Augmenting Multiple-Transmitter Coded Caching using Popularity Knowledge at the Transmitters

Berksan Serbetci, Eleftherios Lampiris, Thrasyvoulos Spyropoulos, Petros Elia
serbetci@eurecom.fr, lampiris@tu-berlin.de, {spyropou, elia}@eurecom.fr

***Abstract*— The work presents a new way of exploiting non-uniform file popularity in caching networks. Focusing on the interference channel with cache-enabled transmitters and receivers, we show how non-uniform file popularity can be used to accelerate the impact of transmitter-side data redundancy in coded caching. This approach is motivated by the recent discovery that under realistic file-size constraints, having content appear in multiple transmitters can boost multiplicatively the speed-up factor attributed to coded caching.**

We formulate the problem through an optimization algorithm, which seeks to optimize the number of transmitters each file is cached at, as a function of that file's popularity. Part of the optimization effort involves a biconvex problem; such problems are traditionally solved by heuristic Alternate Convex Search methods that generally do not guarantee the global optimum. To avoid this, we follow a more involved path which includes the design of a new search algorithm that exploits the properties of the caching problem itself. The overall optimization algorithm provably achieves the globally optimal solution, and does so with a complexity that scales as a polynomial function of the logarithm of the size of the file catalog. In the end, the optimal transmitter-side cache placement yields multiplicative speedup factors over traditional multi-transmitter coded caching algorithms.

I. Introduction

In the context of cache-aided interference-limited communication networks, the work of Maddah-Ali and Niesen [1] revealed how content that is properly placed at the receiver-side caches, can serve as side information that cancels interference and accelerates delivery. Key to this was an ability to employ multicasting even when delivering content that is different from user to user.

In particular, the work in [1] considered a single-stream broadcast (downlink) configuration, where a transmitter with access to a library of N unit-sized files, serves – via a unit-capacity bottleneck link – K receiving users, each endowed with a cache of size equal to the size of M files, or equivalently equal to a fraction $\gamma \triangleq \frac{M}{N}$ of the library. In a setting that involved a *cache-placement phase* and a subsequent *delivery phase* that starts with the users concurrently requesting a file each, the *coded caching* algorithm in [1] allowed for complete delivery of any K independent files, at a rate of $K\gamma + 1$ files at a time, resulting in a worst-case delivery time of

$$T = \frac{K(1-\gamma)}{1+K\gamma}. \tag{1}$$

The work is supported by the ANR-17-CE25-0001 Jeunes Chercheuses et Jeunes Chercheurs project 5C-for-5G, by the ERC project DUALITY (grant agreement no. 725929), and by the ERC project CARENET (grant agreement no. 789190). A longer version of this work is posted on arXiv.

This rate of $K\gamma + 1$ – commonly referred to as the Degrees of Freedom (DoF) performance – simply reflects the speed-up factor due to caching, or equivalently the delivery rate, in units of *files per unit of time*, after normalization by the high-SNR link capacity. This performance – which suggests that, in theory, one can serve an infinite number of files with bounded delay – was shown to be within a multiplicative factor of 2.01 from the information-theoretic optimal [2], and exactly optimal over the class of schemes that employ uncoded cache placement [3]. Key to this performance was a *cached data redundancy* which guaranteed the proper placement of any specific content at $K\gamma$ different caches. This meant that when this specific content is served (as part of an XOR with other contents) to a specific user, then $K\gamma$ other users (not interested in it) can find that part of the XOR in their cache, and cancel it out from that XOR.

A. Subpacketization and the redundancy constraint

While, in theory, the DoF $K\gamma+1$ could increase indefinitely with an increasing K, it soon became clear (see [4]) that such gains could in practice not materialize, partly due to the algorithm's structural requirement that each file be split into an exceedingly large number of subfiles. This number is equal to $\binom{K}{K\gamma}$, and it scales exponentially in K. To date, this constitutes a prohibitive bottleneck which hard-bounds the DoF at modest values, despite progress in interesting works such as [5]–[7].

The above bottleneck forces a dramatic reduction in performance because, when each file size F is bounded, the coded caching algorithm must apply coding over only a bounded number, call it Λ, of users, in order to guarantee that

$$\binom{\Lambda}{\Lambda\gamma} \leq F. \tag{2}$$

As a result, delivery is repeated K/Λ times, yielding a reduced DoF $\Lambda\gamma + 1$, and an increased delivery time

$$T_\Lambda = \frac{K(1-\gamma)}{1+\Lambda\gamma}. \tag{3}$$

The above was proven in [8] to be optimal under the assumption of uncoded cache placement.

B. Exploiting file popularity in caching networks

Taking advantage of non-uniform file popularity has been a key concept for Content Centric and Information-Centric Networks [9], [10], as well as in wireless edge caching works [11] that followed the femto-caching ideas of [12]. Nevertheless, the majority of these works focus on very different

setups than the interference-limited scenarios in which coded caching is commonly applied. Even works that attempt to take into account somewhat more sophisticated PHY capabilities like joint-beamforming or MU-MIMO, often implicitly or explicitly assume that content requests are non-interfering and/or asynchronous [13], [14].

On the other hand, early works using the coded caching framework for receiver-side caching, focused on the worst-case metric, in the expense of gains due to non-uniform popularity distribution is lost. Recent efforts such as [15]–[24], explored various ways of incorporating file popularity with coded caching. In practice, due to subpacketization, the overall speed-up factor of the single-stream coded caching remains well below the original theoretic promises.

C. Multiplicative impact of transmitter-side cache redundancy

In an effort to overcome the above subpacketization limitation, some research (see [25], [26], see also [27]–[31] and many others) sought to apply coded caching in conjunction with other network resources, such as multi-antenna arrays.

Some progress in this direction came with the work in [32] and the subsequent work in [8], which revealed the surprising finding that the aforementioned unavoidable reduction in Λ, and the subsequent performance degradation could be compensated, *in a very accelerated manner*, through increased data redundancy at the transmitter side. As the works in [8], [32] showed, in a setting where the transmitter has access to L transmit antennas, under the assumption that $L \leq K/\Lambda$, one could achieve the worst-case delivery time

$$T = \frac{K(1-\gamma)}{L(1+\Lambda\gamma)} \tag{4}$$

and the corresponding DoF of $L\,(1+\Lambda\gamma)$ which was proven in [8] to be exactly optimal under the assumption of uncoded cache placement. Note that the above impact of L is multiplicative; this is in direct contrast to the additive effect that can be found in the unconstrained case of $\Lambda = K$ which enjoys the maximal DoF of $L + K\gamma$ (cf. [25], [26]).

