = \begin{bmatrix} A_1^{(3)} \\ A_2^{(2)} \end{bmatrix}, \text{ has } Z_1' = B_2^{(1)} + C_2^{(1)}; \tag{13a}$$

$$\text{user 2 wants } \mathbf{f}_2 = \begin{bmatrix} B_2^{(1)} \\ B_1^{(3)} \end{bmatrix}, \text{ has } Z_2' = C_1^{(2)} + A_2^{(2)}; \tag{13b}$$

$$\text{user 3 wants } \mathbf{f}_3 = \begin{bmatrix} C_2^{(1)} \\ C_1^{(2)} \end{bmatrix}, \text{ has } Z_3' = A_1^{(3)} + B_1^{(3)}. \tag{13c}$$

The missing subfiles for user $k,\ k \in [3]$, are collected in column vector $\mathbf{f}_k$ in (13), which allows us to express the cached contents in (13) as

$$\begin{bmatrix} Z_1' \\ Z_2' \\ Z_3' \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{e}_1 & \mathbf{e}_1 \\ \mathbf{e}_2 & \mathbf{0} & \mathbf{e}_2 \\ \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \end{bmatrix}, \tag{14}$$

where we defined row vectors $\mathbf{0} = [0,\, 0]$, $\mathbf{e}_1 = [1,\, 0]$ and $\mathbf{e}_2 = [0,\, 1]$. From (14), each user misses two subfiles and has nothing related to them in its cache. Thus, each user needs to receive at least two linearly independent equations of its missing subfiles. In stage 2, we send X_2 in (12b) that contains one linear combination of the missing subfiles for each user, i.e., the k-th component of X_2 is

$$X_{2,k} = \mathbf{a}\,\mathbf{f}_k, \quad \mathbf{a} := [1,\, -1]. \tag{15}$$

In stage 3, we send X_3 in (12c) that contains one single linear combination of all missing subfiles as

$$X_3 = \mathbf{b} \sum_{k \in [3]} \mathbf{f}_k, \quad \mathbf{b} := [1,\, 1]. \tag{16}$$

Let us see how each users decodes its two missing subfiles. Take user 1 as an example. From its cached content in (13a) and the transmit signals in stage 2 and stage 3, user 1 extracts $A_2^{(2)} + A_1^{(3)}$ as

$$X_3 + X_{2,2} + X_{2,3} - 2Z_1' = A_1^{(3)} + A_2^{(2)} = \mathbf{b}\,\mathbf{f}_1. \tag{17}$$

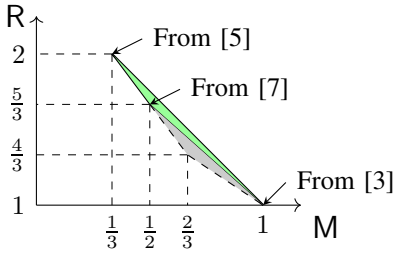

Fig. 1: Memory-load tradeoff in the regime $\mathsf{M} \leq 1$ (and the resulting load is in the range $[1, 2]$) for $\mathsf{K} = \mathsf{N} = 3$. We proved that the green region is achievable by LinP in [7]. The dashed line at the bottom of the gray region is the converse bound [10]. In the gray region, known inner and outer bounds do not match and only non-linear coding placement could improve on [7].

With the 'interference-free' computed signal in (17) together with $X_{2,1} = \mathbf{a}\,\mathbf{f}_1$ in (15), user 1 has two equations in two unknowns. We thus chose the vectors $\mathbf{a}$ and $\mathbf{b}$ to form a full rank (thus invertible) square matrix

$$\mathbf{G} := \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \tag{18}$$

so that user 1 recovers $\mathbf{f}_1$ as

$$\mathbf{f}_1 = \mathbf{G}^{-1} \begin{bmatrix} X_{2,1} \\ X_3 + X_{2,2} + X_{2,3} - 2Z_1' \end{bmatrix}. \tag{19}$$

We proceed similarly for the other users.

d) Load: Each users cached 3 messages of size $1/6$ of a file. The server sends no more than 10 messages of size $1/6$ of a file. Thus the point $(1/2, 5/3)$ is achievable. Note: for this scheme to work, we need a field size of cardinality at least 3, so the square matrix in (18) is invertible.

Our goal in this paper is to generalize this scheme to all values of $\mathsf{N} = \mathsf{K} \geq 3$.

