

Multi-Transmitter Coded Caching Networks with Transmitter-side Knowledge of File Popularity

Berkas Serbetci, Eleftherios Lampiris, Thrasyvoulos Spyropoulos, Giuseppe Caire, Petros Elia

Abstract—This work presents a new way of exploiting non-uniform file popularity in coded caching networks. Focusing on a fully-connected fully-interfering wireless setting with multiple cache-enabled transmitters and receivers, we show how non-uniform file popularity can be used very efficiently to accelerate the impact of transmitter-side data redundancy on receiver-side coded caching. This approach is motivated by the recent discovery that, under any realistic file-size constraint, having content appear in multiple transmitters can in fact dramatically boost the speed-up factor attributed to coded caching.

We formulate an optimization problem that exploits file popularity to optimize the placement of files at the transmitters. Consequently, we propose a search algorithm that solves the problem at hand while reducing the variable search space significantly. We also prove an analytical performance upper bound, which is in fact met by our algorithm in the regime of many receivers. Our work reflects the benefits of allocating higher cache redundancy to more popular files, but also reflects a law of diminishing returns where for example very popular files may in fact benefit from minimum redundancy. In the end, this work reveals that in the context of coded caching, employing multiple transmitters can be a catalyst in fully exploiting file popularity, as it avoids various asymmetry complications that appear when file popularity is used to alter the receiver-side cache placement.

Index Terms—Coded caching, optimization, femtocaching.

I. INTRODUCTION

In the context of cache-aided, interference-limited communication networks, the work of Maddah-Ali and Niesen [2] revealed how content that is properly placed at the caches of the receivers, can be used as side information to cancel interference and reduce delivery time.

In particular, the work in [2] considers a single-antenna broadcast (downlink) configuration, where a transmitter has access to a library of N files, each of size F bits. The transmitter serves—via a unit-capacity bottleneck link—a set

B. Serbetci and P. Elia are with the Communication Systems Department of EURECOM, 06410 Sophia Antipolis, France, email: {serbetci, elia}@eurecom.fr.

T. Spyropoulos is with the Electrical and Computer Engineering Department of the Technical University of Crete, Greece, email: spyropoulos@tuc.gr. This work was partially performed while at EURECOM.

G. Caire is with the Communications and Information Theory Group (CommIT) of the Technical University of Berlin, 10587 Berlin, Germany, email: caire@tu-berlin.de.

E. Lampiris was previously with EURECOM and the Technical University of Berlin, email: lampiris@eurecom.fr.

The work is supported by French National Research Agency (ANR) under the “5C-for-5G” JCJC project with reference number ANR-17-CE25-0001, by the ERC project DUALITY (grant agreement no. 725929), and by the ERC project CARENET (grant agreement no. 789190). Parts of this work have been published in the 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2020) [1].

of K receiving users, each endowed with a cache of size $M \cdot F$ bits, corresponding to a fraction $\gamma \triangleq \frac{M}{N}$ of the library. The setting involves a *cache-placement phase* where the caches are filled with content in a manner oblivious to future demands, and then a subsequent *delivery phase* which starts with each user simultaneously demanding an independent file.

By exploiting *content redundancy* where each bit of data can be placed at $KM/N = K\gamma$ users, the algorithm in [2] could multicast different messages to $K\gamma + 1$ users at a time because each receiver could cancel the interference by accessing their own cache. This speedup factor of $K\gamma + 1$ is commonly referred to as the Degrees-of-Freedom (DoF) performance, and it implies a worst-case delivery time¹ equal to

$$T_{MN} = \frac{K(1-\gamma)}{1+K\gamma} \stackrel{K \rightarrow \infty}{=} \frac{1-\gamma}{\gamma}. \quad (1)$$

The above delay was shown in [3] to be within a multiplicative factor of at most 2.01 from the information-theoretic optimal, and to be exactly optimal over the class of schemes that employ uncoded cache placement [4], [5].

Subpacketization and the redundancy constraint: The above unbounded gain is in practice infeasible, mainly because it requires each file to be divided (subpacketized) into at least $\binom{K}{K\gamma}$ subfiles. Having files that do not scale exponentially in K , constitutes a prohibitive fundamental bottleneck [6], [7] which hard-bounds the DoF at very modest values².

A simple way to abide by the file-size constraint, is simply to assign the same cache content to entire groups of users (cf. [11]). With this number of groups Λ being constrained as

$$\binom{\Lambda}{\Lambda\gamma} \leq F, \quad (2)$$

the placement algorithm of [2] is used to create Λ different caches, and to assign the same cache to all the users belonging to the same group. Then, to satisfy the user demands, the delivery algorithm of [2] — which now enjoys a reduced DoF $\Lambda\gamma + 1$ — is repeated $\frac{K}{\Lambda}$ times, resulting in a delivery time of

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

Coded caching with transmitter-side cache redundancy:

As it turns out, the above subpacketization bottleneck is intimately connected, not only to the content redundancy $K\gamma$ at the receiver side, but also at the transmitter side. This connection was made in [12] which — in the context of

¹This is the normalized time that guarantees the successful delivery of all requested files, independent of the file-demand pattern.