This same powerful multiplicative effect can be found in the setting of interest here. This setting – first introduced in [26] – involves K_T transmitters, each with a cache of size equal to a fraction γ_T of the library, which serve via a fully connected channel, the K cache-aided receivers. In this configuration, the transmitter-side cache redundancy $K_T\gamma_T$ now plays the role of L, and again, as long as $K_T\gamma_T \leq K/\Lambda$, the delay of

$$T = \frac{K(1-\gamma)}{K_T\gamma_T(1+\Lambda\gamma)} \tag{5}$$

implies a multiplicatively boosted DoF of $K_T\gamma_T\,(1+\Lambda\gamma)$.

D. Current contributions: Boosting transmitter-side data redundancy using file popularity

The above described how, the transmitter-side data redundancy $L = K_T\gamma_T$ under limited subpacketization, which forces Λ to be limited, has a powerful multiplicative impact. Here, we seek to exploit file popularity in the transmitter-sided placement, to further boost this impact.

While in a setting that is agnostic to file popularity, each file appears at exactly $K_T\gamma_T$ transmitters, our aim here will be to optimize this placement such that (generally but not always) popular files experience higher redundancy by being cached in more transmitters, while less popular files will inevitably be cached in fewer transmitters. We solve an optimization problem whose objective is to determine the redundancy for each file, under a sum-cache constraint. As we will see later on, the new placement provides substantially speed-up of the delivery of popular files, in a manner that far outweighs the increased delays for the unpopular files simply because these increased delays appear much less often. Part of the effort in our work is to guarantee convergence to the global optimum, and to reduce the complexity of the optimization problem. Both are achieved by designing a new search algorithm that exploits the properties of the caching problem itself.

The approach in our work comes in contrast to existing coded-caching efforts which place emphasis on adapting the receiver-side placement in order to reflect the file popularity.

II. System Model and Caching-Delivery Policies

A. System model

We consider the fully-connected, multi-transmitter coded caching setting where K_T single-antenna transmitters serve, via a fully connected channel, K single-antenna receivers/users. Each transmitter and each receiver can store fraction $\gamma_T \in [\frac{1}{K_T}, 1]$ and fraction $\gamma \in [0,1]$ of the library, respectively. We assume that the library is comprised of N files $W^1, W^2, \ldots, W^N$, and that each file – as stated before – has size F bits[1]. We assume that a single transmitter-to-receiver link has (normalized) capacity equal to one file per unit of time, as well as that the channel between any set of transmitters and receivers is of full rank with probability one[2].

We assume that each user concurrently requests one file, independently of each other, and further that the file popularity follows a distribution known during cache placement. In particular, the solution of the optimization problem itself will assume a Zipf distribution [33] with parameter α, under which the probability that file W^n is requested, takes the form

$$p_n = \frac{n^{-\alpha}}{\sum_{n=1}^{N} n^{-\alpha}},\ \forall n \in [N]. \tag{6}$$

Without loss of generality, we assume that files are indexed with decreasing popularity such that $p_i \geq p_j$ for any $i < j$.

B. Caching and delivery policy

Below we describe the role of the parameters that define the optimization problem, and which – once optimized – define the optimal caching-and-delivery policy. We begin with describing how to split the library into an arbitrary number of sub-libraries, each consisting of a different number of files. Then we define the transmitter-side cache redundancy

[1]This assumption is common, as non-equal sized files can be handled by making a content chunk the basic caching unit, as in [12].

[2]This requirement holds true in many wireless settings as well as in wired settings with network-coding capabilities at the intermediate network nodes.

that a file enjoys depending on the sub-library it belongs to, and we show how such an arbitrary (not yet optimized) redundancy can be achieved under the cache-size constraint. These parameters (number of sub-libraries, size of each sub-library, and redundancy attributed to each sub-library) are mapped into an objective function that reflects their role in the delivery time. The subsequent optimization (in the next section) yields the optimizing solution for these parameters, and thus defines the optimized caching and delivery policies.

1) Library segmentation policy: The first step segments the file library into an arbitrary number of Q disjoint sub-libraries

$$\mathcal{B}_q = [n_{q-1}+1 : n_q],\ q = 1, 2, \ldots, Q \tag{7}$$

where each sub-library $\mathcal{B}_q$ contains all the files that are indexed[3] from $n_{q-1}+1$ to n_q. Our segmentation policy thus results in a first sub-library consisting of the most popular files $\mathcal{B}_1 = \{1, 2, \ldots, n_1\}$, then a second sub-library $\mathcal{B}_2 = \{n_1+1, \ldots, n_2\}$, until the sub-library $\mathcal{B}_Q = \{n_{Q-1}, \ldots, N\}$ containing the least popular files. This segmentation — which asks that each sub-library contains clusters of successively indexed files — will be entirely defined by the Q-length vector

$$\mathbf{n} \triangleq [n_1, n_2, \ldots, n_Q].$$

2) Redundancy assignment policy: Based on the above division, our policy asks that each file from $\mathcal{B}_q$ be cached, in its entirety, at L_q different transmitters. The vector

$$\mathbf{L} \triangleq [L_1, L_2, \ldots, L_Q]$$

will thus fully define the redundancy attributed to each file of the library. Naturally any such allocation must satisfy

$$\sum_{q=1}^{Q} L_q |\mathcal{B}_q| = \sum_{q=1}^{Q} L_q (n_q - n_{q-1}) \leq K_T \gamma_T N \tag{8}$$

because of the cache-size constraint at the transmitter side.

3) Transmitter-side caching policy: To implement the above redundancy allocation, for any $Q, \mathbf{n}, \mathbf{L}$, we extend the approach in [32] to account for multiple sub-libraries with different redundancies.