III. New Scheme for $\mathsf{N} = \mathsf{K} > 3$ and $\mathsf{M} = 1/(\mathsf{N} - 1)$

Our main result in this paper is the generalize our scheme for $\mathsf{N} = \mathsf{K} = 3$ in Section II-E to any $\mathsf{N} = \mathsf{K} \geq 3$:

Theorem 1. Given an (N, N) coded caching system, the point

$$(\mathsf{M}, \mathsf{R}) = \left(\frac{1}{\mathsf{N} - 1}, \mathsf{N} - 1 - \frac{1}{\mathsf{N}} \right) \tag{20}$$

is achievable with LinP and is optimal.

A. Converse

We start by identifying a lower bound that we then aim to shows to be achievable. Unfortunately, [10] does not provide lower bounds for $\mathsf{N} = \mathsf{K} > 3$. However, we note that the lower bound from [10] we used in Section II-E for the case $\mathsf{N} = \mathsf{K} = 3$ is a special case of [9]. By letting $\ell = 1$ and $\mathsf{K} = \mathsf{N}$ in [9, Corollary 1] we obtain

$$\mathsf{R}^\star \geq \max_{s\in[\mathsf{N}]} \mathsf{R}_s^{(\mathrm{STC})}, \tag{21}$$

$$\mathsf{R}_s^{(\mathrm{STC})} := \frac{\mathsf{N}^2 - (\mathsf{N}-s)^2}{\mathsf{N}} - s\mathsf{M}. \tag{22}$$

We next consider two values for the parameter s in (22), namely $s \in \{\mathsf{N}, \mathsf{N}-1\}$, and get

$$\mathsf{R}_{\mathsf{N}}^{(\mathrm{STC})} = \mathsf{N} - \mathsf{N}\mathsf{M}, \text{ (cut-set bound);} \tag{23}$$

$$\mathsf{R}_{\mathsf{N}-1}^{(\mathrm{STC})} = \frac{\mathsf{N}^2-1}{\mathsf{N}} - (\mathsf{N}-1)\mathsf{M}. \tag{24}$$

Note that $\mathsf{R}_{\mathsf{N}}^{(\mathrm{STC})} = \mathsf{R}_{\mathsf{N}-1}^{(\mathrm{STC})}$ at $\mathsf{M} = \frac{1}{\mathsf{N}}$, which gives

$$\mathsf{R}^\star(1/\mathsf{N}) \geq \mathsf{N} - 1; \tag{25}$$

the point $(\mathsf{M}, \mathsf{R}) = (\frac{1}{\mathsf{N}}, \mathsf{N}-1)$ is achievable–see Section II-C. Inspired by optimal point for $\mathsf{N} = \mathsf{K} = 3$ in Section II-E, we evaluate the bound $\mathsf{R}_{\mathsf{N}-1}^{(\mathrm{STC})}$ at $\mathsf{M} = \frac{1}{\mathsf{N}-1}$ and obtain

$$\mathsf{R}^\star\left(\frac{1}{\mathsf{N}-1}\right) \geq \frac{\mathsf{N}^2-\mathsf{N}-1}{\mathsf{N}} = \frac{\mathsf{N}^2(\mathsf{N}-2)+1}{\mathsf{N}(\mathsf{N}-1)}. \tag{26}$$

We now shall prove that the point in (20), rewritten as

$$(\mathsf{M}, \mathsf{R}) = \left(\frac{\mathsf{N}}{\mathsf{N}(\mathsf{N}-1)}, \frac{\mathsf{N}^2(\mathsf{N}-2)+1}{\mathsf{N}(\mathsf{N}-1)}\right), \tag{27}$$

is achievable.

B. Achievability

Recall that we focus on $\mathsf{K} = \mathsf{N}$ and we aim to achieve (27).

a) Placement Phase: For every $n \in [\mathsf{N}]$, we partition each file into $\mathsf{N}(\mathsf{N}-1)$ equal-length subfiles, denoted as

$$F_n = \left[F_{n,j}^{(k)} : k \in [\mathsf{K}], j \in [\mathsf{N}-1]\right] \in \mathbb{F}^{\mathsf{N}(\mathsf{N}-1)\times 1}, \tag{28}$$

where we consider each subfile $F_{n,j}^{(k)}$ as an element in a finite field $\mathbb{F}$ of sufficiently large size. We use the superscript (k) to indicate the subfiles that are placed in the cache of user $k \in [\mathsf{K}]$.