The placement is done sequentially. We start from the first sub-library and we consecutively cache the whole first file into the first L_1 transmitters, then the second file (of the first sub-library) into transmitters L_1+1 through $1+(2L_1-1 \mod K_T)$, and so on. We note that the selection of the transmitters is always done using the modulo operation, which means that when we place a file at the last transmitter, we continue the process with the first transmitter.

After storing each file of the first sub-library in a total of L_1 transmitters each, we proceed with the second sub-library. Continuing from the transmitter after the one last used, i.e. continuing from transmitter $1+(n_1 \cdot L_1 \mod K_T)$, we again sequentially fill the caches, starting from the first file of the second sub-library, which we now store in L_2 consecutive transmitters, and so on. The process is repeated for each sub-library, with a new L_q, starting every time from the transmitter after the one last used. Overall, the above process which stores each file of sub-library $\mathcal{B}_q$ in exactly L_q distinct transmitters, guarantees the size constraint that each transmitter stores $\gamma_T N$ files. Finally, it is easy to show that we can allow for non-integer L_q's by applying basic memory sharing techniques [26], which can facilitate the continuous relaxation proposed in Section III.

4) Receiver-side caching policy: The receivers will cache using the algorithm of [1], modified to form Λ different caches. Toward this, each file $W^n,\ n \in [N]$, is split into $\binom{\Lambda}{\Lambda\gamma}$ equally-sized subfiles $\{W^n_\tau\}_{\tau \subset [\Lambda], |\tau| = \Lambda\gamma}$, such that each subfile W^n_τ is indexed by a $\Lambda\gamma$-tuple τ whose entries come from $[\Lambda]$. Then the ℓ^{th} cache takes the form

$$\mathcal{Z}_\ell = \{W^n_\tau\ :\ \ell \in \tau, \forall n \in [N]\},\ \forall \ell \in \Lambda \tag{9}$$

which simply means that cache ℓ consists of all subfiles W^n_τ, whose index τ contains ℓ. In the end, each user is assigned one of the Λ caches, in a round-robin manner.

5) Content delivery policy: The delivery scheme will follow exactly the algorithm of [32], and — as a function of any given $(Q, \mathbf{n}, \mathbf{L})$ — it will concurrently serve multiple files from one sub-library at a time. For further details on the precoding structure that allows for this concurrent delivery, the reader is referred to [32]. The transmission policy here will ask that no coding is done across files from different sub-libraries.

The problem is naturally of a stochastic nature, as the number of users requesting files from library $\mathcal{B}_q$, changes as a function of the file demand vector. Assuming that at any particular instance of the problem (i.e., for any fixed file demand vector), there are K_q users requesting files from each library $\mathcal{B}_q$, and assuming that K_q is sufficiently large, then the algorithm in [32] can serve $L_q(1+\Lambda\gamma)$ users at a time. This in turn means that our policy, for this instance, requires delay

$$T_q = \frac{K_q(1-\gamma)}{L_q(1+\Lambda\gamma)} \tag{10}$$

to complete delivery of the requested files from $\mathcal{B}_q$.

The policy also asks that the files of the most popular sub-library $\mathcal{B}_1$ are cached at only one transmitter each, and that they are delivered one at a time, without employing coded caching[4]. For sufficiently large K, this implies delay

$$T_1 = n_1. \tag{11}$$

Our metric of performance will be the delay to deliver all requested files, averaged over all possible demand K-tuples.

The following lemma, which holds for sufficiently large K, identifies this average delay $T(\mathbf{n}, \mathbf{L}, Q)$, for any given fixed choice of $(\mathbf{n}, \mathbf{L}, Q)$.

[3]Note the small abuse of notation where $\mathcal{B}_q$ can interchangeably refer to a set of files or the set of these files' indices. Furthermore the notation omits the dependence on Q, which will always be implied. For completeness of notation, we will assume that $n_0 = 0$, and we note that $n_Q = N$.

[4]Such policy compensates for the inability of the employed algorithm in [32] to exploit the possibility that multiple users ask for the same file. For the interested reader, we note that the problem of multi-transmitter coded caching that exploits natural multicasting, remains an open problem.

Lemma 1. *For any fixed $\mathbf{n}, \mathbf{L}, Q$, the average delay of our caching and delivery policy, takes the form*

$$T(\mathbf{n}, \mathbf{L}, Q) = n_1 + \sum_{q=2}^{Q} \frac{K(1-\gamma)\sum_{j=n_{q-1}+1}^{n_q} p_j}{L_q(1+\Lambda\gamma)}. \quad (12)$$

Proof. Directly from (10) and (11) we can see that, at any given instance of the problem, the delay

$$T = n_1 + \sum_{q=2}^{Q} \frac{K_q(1-\gamma)}{L_q(1+\Lambda\gamma)} \quad (13)$$

guarantees delivery of all requests, one sub-library at a time. In averaging over all demand vectors, K_q becomes a random variable whose average takes the form $\mathbb{E}[K_q] = \sum_{j=n_{q-1}+1}^{n_q} K p_j$. Finally the proof is completed by noticing that $\mathbf{n}, \mathbf{L}, Q$ are independent of the instance of the problem as they are naturally independent of the demand vector. □

III. Optimization Problem

We here seek to optimize the choice of $\mathbf{n}, \mathbf{L}, Q$, in order to attain the optimal average delivery time

$$T^* \triangleq \min_{\mathbf{n},\mathbf{L},Q} \; T(\mathbf{n}, \mathbf{L}, Q). \quad (14)$$

Under our policy and cache-size constraints, and using directly the objective function from Lemma 1 (Eq. (12)), the optimization problem takes the following form.

Problem 1.

$$\min_{\mathbf{n},\mathbf{L},Q} \; T(\mathbf{n}, \mathbf{L}, Q) \quad \text{(P1-a)}$$

$$\mathbf{s.t.} \quad L_q \in [1, K_T], \; \forall q \in \{2, \ldots, Q\}, \quad \text{(P1-b)}$$

$$L_q \leq K \sum_{j=n_{q-1}+1}^{n_q} p_j \min\left\{\frac{1}{\Lambda}, \frac{1-\gamma}{1+\Lambda\gamma}\right\}, \; \forall q \in \{2, \ldots, Q\}, \quad \text{(P1-c)}$$

$$n_1 + \sum_{q=2}^{Q} L_q(n_q - n_{q-1}) \leq K_T \gamma_T N, \quad \text{(P1-d)}$$

$$n_q \geq n_{q-1}, \; \forall q \in \{2, \ldots, Q\}, \quad \text{(P1-e)}$$

$$0 \leq n_q \leq N, \; \forall q \in \{1, \ldots, Q\}, \quad \text{(P1-f)}$$

$$n_q \in \mathbb{Z}, \; \forall q \in \{1, \ldots, Q\}, \quad \text{(P1-g)}$$

$$L_q \geq L_{q+1}, \; \forall q \in \{2, \ldots, Q-1\}, \quad \text{(P1-h)}$$

$$1 \leq Q \leq K_T, \; Q \in \mathbb{Z}. \quad \text{(P1-i)}$$

In the above, the constraint in (P1-b) forces each file to be stored in at least one transmitter, the constraint in (P1-c) is required for the objective function to hold (cf. (13), see also [32]), the constraint in (P1-d) reflects the cache-size constraint, the constraint in (P1-e) follows directly from the construction as the sub-libraries must be divided into files with consecutive ordering, the constraint in (P1-f) reflects the boundaries of the file library, the constraint in (P1-g) forces files to be clustered into sub-libraries without self-segmentation, constraint (P1-h) follows from the idea of allocating more antennas to sub-libraries consisting of more popular files (except for the sub-library where the files will be multicasted and $L_1 = 1$ always holds.), and constraint (P1-i) reflects that there can be at most K_T sub-libraries because there are only K_T options for L_q.

Remark 1. *Problem 1 is non-convex since the optimization variables are discrete.*

A. Overview of optimization steps

We proceed with an overview of the steps (and the notation) that we follow to reach the optimal average delivery time $T^* = \min_{\mathbf{n},\mathbf{L},Q} \; T(\mathbf{n}, \mathbf{L}, Q)$, that solves Problem 1.

Step 1: In Section III-B we prove that, for fixed Q, and for the continuous relaxation of $\mathbf{n}$, finding the optimal

$$T^*(Q) \triangleq \min_{\mathbf{n},\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q). \quad (15)$$

is a biconvex problem.

Step 2: Next, in Section III-C we show that for fixed Q and fixed $\mathbf{n}$, we can formulate a convex optimization problem and solve it using Karush-Kuhn-Tucker (KKT) conditions (cf. [34]) to obtain the optimal redundancy allocation $\mathbf{L}^*(\mathbf{n}, \mathbf{Q})$ and the corresponding optimal average delivery time

$$T^*(\mathbf{n}, Q) \triangleq \min_{\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q). \quad (16)$$

Step 3: Next in Section III-D — with the bi-convex nature of the formulated problem in Step 1 at hand — we design a new search algorithm that exploits the properties of the caching problem itself by adjusting the search space of $\mathbf{n}$. Instead of searching for $\mathbf{n} \in [0, N]^{Q-1}$, we identify a new smaller search space S_Q for $\mathbf{n}$ that guarantees optimality, such that

$$\begin{aligned} T^*(Q) &= \min_{\mathbf{n},\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q) \\ &= \min_{\mathbf{n} \in [0,N]^Q} \; \min_{\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q) \\ &= \min_{\mathbf{n} \in [0,N]^Q} T^*(\mathbf{n}, Q) \quad (17) \\ &= \min_{\mathbf{n} \in S_Q} T^*(\mathbf{n}, Q). \quad (18) \end{aligned}$$

We will prove the transition from Eq. (17) to Eq. (18) in Theorem 2 in Section III-D.

Step 4: With each $T^*(Q)$ available from the previous step, Section III-E presents an algorithm that iteratively computes the optimal $T^* = \min_{Q \in [K_T]} T^*(Q)$.

B. Step 1: Modified problem for fixed Q and biconvexity

We first seek to calculate, for a fixed Q and the continuous relaxation of $\mathbf{n}$, the optimal $T^*(Q) = \min_{\mathbf{n},\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q)$ from (15). For ease of reference, we declare the problem as below.

Problem 2.

$$T^*(Q) = \min_{\mathbf{n},\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q) \quad (19)$$

$$\textit{s.t.} \quad \text{(P1-b)} - \text{(P1-f)}, \text{(P1-h)}.$$

Lemma 2. *Problem 2 is a biconvex problem.*

Proof. Due to lack of space, the proof is relegated to the longer version of this work. □

Biconvex problems are traditionally solved by heuristic Alternate Convex Search (ACS) methods which generally do not guarantee finding the global optimum. Because of this, we will – as outlined before – first compute in Step 2 the optimal $T^*(\mathbf{n}, Q)$ for fixed Q and fixed $\mathbf{n}$, and then we will proceed with Steps 3 and 4 to compute $T^*(Q)$ and then the overall optimal T^*.

C. Step 2: Optimal solution for fixed Q and fixed $\mathbf{n}$

We present our modified optimization problem to find — for fixed Q and $\mathbf{n}$ — the optimal redundancy $L^*(\mathbf{n}, Q)$ and the corresponding optimal

$$T^* = \min_{\mathbf{n},\mathbf{L},Q} \; T(\mathbf{n}, \mathbf{L}, Q) = \min_{\mathbf{n},Q} \underbrace{\left\{ \min_{\mathbf{L}} T(\mathbf{n}, \mathbf{L}, Q) \right\}}_{T^*(\mathbf{n},Q)}.$$

For simplicity, for this subsection only, we will use the simplified notation $\mathbf{L} \triangleq L(\mathbf{n}, Q)$ and $\mathbf{L}^* \triangleq \mathbf{L}^*(\mathbf{n}, Q)$ that assumes the dependence on the fixed Q and $\mathbf{n}$.

Problem 3. *Find the optimal*

$$T^*(\mathbf{n}, Q) = \min_{\mathbf{L}} \; T(\mathbf{n}, \mathbf{L}, Q) \tag{20}$$
$$\textit{s.t.} \quad \text{(P1-b)} - \text{(P1-e)}.$$

Lemma 3. *Problem 3 is a convex optimization problem.*

Proof. This follows from the bi-convexity[5] of Lemma 2. □

We now have the following.