For every user $k \in [\mathsf{K}]$, we place N linear combinations of the $\mathsf{N}^2(\mathsf{N}-1)$ subfiles in each cache as

$$Z_k = \mathbf{E}\Big[F_{n,j}^{(k)} : n \in [\mathsf{N}], j \in [\mathsf{N}-1]\Big] \in \mathbb{F}^{\mathsf{N}\times 1}, \tag{29}$$

with encoding matrix $\mathbf{E}$ given by

$$\mathbf{E} := [\underbrace{\mathbf{I}_{\mathsf{N}}, \mathbf{I}_{\mathsf{N}}, \ldots, \mathbf{I}_{\mathsf{N}}}_{\mathsf{N}-1 \text{ times}}] \in \mathbb{F}^{\mathsf{N}\times \mathsf{N}(\mathsf{N}-1)}, \tag{30}$$

so that each subfile appears in one linear combination and each linear combinations involves $\mathsf{N}-1$ subfiles. Note that we use the same cache encoding matrix $\mathbf{E}$ for all users.

b) Delivery Phase: We consider the case where all N files are demanded[1]. We aim to deliver $\mathsf{N}^2(\mathsf{N}-2)+1$ linear combinations of subfiles in three stages as follows. Without loss of generality (up to a permutation on the indices of the users) assume $d_k = k, \forall k \in [\mathsf{K}]$.

Stage 1: for every user $k \in [\mathsf{K}]$, we send those subfiles to 'unlock' user k desired subfiles within its cache In total we send $\mathsf{N}(\mathsf{N}-1)(\mathsf{N}-2)$ subfiles Each user recovers $(\mathsf{N}-1)^2$ subfiles of its desired file, and still misses $\mathsf{N}(\mathsf{N}-1)-(\mathsf{N}-1)^2 = \mathsf{N}-1$ subfiles of the demanded file.

Let us indicate with

$$\mathbf{f}_k := \Big[F_{d_k,j_{k,i}}^{(i)} : i \in [\mathsf{N}] \setminus \{k\}\Big] \in \mathbb{F}^{\mathsf{N}-1\times 1}, \quad \forall k \in [\mathsf{K}], \tag{31}$$

for some $j_{k,i} \in [\mathsf{N}-1]$, the column vector of subfiles that user k is still missing after stage 1. Note that the missing subfiles at user $k \in [\mathsf{K}]$ in (31) are not in its cache, i.e., $\mathbf{f}_k$ contains exactly one subfile with superscript (i) for all $i \neq k$.

By properly arranging the components in vectors $\mathbf{f}_k$'s in (31), we can write the coded part of the cached contents as

$$Z'_k = \mathbf{e}_k \sum_{j\in[\mathsf{N}]\setminus\{k\}} \mathbf{f}_j \in \mathbb{F}, \quad \forall k \in [\mathsf{N}-1], \tag{32}$$

$$Z'_{\mathsf{N}} = \sum_{j\in[\mathsf{N}-1]} \mathbf{e}_j\, \mathbf{f}_j \in \mathbb{F}, \tag{33}$$

where $\mathbf{e}_j \in \mathbb{F}^{1\times \mathsf{N}-1}$ is the j-th standard basis row vector. This is possible because, for every user $k \in [\mathsf{N}]$, the subfiles in $\mathbf{f}_k$ are not in Z'_k, and each subfile only appears once in all Z'_k.

[1]If the number of distinct demanded files is no more than $\mathsf{N}-2$, we send all the subfiles of the demanded files. If the number of distinct demanded files is $\mathsf{N}-1$, i.e., two users must have demanded the same file, the delivery is a variation on what described here. See also example next.

Stage 2: for each user $k \in [\mathsf{K}]$, we send $\mathsf{N}-2$ linear combinations of the subfiles that are still missing at user k, as

$$X_{2,k,j} = \mathbf{g}_j \, \mathbf{f}_k \in \mathbb{F}, \quad \forall j \in [\mathsf{N}-2], \tag{34}$$

where $\mathbf{g}_j \in \mathbb{F}^{1\times \mathsf{N}-1}$ is an encoding vector to be determined later. We use the same encoding vectors for all users.

Stage 3: we send one linear combination as

$$X_3 = \mathbf{g}_{\mathsf{N}-1} \sum_{k\in[\mathsf{K}]} \mathbf{f}_k \in \mathbb{F}, \tag{35}$$

where $\mathbf{g}_{\mathsf{N}-1} \in \mathbb{F}^{1\times \mathsf{N}-1}$ is an encoding row vector to be determined later.

c) Decoding: User $k \in [\mathsf{K}]$ proceeds to collect

$$\begin{bmatrix} X_{2,k,1} & \dots & X_{2,k,\mathsf{N}-2} & \mathcal{L}_k \end{bmatrix}^T = \mathbf{G}\, \mathbf{f}_k, \quad \forall k \in [\mathsf{N}], \tag{36}$$

where $\mathcal{L}_k$ in (36) denotes a linear combination (to be discussed next) of the cached content and delivered messages

$$(Z'_k, X_3, \{X_{2,u,j} : u \in [\mathsf{K}] \setminus \{k\}, j \in [\mathsf{N}-2]\}), \tag{37}$$

and the square matrix $\mathbf{G}$ in (36) is defined as

$$\mathbf{G} := \begin{bmatrix} \mathbf{g}_1 & \dots & \mathbf{g}_{\mathsf{N}-2} & \mathbf{g}_{\mathsf{N}-1} \end{bmatrix}^T \in \mathbb{F}^{\mathsf{N}-1\times \mathsf{N}-1}. \tag{38}$$

Therefore, successful recovering of the missing subfiles by solving (36) is possible if $\mathbf{G}$ in (38) is full rank.

Next we identify another property the matrix $\mathbf{G}$ needs to satisfy (in addition to be full rank), which is derived by considerations on the linear combinations $\mathcal{L}_k$ in (36). The key observation is that, for each $j \in [\mathsf{N}-1]$, we can find scalars $v_{j,1}, \dots, v_{j,\mathsf{N}-1}$ in $\mathbb{F}$ such that

$$\sum_{\ell\in[\mathsf{N}-1]} v_{j,\ell}\mathbf{g}_\ell = \mathbf{e}_j \iff \begin{bmatrix} v_{j,1}, \cdots, v_{j,\mathsf{N}-1} \end{bmatrix} = \mathbf{e}_j \mathbf{G}^{-1}. \tag{39}$$

Let us indicate with $\mathbf{v} = [\tilde{\mathbf{v}}, v_{\mathsf{N}-1}]$ the row vector of length $\mathsf{N}-1$ obtained by concatenating the row vector $\tilde{\mathbf{v}}$ of length $\mathsf{N}-2$ with the scalar $v_{\mathsf{N}-1}$. Let us indicate with $\tilde{\mathbf{G}}$ the matrix obtained by selecting the first $\mathsf{N}-2$ rows of the matrix $\mathbf{G}$ in (38). Finally, let us indicate with $\mathbf{X}_{2,k} := \tilde{\mathbf{G}}\mathbf{f}_k$. For every user $k \in [\mathsf{N}-1]$, we choose $\mathcal{L}_k$ as

$$\mathcal{L}_k = \tilde{\mathbf{v}}_k \sum_{j\in[\mathsf{N}]\setminus\{k\}} \mathbf{X}_{2,j} + v_{k,\mathsf{N}-1} X_3 - Z'_k \tag{40a}$$

$$= \tilde{\mathbf{v}}_k \tilde{\mathbf{G}} \sum_{j\in[\mathsf{N}]\setminus\{k\}} \mathbf{f}_j + v_{k,\mathsf{N}-1}\mathbf{g}_{\mathsf{N}-1} \sum_{j\in[\mathsf{N}]} \mathbf{f}_j - \mathbf{e}_k \sum_{j\in[\mathsf{N}]\setminus\{k\}} \mathbf{f}_j \tag{40b}$$

$$= (\mathbf{v}_k\mathbf{G} - \mathbf{e}_k) \sum_{j\in[\mathsf{N}]\setminus\{k\}} \mathbf{f}_j + v_{k,\mathsf{N}-1}\mathbf{g}_{\mathsf{N}-1}\mathbf{f}_k \tag{40c}$$

$$= \mathbf{g}_{\mathsf{N}-1}\mathbf{f}_k \qquad \text{if and only if} \tag{40d}$$

$$\mathbf{v}_k = \mathbf{e}_k\mathbf{G}^{-1} \text{ and } v_{k,\mathsf{N}-1} = \mathbf{G}^{-1}\big[\{k\},\{\mathsf{N}-1\}\big] = 1. \tag{40e}$$

Note that row vectors $\mathbf{e}_k\mathbf{G}^{-1}$ exists for all $k \in [\mathsf{N}-1]$ as we chose $\mathbf{G}$ to be invertible, but in addition we also need that the last column of $\mathbf{G}^{-1}$ contains all non-zero element as we want to be able to 'normalize' it to the all-one vector as per (40e).