Theorem 1. *The optimal antenna allocation for Problem 3 satisfies*

$$L_i^* = \begin{cases} U_i, & \textit{if } \nu^* < \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{(n_i - n_{i-1})(U_i)^2}, \\ 1, & \textit{if } \nu^* > \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{n_i - n_{i-1}}, \\ \phi(\nu^*), & \textit{otherwise}, \end{cases} \tag{21}$$

where

$$U_i = \min \left\{ K_T, K \sum_{j=n_{i-1}+1}^{n_i} p_j \min \left\{ \frac{1}{\Lambda}, \frac{1-\gamma}{1+\Lambda\gamma} \right\} \right\},$$

where $\phi(\nu^*)$ is the solution over L_i of

$$-\frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}}^{n_i} p_j}{L_i^*} + \nu^* (n_i - n_{i-1}) = 0, \tag{22}$$

and where ν^* can be obtained as the unique solution to the additional constraint

$$n_1 + \sum_{i=2}^{Q} L_i^* (n_i - n_{i-1}) = K_T \gamma_T N. \tag{23}$$

[5]The constraint in (P1-f) is not necessary for convexity of $T^*(\mathbf{n}, Q)$ in L, and thus it is omitted in the above formulation.

Proof. The Lagrangian function corresponding to Problem 3 is given by

$$L(\mathbf{L}, \nu, \eta, \omega) \triangleq n_1 + \frac{K(1-\gamma)}{(1+\Lambda\gamma)} \sum_{i=2}^{Q} \frac{\sum_{j=n_{i-1}+1}^{n_i} p_j}{L_i} + \nu \left(n_1 + \sum_{i=2}^{Q} L_i (n_i - n_{i-1}) - K_T \gamma_T N \right) - \sum_{i=2}^{Q} \eta_i (-L_i + 1) + \sum_{i=2}^{Q} \omega_i (L_i - U_i), \tag{24}$$

where $\mathbf{L}$, $\boldsymbol{\eta}$, $\boldsymbol{\omega} \in \mathbb{R}_+^Q$ and $\nu \in \mathbb{R}$.

Let $\mathbf{L}^*$, $\boldsymbol{\eta}^*$, $\boldsymbol{\omega}^*$ and ν^* be primal and dual optimal. The KKT conditions for Problem 3 state that

$$n_1 + \sum_{i=2}^{Q} L_i^* (n_i - n_{i-1}) \le K_T \gamma_T N, \tag{25}$$
$$1 \le L_i^* \le U_i, \quad \forall i = 2, \ldots, Q, \tag{26}$$
$$\eta_i^* \ge 0, \quad \forall i = 2, \ldots, Q, \tag{27}$$
$$\omega_i^* \ge 0, \quad \forall i = 2, \ldots, Q, \tag{28}$$
$$\eta_i^* (-L_i^* + 1) = 0, \quad \forall i = 2, \ldots, Q, \tag{29}$$
$$\omega_i^* (L_i^* - U_i) = 0, \quad \forall i = 2, \ldots, Q, \tag{30}$$
$$-\frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}}^{n_i} p_j}{(L_i^*)^2} + \nu^* (n_i - n_{i-1}) - \eta_i^* + \omega_i^* = 0, \quad \forall i = 2, \ldots, Q, \tag{31}$$

and thus from (29), (30) and (31), we have

$$\omega_i^* = \frac{\eta_i^*}{U_i} - \frac{\nu (n_i - n_{i-1}) L_i^*}{U_i} + \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{L_i^*}, \tag{32}$$

which, when inserted into (30), gives

$$\left[\frac{\eta_i^*}{U_i} - \frac{\nu (n_i - n_{i-1}) L_i^*}{U_i} + \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{L_i^*} \right] \times (L_i^* - U_i) = 0. \tag{33}$$

From (33), we see that $0 < L_i^* < U_i$ holds only if

$$\nu^* = \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{(n_i - n_{i-1}) (L_i^*)^2}.$$

Since we know that $1 \le L_i^* \le U_i$, this implies that

$$\nu^* \in \left[\frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{(n_i - n_{i-1}) (U_i)^2}, \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{n_i - n_{i-1}} \right].$$

If now

$$\nu^* < \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{(n_i - n_{i-1}) (U_i)^2},$$

then we have $\omega_i^* > 0$. Thus, from (30), we have that $L_i^* = U_i$. Similarly, if

$$\nu^* > \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{j=n_{i-1}+1}^{n_i} p_j}{n_i - n_{i-1}},$$

we have that $\eta_i^* > 0$. Hence, from (29), we get that $L_i^* = 1$.

Finally, since $n_1 + \sum_{i=2}^{Q} L_i^* (n_i - n_{i-1}) = K_T \gamma_T N$ is a decreasing function in ν, solving $Q-1$ equations of (22) while satisfying (23), gives the unique solution ν^*. □

Remark 2. *It is easy to verify that the solution given in Theorem 1 always satisfies the constraint in* (P1-f)*.*

D. Step 3: Search algorithm for $\mathbf{n}$

With $T^*(\mathbf{n}, Q)$ at hand from before, we will here design a search algorithm that adjusts the search space of $\mathbf{n}$ to compute $T^*(Q)$ in a non exhaustive manner. For fixed Q, let

$$T^*(Q) = \min_{\mathbf{n} \in [0,N]^{Q-1}} T^*(\mathbf{n}, Q) \tag{34}$$

be the optimal delay corresponding to the optimal

$$\mathbf{n}^*(Q) = \arg \min_{\mathbf{n} \in [0,N]^{Q-1}} T^*(\mathbf{n}, Q). \tag{35}$$

Instead of searching for $\mathbf{n} \in [0, N]^{Q-1}$, which would obviously have the worst case complexity, we consider a new search space $S_Q \subseteq (R_Q)^{Q-1}$ for $\mathbf{n}$, for some $R_Q \subseteq [0, N]$ to be designed. The detailed sketch of the algorithm that defines S_Q is given in Algorithm 1.

We briefly explain the intuition behind the algorithm. Since Algorithm 2 — presented in Section III-E — is recursive, it accepts as input Q as well as the optimal *'cut-off points'* $\mathbf{n}^*(Q-1)$ for the case of $Q-1$ sub-libraries. The stopping condition ensures finding the optimum $T^*(Q)$.

To compute the first $T^*(\mathbf{n}, Q)$, we need an $\mathbf{n}$. In the loop given in step 5, we first define the search range $S_1 \in [0, n_1^*(Q-1)]$, and the reasoning behind the boundaries will be explained in the proof of Theorem 2. We set $n_1 = \lfloor \frac{0+n_1^*(Q-1)}{2} \rfloor$, and fix it. Next, we define the search range $S_2 \in [n_1 + 1, n_2^*(Q-1)]$, we set $n_2 = \lfloor \frac{n_1+1+n_2^*(Q-1)}{2} \rfloor$, and fix it. We continue following steps 6 and 7 until fixing n_{Q-1} (noting that $n_Q = N$ for any Q). Now since we have the first $\mathbf{n}$, we compute $T^*(\mathbf{n}, Q)$ by solving Problem 3 in step 8. We update the search space S_{Q-1} by moving to step 10 or 14 depending on the sign of the discrete derivative of $T^*(\mathbf{n}, Q)$ with respect to n_{Q-1}, and continue updating its search space until obtaining the optimal $T^*(\mathbf{n}, Q)$ for fixed $n_1, \ldots, n_{Q-2}$. Having reached this point, we move to step 19, set $stop(Q-1) = 1$ and move to step 9 to take the discrete derivative of $T^*(\mathbf{n}, Q)$ with respect to n_{Q-2} and update the search space S_{Q-2} accordingly. The algorithm continues by setting a new n_{Q-2} as explained above, defining the new search space S_{Q-1} and continuing with the same procedure iteratively until finding the optimal n_1^*, or equivalently finding the optimal $\mathbf{n}^*$, hence the optimal $T^*(\mathbf{n}^*, Q)$, and thus the optimal $T^*(Q)$.

Theorem 2. *For any fixed Q, Algorithm 1 with the defined search range of the cut-off points, yields the optimal*

$$\begin{aligned} T^*(Q) &\triangleq \min_{\mathbf{n} \in [0,N]^Q} T^*(\mathbf{n}, Q) \\ &= \min_{\mathbf{n} \in S_Q \subseteq (R_Q)^{Q-1}} T^*(\mathbf{n}, Q), \end{aligned}$$

Algorithm 1: The search algorithm that defines S_Q for computing $T^*(Q)$

input : Number of sub-libraries Q and all the prior information for the optimal cut-off points $\mathbf{n}^*(Q-1)$ for the case of $Q-1$ sub-libraries.

output: The optimal average delivery time $T^*(Q)$, the optimal cut-off points $\mathbf{n}^*(Q)$, and the optimal redundancy $\mathbf{L}^*(\mathbf{n}^*(Q), Q)$.

1 Set $\mathbf{stop} = [0]^{Q-1}$;
2 **while** $\mathbf{stop} \neq [1]^{Q-1}$ **do**
3 Set $i = 1$;
4 Set $n_0 = -1$;
5 **for** $q \leftarrow i$ *to* $Q-1$ **do**
6 Set $S_q = [n_{q-1} + 1, n_q^*(Q-1)]$;
7 $n_q = \left\lfloor \frac{n_{q-1}+1+n_q^*(Q-1)}{2} \right\rfloor$;
8 Compute $T^*(\mathbf{n}, Q)$ by solving Problem 3;
9 Set j = q;
10 **if** $\Delta_{n_j} T^*(\mathbf{n}, Q) < 0$ **then**
11 Update $S_j = [n_j + 1, n_j^*(Q-1)]$;
12 Set $stop(k) = 0,\ \forall k \in [j+1, Q-1]$;
13 Set $q = j$ and go back to step 7;
14 **else if** $\Delta_{n_j} T^*(\mathbf{n}, Q) > 0$ **then**
15 Update $S_j = [n_{j-1} + 1, n_j - 1]$;
16 Set $stop(k) = 0,\ \forall k \in [j+1, Q-1]$;
17 Set $q = j$ and go back to step 7;
18 **else**
19 Set $stop(j) = 1$;
20 **if** $j = 1$ **then**
21 Go back to step 2 of this algorithm;
22 **else**
23 Set $q = j - 1$ and go back to step 9;

where S_Q *is described in Algorithm 1.*

Proof. For $Q = 2$, we can find the optimal $n_1^*(2)$, $L_2^*(n_1^*(2), 2)$ and $T^*(2)$ by searching for $n_1(2) \in [0, N]$, since it is guaranteed that a unique solution exists due to biconvexity of Problem 2. Moreover, we can do a logaritmic search instead of an exhaustive search again due to biconvexity, since $n_1(2) \in \{0, \ldots, N\} \subseteq [0, N]$. This will maximize the redundancy $L_2^*(n_1^*(2), 2)$ allocated to the second sub-library. Now, with $Q = 3$ sub-libraries, the maximum redundancy had already been allocated for all the files within the second sub-library (the second sub-library for $Q = 2$). Thus, if $n_1(3) \in [n_1^*(2) + 1, N]$, it is guaranteed that $T(3) > T^*(2)$ since the total capacity for the allocated antennas will be reduced for the remaining files in the second and third sub-libraries. This concludes that the optimal $n_1^*(3) \in [0, n_1^*(2)]$. Similar arguments hold for all elements in $\mathbf{n}^*(Q)$. Also, since $\mathbf{n} \in \{0, N\}^Q \subseteq [0, N]^Q$, we can perform a logaritmic search within each speficied range due to biconvexity. Thus, we conclude that it is sufficient to search for $\mathbf{n} \in S_Q$ to obtain $T^*(Q)$, where S_Q is presented in Algorithm 1. □

Theorem 3. *The complexity of Algorithm 1 is* $\mathcal{O}\left((\log_2 N)^{Q-1}\right)$.