For user N, we proceed similarly to what done above but with $\mathbf{v}_j = [\tilde{\mathbf{v}}_j, 1]$ because of the conditions in (40e) for all $k \in [\mathsf{N}-1]$; we have

$$\mathcal{L}_{\mathsf{N}} = \sum_{j\in[\mathsf{N}-1]} \tilde{\mathbf{v}}_j \mathbf{X}_{2,j} + X_3 - Z'_{\mathsf{N}} \tag{41a}$$

$$= \sum_{j\in[\mathsf{N}-1]} \tilde{\mathbf{v}}_j \tilde{\mathbf{G}}\mathbf{f}_j + \mathbf{g}_{\mathsf{N}-1} \sum_{j\in[\mathsf{N}]} \mathbf{f}_j - \sum_{j\in[\mathsf{N}-1]} \mathbf{e}_j\mathbf{f}_j \tag{41b}$$

$$= \sum_{j\in[\mathsf{N}-1]} (\mathbf{v}_j\mathbf{G} - \mathbf{e}_j)\mathbf{f}_j + \mathbf{g}_{\mathsf{N}-1}\mathbf{f}_{\mathsf{N}} \tag{41c}$$

$$= \mathbf{g}_{\mathsf{N}-1}\mathbf{f}_{\mathsf{N}}, \tag{41d}$$

where the last equality follows from (40e).

Remark 1. An example of matrix $\mathbf{G}$ that satisfies the conditions in (40e) is

$$\mathbf{G}^{-1} = \begin{bmatrix} \mathbf{I}_{N-2} & \mathbf{1}_{N-2}^T \\ \mathbf{0}_{N-2} & 1 \end{bmatrix}, \tag{42}$$

where $\mathbf{0}_i$ is the all-zero row vector of length i, $\mathbf{1}_i$ is the all-one row vector of length i, and $\mathbf{I}_i$ is the identity matrix of dimension i; thus

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{N-2} & -\mathbf{1}_{N-2} \\ \mathbf{0}_{N-2}^T & 1 \end{bmatrix}. \tag{43}$$

This choice of $\mathbf{G}$ in (43) renders the delivery and the decoding particularly simple, and has the added advantage that only requires a finite filed of size 3, i.e., the finite filed size does not scaling with the system parameters. □

d) Load: In total we transmitted $\mathsf{N}(\mathsf{N}-1)(\mathsf{N}-2)+(\mathsf{N}-2)+1=\mathsf{N}^2(\mathsf{N}-2)+1$ messages with size $\frac{1}{\mathsf{N}(\mathsf{N}-1)}$, thus the load is $\mathsf{R}=\mathsf{N}-1-\frac{1}{\mathsf{N}}$ as claimed.

C. Example

We conclude this section with another example, for $\mathsf{N}=\mathsf{K}=4$, to further clarify the notation and the various stages of the delivery with the choice of matrix $\mathbf{G}$ in (43).

The files are denoted as A, B, C, D. We first partition each file into 12 equal-length subfiles, similarly to (10).

a) Placement Phase: For every user $k \in [4]$, the server populates cache content for Z_k as follows,

$$Z_k = \begin{bmatrix} A_1^{(k)} + B_2^{(k)} + C_3^{(k)} \\ A_2^{(k)} + B_3^{(k)} + D_1^{(k)} \\ A_3^{(k)} + C_1^{(k)} + D_2^{(k)} \\ B_1^{(k)} + C_2^{(k)} + D_3^{(k)} \end{bmatrix}. \tag{44}$$

b) Delivery Phase with at most $\mathsf{N}-2=2$ *Distinct Demanded Files:* We send the demanded files. In total, we transmit no more than 24 subfiles.

c) Delivery Phase with $\mathsf{N}-1=3$ *Distinct Demanded Files:* Assume demands $[A, B, C, C]$. The server first sends

$$X_1 = (B_2^{(1)}, C_3^{(1)}, A_1^{(2)}, C_3^{(2)}, A_1^{(3)}, B_2^{(3)}, A_1^{(4)}, B_2^{(4)}). \tag{45}$$