Proof. We update n_q's sequentially. For fixed Q, we search for $n^*_{Q-1}(Q)$ for fixed $[n_1(Q), \ldots, n_{Q-2}(Q)]$ by updating S_{Q-1} using a binary search, which naturally has a logarithmic complexity. After obtaining $n^*_{Q-1}(Q)$ for fixed $[n_1(Q), \ldots, n_{Q-2}(Q)]$, we do a logarithmic search for $n_{Q-2}(Q)$ by updating S_{Q-2}, picking a new $n_{Q-2}(Q)$ and finding the new $n^*_{Q-1}(Q)$ once again with a logarithmic search. Once we find the optimum $n^*_{Q-2}(Q)$ and $n^*_{Q-1}(Q)$ for fixed $[n_1(Q), \ldots, n_{Q-3}(Q)]$, we run a logarithmic search for $n_{Q-3}(Q)$, and so forth. The algorithm stops when we find the optimal $n^*_1(Q)$ (and hence the optimal $\mathbf{n}^*(Q)$). Therefore, we run a concatenated logarithmic search in $Q-1$ levels, hence the complexity of Algorithm 1 is $\mathcal{O}\left((\log_2 N)^{Q-1}\right)$. □

Remark 3. *Obviously, this complexity is for the worst case. In practice, the complexity of Algorithm 1 is much smaller, because the upper boundary of* S_q, $\forall q \in \{1, \ldots, Q-1\}$ *is limited by* $\mathbf{n}^*(Q-1)$. *In Section IV, we will numerically illustrate this substantially reduced complexity.*

E. Step 4: The Main Algorithm

In this subsection, we will give the details of our main algorithm that yields the optimal $Q^*, \mathbf{n}^*, \mathbf{L}^*$, and eventually the optimal delay by optimizing across Q to get $T^* = \min_{Q\in[2,K_T]} T^*(Q)$.

Theorem 4. *For*

$$Q^* = \arg \min_{Q\in[2,K_T]} T^*(Q), \tag{36}$$

the globally optimum average delay takes the form $T^* = T(Q^*)$, *and it is achieved with* Q^* *sub-libraries, with boundaries* $\mathbf{n}^*(Q^*)$ *and redundancy* $\mathbf{L}^*(\mathbf{n}^*(Q^*), Q^*)$.

Proof. As Q increases, it is clear that the L_q values will start decreasing at some point because of the constraint of Eq. (P1-c), and in particular because the sum of the p_j's decreases as Q increases. This in turn will increase $T(\mathbf{n}, \mathbf{L}, Q)$. Hence, there exists an optimal number of sub-libraries Q^* that will give the global minimum for the delay $T(\mathbf{n}, \mathbf{L}, Q)$. □

Finally, we will present our main algorithm (Algorithm 2) that obtains the solution to the main Problem 1.

Algorithm 2: Main algorithm

```
Set T*(0) = ∞;
Set Q = 1 and compute T*(1) = min{N, T_L};
for Q ← 2 to K_T do
    while T*(Q) < T*(Q − 1) do
        Q = Q + 1;
        Compute T*(Q) by solving Problem 3 and
          using Algorithm 1 for the search range of n;
Q* = Q − 1;
T* = T*(Q*);
```

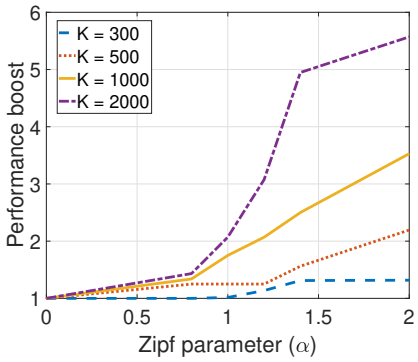

Fig. 1. The multiplicative boost in the performance of a system compared to the baseline with uniform file popularity. The comparison is displayed here for various Zipf parameters and for different K.

In brief, starting at $Q = 2$, the algorithm finds the optimal boundaries $\mathbf{n}^*(Q)$ and optimal redundancy $\mathbf{L}^*(\mathbf{n}^*(Q), Q)$, for each fixed Q, and then increases Q.

Theorem 5. *The complexity of Algorithm 2 is* $\mathcal{O}\left(K_T(\log_2 N)^{K_T-1}\right)$.

Proof. The above is direct from Theorem 3 and from the fact that there can be at most K_T many sub-libraries. □

IV. Numerical Evaluation

We consider the scenario with $\gamma = \gamma_T = 0.1$, with $K_T = 50$ transmitters, with $\Lambda = 40$, and with catalog size equal to $N = 6000$ files. We run the main algorithm given in Algorithm 2 to find the optimal number of sub-libraries, the optimal number of antennas allocated to each sub-library, and in the end the optimal delivery time for different numbers of users K and different Zipf parameters α. In Figure 1, we see the performance boost (T^*/T_L) for $K = 300, 500, 1000, 2000$ for different values of α between 0 and 2. We see that the performance boost increases with α (as expected), and we also see that the boost increases with K.

For the above setting, Table I describes the optimal sub-library boundaries $\mathbf{n}^*$ and the optimal $\mathbf{L}^*$. While an increasing K implies an increase in the optimal Q^*, it was the case that for all realistic scenarios that we have tested, this never exceeded $Q^* = 4$. Consequently the search space of $\mathbf{n}$ remained small for the first elements, and the complexity of Algorithm 1 was much smaller than the described worst-case complexity of $\mathcal{O}\left((\log_2 N)^{Q-1}\right)$. In fact, even without considering the acceleration effect of Algorithm 1, the overall complexity for this example scenario was equal to $\mathcal{O}\left(\log_2(6000) + (\log_2(6000))^2 + (\log_2(6000))^3\right) = \mathcal{O}(2147)$. This further reduced – after considering the effect of Algorithm 1 – to 105 iterations in the worst case.