Next the server transmits all missing files for users 1 and 2 (who demand a file that no one else demands), as

$$X_2 = \begin{pmatrix} A_2^{(1)} & A_3^{(1)} & A_2^{(2)} & A_3^{(2)} \\ A_2^{(3)} & A_3^{(3)} & A_2^{(4)} & A_3^{(4)} \\ B_1^{(1)} & B_3^{(1)} & B_1^{(2)} & B_3^{(2)} \\ B_1^{(3)} & B_3^{(3)} & B_1^{(4)} & B_3^{(4)} \end{pmatrix}. \tag{46}$$

For users 3 and 4 (who demand the same file), we first transmit those subfiles demanded by both, then we send one multicast message that involves subfiles known by only one of them

$$X_3 = \begin{pmatrix} C_1^{(1)} & C_2^{(1)} & C_1^{(2)} & C_2^{(2)} \\ C_1^{(3)} & C_2^{(3)} & C_1^{(4)} & C_2^{(4)} \end{pmatrix} \cup \{C_3^{(3)} + C_3^{(4)}\}. \tag{47}$$

In total, we transmit the equivalent of 33 subfiles.

d) Delivery Phase when all $\mathsf{N} = 4$ *Files are Demanded:* Assume demand vector to be $[A, B, C, D]$. We use

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{G}^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{48}$$

In stage 1, we 'unlock' the desired subfiles for each user as

$$X_1 = \begin{pmatrix} B_2^{(1)} & C_3^{(1)} & B_2^{(1)} & D_1^{(1)} & C_1^{(1)} & D_2^{(1)} \\ A_1^{(2)} & C_3^{(2)} & A_2^{(2)} & D_1^{(2)} & C_2^{(2)} & D_3^{(2)} \\ A_1^{(3)} & B_2^{(3)} & A_3^{(3)} & D_2^{(3)} & B_1^{(3)} & D_3^{(3)} \\ A_2^{(4)} & B_2^{(4)} & A_3^{(4)} & C_1^{(4)} & B_1^{(4)} & C_2^{(4)} \end{pmatrix} \tag{49}$$

For every user $k \in [4]$, the vector of missing subfiles for user k after stage 1 as follows

$$\mathbf{f}_1 = [A_1^{(4)}, A_3^{(2)}, A_2^{(3)}]^T, \tag{50a}$$

$$\mathbf{f}_2 = [B_1^{(1)}, B_2^{(4)}, B_3^{(3)}]^T, \tag{50b}$$

$$\mathbf{f}_3 = [C_2^{(1)}, C_1^{(2)}, C_3^{(4)}]^T, \tag{50c}$$

$$\mathbf{f}_4 = [D_3^{(1)}, D_2^{(2)}, D_1^{(3)}]^T. \tag{50d}$$

and the caches can be expressed as

$$Z_1' = B_1^{(1)} + C_2^{(1)} + D_3^{(1)} = \mathbf{e}_1(\mathbf{f}_2 + \mathbf{f}_3 + \mathbf{f}_4), \tag{50e}$$

$$Z_2' = A_3^{(2)} + C_1^{(2)} + D_2^{(2)} = \mathbf{e}_2(\mathbf{f}_1 + \mathbf{f}_3 + \mathbf{f}_4), \tag{50f}$$

$$Z_3' = A_2^{(3)} + B_3^{(3)} + D_1^{(3)} = \mathbf{e}_3(\mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_4), \tag{50g}$$

$$Z_4' = A_1^{(4)} + B_2^{(4)} + C_3^{(4)} = \mathbf{e}_1\mathbf{f}_1 + \mathbf{e}_2\mathbf{f}_2 + \mathbf{e}_3\mathbf{f}_3. \tag{50h}$$

In stage 2 we send

$$\mathbf{X}_{2,1} = \tilde{\mathbf{G}}\mathbf{f}_1 = \begin{bmatrix} A_1^{(4)} - A_2^{(3)} \\ A_3^{(2)} - A_2^{(3)} \end{bmatrix}, \mathbf{X}_{2,2} = \tilde{\mathbf{G}}\mathbf{f}_2 = \begin{bmatrix} B_1^{(1)} - B_3^{(3)} \\ B_2^{(4)} - B_3^{(3)} \end{bmatrix}, \tag{51a}$$

$$\mathbf{X}_{2,3} = \tilde{\mathbf{G}}\mathbf{f}_3 = \begin{bmatrix} C_2^{(1)} - C_3^{(4)} \\ C_1^{(2)} - C_3^{(4)} \end{bmatrix}, \mathbf{X}_{2,4} = \tilde{\mathbf{G}}\mathbf{f}_4 = \begin{bmatrix} D_3^{(1)} - D_1^{(3)} \\ D_2^{(2)} - D_1^{(3)} \end{bmatrix}, \tag{51b}$$

and in stage 3 we send

$$X_3 = \mathbf{g}_3 \sum_{j \in [4]} \mathbf{f}_j = A_2^{(3)} + B_3^{(3)} + C_3^{(4)} + D_1^{(3)}. \tag{51c}$$

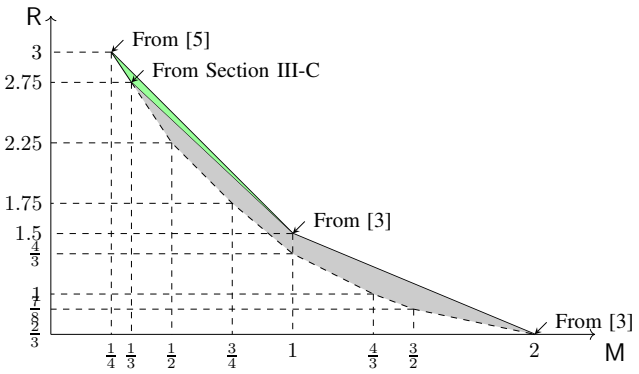

Fig. 2: Memory-load tradeoff in the regime $\mathsf{M} \in [1/4, 2]$ (for which the load is in the range $[2/3, 3]$) for $\mathsf{K} = \mathsf{N} = 4$. The green region is achieved by our new LinP scheme. The dashed line at the bottom of the gray region is a converse bound from [4]. In the gray region, known inner and outer bounds do not match.

e) Decoding: We verify that $\mathcal{L}_k$ delivers the desired missing linearly independent linear combination for every user $k \in [\mathsf{N}]$, indeed

$$\mathcal{L}_1 = X_3 + X_{2,2,1} + X_{2,3,1} + X_{2,4,1} - Z'_1 = A_2^{(3)}, \tag{52a}$$

$$\mathcal{L}_2 = X_3 + X_{2,1,2} + X_{2,3,2} + X_{2,4,2} - Z'_2 = B_3^{(3)}, \tag{52b}$$

$$\mathcal{L}_3 = X_3 - Z'_3 = C_3^{(4)}, \tag{52c}$$

$$\mathcal{L}_4 = X_3 + X_{2,1,1} + X_{2,2,2} - Z'_4 = \mathbf{g}_{\mathsf{N}-1}\mathbf{f}_4 = D_1^{(3)}. \tag{52d}$$

Thus, every user $k \in [4]$ recovers it desired subfiles. The memory-load tradeoff is shows in Fig. 2.

IV. Conclusions

In this paper we propose a scheme with linear coded placement for $\mathsf{N} = \mathsf{K}$, which is optimal and extends the known memory regime over which optimality is known to $\mathsf{M} \in [0, 1/(\mathsf{N}-1)]$. Open questions include extension of known results with linear coded placement from the single file retrieval to scalar linear function retrieval.

This work has been supported in part by NSF Awards 1910309 and 2312229.

References

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

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

[3] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, 2017.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 647–663, 2018.

[5] Z. Chen, P. Fan, and K. B. Letaief, "Fundamental limits of caching: Improved bounds for users with small buffers," *IET Communications*, vol. 10, no. 17, pp. 2315–2318, 2016.

[6] C. Tian and J. Chen, "Caching and delivery via interference elimination," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1548–1560, 2018.

[7] Y. Ma and D. Tuninetti, "Coded caching with linear coded placement: Exact tradeoff for the three user case," in *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2023.

[8] Y. Yao and S. A. Jafar, "The capacity of 3 user linear computation broadcast," *arXiv preprint arXiv:2206.10049*, 2022.

[9] A. Sengupta and R. Tandon, "Improved approximation of storage-rate tradeoff for caching with multiple demands," *IEEE Transactions on Communications*, vol. 65, no. 5, pp. 1940–1955, 2017.

[10] C. Tian, "Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching," *Entropy*, vol. 20, no. 8, p. 603, 2018.