V. Conclusion

In this work we addressed the problem of optimizing the number of transmitters each file is cached at, as a function of that file's popularity. By using the biconvexity property of the

	$K = 300$		$K = 500$		$K = 1000$		$K = 2000$	
α	$\mathbf{n}^*$	$\mathbf{L}^*$	$\mathbf{n}^*$	$\mathbf{L}^*$	$\mathbf{n}^*$	$\mathbf{L}^*$	$\mathbf{n}^*$	$\mathbf{L}^*$
0.8	$[0]$	$[5.0000]$	$[0, 1712]$	$[9.2158, 3.2842]$	$[0, 550]$	$[13.7520, 4.1168]$	$[0, 191, 1439]$	$[20.4944, 7.1939, 3.7507]$
1	$[1]$	$[5.0007]$	$[0, 1827]$	$[10.8980, 1.6020]$	$[2, 656]$	$[14.9947, 3.7783]$	$[0, 157, 1278]$	$[30.3803, 8.1446, 3.4096]$
1.2	$[1]$	$[5.0007]$	$[0, 22]$	$[7.7143, 4.7857]$	$[2, 663]$	$[14.8124, 2.5757]$	$[2, 345]$	$[27.5842, 3.6314]$
1.4	$[1]$	$[5.0007]$	$[3]$	$[5.0020]$	$[0, 43]$	$[21.0729, 3.9271]$	$[0, 233]$	$[46.6118, 3.3188]$
2	$[1]$	$[2.9401]$	$[1]$	$[4.9001]$	$[3]$	$[4.3115]$	$[5]$	$[5.0033]$

TABLE I

OPTIMAL SUB-LIBRARY BOUNDARIES $\mathbf{n}^*$ AND OPTIMAL ANTENNA ALLOCATIONS $\mathbf{L}^*$ FOR THE SIMULATED SETTING.

formulation, we designed a new search algorithm that obtains the globally optimal solution, achieving the minimum average delivery time, and doing so with a complexity that scales as a polynomial function of the logarithm of the size of the file catalog. In the end we showed that the optimal transmitter-side cache placement yields multiplicative speedup factors over traditional multi-transmitter coded caching algorithms. An interesting extension of this work would be to simultaneously optimize for both transmitter and receiver redundancy. Another research direction whose need is highlighted here is on the design of multi-transmitter coded caching algorithms that can exploit natural multicasting opportunities.

REFERENCES

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Inf. Theory*, vol. 60, pp. 2856–2867, May 2014.

[2] 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, pp. 647–663, Jan 2019.

[3] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *IEEE Inf. Theory Workshop (ITW)*, Sep. 2016.

[4] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Transactions on Information Theory*, vol. 62, pp. 5524–5537, Oct 2016.

[5] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Transactions on Information Theory*, vol. 63, pp. 5821–5833, Sep. 2017.

[6] L. Tang and A. Ramamoorthy, "Coded caching schemes with reduced subpacketization from linear block codes," *IEEE Transactions on Information Theory*, vol. 64, pp. 3099–3120, April 2018.

[7] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Transactions on Information Theory*, vol. 64, pp. 5755–5766, Aug 2018.

[8] E. Parrinello, A. Unsal, and P. Elia, "Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching," *IEEE Transactions on Information Theory*, pp. 1–1, 2019.

[9] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, March 2010.

[10] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013.

[11] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. on Sel. Areas in Comm.*, vol. 36, pp. 1111–1125, June 2018.

[12] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. on Inf. Theory*, vol. 59, Dec 2013.

[13] A. Tuholukova, G. Neglia, and T. Spyropoulos, "Optimal cache allocation for femto helpers with joint transmission capabilities," in *2017 IEEE Int. Conference on Comm. (ICC)*, pp. 1–7, May 2017.

[14] W. C. Ao and K. Psounis, "Distributed caching and small cell cooperation for fast content delivery," in *16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '15, (New York, NY, USA), pp. 127–136, Association for Computing Machinery, 2015.

[15] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. on Inf. Theory*, vol. 63, Feb 2017.

[16] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order optimal coded caching-aided multicast under zipf demand distributions," in *The Eleventh Int. Symp. on Wireless Comm. Systems (ISWCS)*, 2014.

[17] P. Quinton, S. Sahraei, and M. Gastpar, "A novel centralized strategy for coded caching with non-uniform demands," *arXiv preprint arXiv:1801.10563*, 2018.

[18] E. Ozfatura and D. Gunduz, "Uncoded caching and cross-level coded delivery for non-uniform file popularity," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[19] H. Ding and L. Ong, "An improved caching scheme for nonuniform demands and its optimal allocation," in *2017 3rd IEEE Int. Conf. on Comp. and Comm. (ICCC)*, pp. 389–393, Dec 2017.

[20] S. A. Saberali, L. Lampe, and I. Blake, "Full characterization of optimal uncoded placement for the structured clique cover delivery of nonuniform demands," *IEEE Trans. Inf. Theory*, pp. 1–1, 2019.

[21] C. Chang and C. Wang, "Coded caching with heterogeneous file demand sets — the insufficiency of selfish coded caching," in *2019 Int. Symp. on Inf. Theory (ISIT)*, pp. 1–5, July 2019.

[22] H. Al-Lawati, N. Ferdinandy, and S. C. Draperz, "Coded caching with non-identical user demands," in *2017 15th Canadian Workshop on Information Theory (CWIT)*, pp. 1–5, June 2017.

[23] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Transactions on Information Theory*, vol. 64, pp. 349–366, Jan 2018.

[24] Y. Deng and M. Dong, "Structure of optimal cache placement for coded caching with heterogeneous demands," *arXiv preprint arXiv:1912.01082*, 2019.

[25] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Transactions on Information Theory*, vol. 62, pp. 7253–7271, Dec 2016.

[26] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Transactions on Information Theory*, vol. 63, pp. 3092–3107, May 2017.

[27] J. Zhang, F. Engelmann, and P. Elia, "Coded caching for reducing csit-feedback in wireless communications," in *2015 53rd Annual Allerton Conf. on Comm., Control, and Comp. (Allerton)*, Sep. 2015.

[28] E. Piovano, H. Joudeh, and B. Clerckx, "On coded caching in the overloaded miso broadcast channel," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2795–2799, June 2017.

[29] J. Zhang and P. Elia, "Fundamental limits of cache-aided wireless BC: Interplay of coded-caching and CSIT feedback," *IEEE Transactions on Information Theory*, vol. 63, pp. 3142–3160, May 2017.

[30] E. Lampiris and P. Elia, "Resolving a feedback bottleneck of multi-antenna coded caching," *arXiv preprint arXiv:1811.03935*, 2018.

[31] E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," in *IEEE Int. Symp. on Inf. Theory (ISIT)*, June 2017.

[32] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 36, pp. 1176–1188, June 2018.

[33] M. E. J. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary Physics*, vol. 46, pp. 323–351, 2019.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, 2004.