²Some interesting progress on this, can be found in [6]–[10].

multiple transmitting nodes (see [13]–[16]) — employed a novel fusion of coded caching and multi-antenna precoding, to dramatically reduce the subpacketization requirements of coded caching, and in the process to show that having multiple transmitter-side redundancy can in fact multiplicatively boost the caching gain. In particular, for the coded caching scenario in [13] (see also [14]) where the K receivers are served by K_T transmitters each having access to a cache of normalized size $\gamma_T \in [\frac{1}{K_T}, 1]$, the work in [12] showed that for $L \triangleq K_T \gamma_T$, and under the subpacketization of (2) with $\Lambda \leq \frac{K}{L}$, then one can get a dramatically decreased delivery time of

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

which is optimal under the assumption of uncoded placement [17].

The above performance is achieved when each library file enjoys, on the transmitter side, an identical cache-redundancy equal to L , i.e. each file is cached at *exactly* L transmitters. In our current work here, we propose and explore the endowing of some (generally more popular) files, with higher redundancy than their less popular counterparts. As we will see, this approach will not only improve performance, but will also allow us to utilize file popularity without breaking the symmetry of coded caching, as can often be the case when file popularity is used to alter the placement at the receiver side. This will become clearer below when we recall some existing methods of utilizing this knowledge.

File popularity in coded caching, and the problem of symmetry: Before recalling how non-uniform file popularity has been used in coded-caching, let us quickly recall that exploiting file popularity has been a key concept from the early works of Content Delivery Network systems [18], [19], Content-Centric and Information-Centric Networks [20], [21], multi-tier networks [22], as well as in wireless edge caching works [23] that followed the femto-caching ideas of [24]. Such works generally focus on exploiting caches to ‘prefetch’ content, and have little to do with using caches to handle network interference. Even works that do consider PHY-aspects like multi-antenna beamforming, often assume that transmissions are, in essence, non-interfering [25], [26].

The connection between caching and interference management was mainly explored by works capitalizing on the interplay between coded caching and multiple transmitters, which initially focused on worst-case metrics, thus neglecting the effects of non-uniform file popularity. Recently a variety of works such as [27]–[36], explored different ways of exploiting this popularity in the single-stream coded caching model. As these efforts progressed, it was soon realized that incorporating file popularity in coded caching, brings to the fore a certain non-beneficial asymmetry which we discuss below.

In general, knowing the file popularity, would allow the grouping of similarly popular files, in order to allocate more cache space to popular files, thus leading to higher redundancy for more popular files and faster delivery. Given, though, the multicast nature of coded caching, this approach brings to the fore the dilemma of whether or not multicast delivery messages should combine content from files that are dissimilar

in terms of popularity. This is an important dilemma with serious ramifications. Choosing to not encode across different sub-libraries negates the very idea of coded caching, which benefits from encoding over as many users as possible. After all, the gains of coded caching are proportional to how many users/files one encodes over. Instead, here, not encoding across sub-libraries, forces algorithms to separately deliver one sub-library after the other, which is a time-consuming process. On the other hand, choosing to have popular and unpopular (sub) files coexist in a single transmission, can suffer from a certain asymmetry in the size of the composite subfiles. In principle, popular subfiles will tend to be longer than unpopular ones³. This can in turn force very substantial zero-padding, which implies that only a fraction of the delivered bits actually corresponds to real content.

Drawing from the first paradigm, different works [27]–[30] consider multicast messages (taken from [2]) which are composed of content from only one sub-library at a time. As was nicely shown in [28], this approach provides for a bounded gap to the information-theoretic optimal⁴. This gap in [28] was shown in [30] to vanish for the special case of $K = 2$.

Following the second paradigm, works such as [31]–[33] facilitate coding across sub-libraries, after optimizing the amount of cache each file can occupy as a function of its popularity. Interestingly, in some cases such as in [28], [33], the optimization suggests — under certain very important assumptions — the need for only a very small number of sub-libraries. A similar conclusion was drawn in [36] for a decentralized setting⁵. Another interesting decentralized approach can be found in [34] which combines a popularity-aware placement with a clique-cover delivery algorithm, to achieve — under the assumption of a Zipf distribution (cf. [38]) and in the limit of large K and large $K\gamma$ — an order optimal performance⁶. This performance was further improved in [35] which presented a delivery algorithm based on index coding (cf. [39]).

Current contribution: Boosting the impact of transmitter-side data redundancy using file popularity

The dramatic impact of transmitter-side cache redundancy in coded caching, together with the aforementioned problem of symmetry, are two main motivating factors of our work. Focusing on a setting with K_T cache-aided transmitters tasked with serving K cache-aided receiving users, we explore the effect of allowing different files to experience different transmitter-side redundancies, depending on their popularity. In the context of coded caching, this constitutes a novel approach that allows us to benefit from a non-uniform file popularity, while having receivers that are agnostic to this popularity.

³This goes back to having designated more cache space for popular files, which often implies that the multicast messages will carry popular subfiles that are larger than their unpopular counterparts.

⁴The multiplicative gap is approximately 50. Naturally the metric is the average delivery time, averaged over all demands.

⁵Centralized coded caching [2] assumes that the identity of the users is known during the placement phase and provides a deterministic caching strategy. On the other hand, decentralized coded caching [37] assumes that the identity of the users is unknown during the placement and each chunk of a file is cached with a specified probability.

⁶This means that, in the limit of infinite K , the gap to optimal is finite.

The main objective is to optimally divide the library into an arbitrary number of non-overlapping sub-libraries, and then to optimize the number of transmitters allocated to the files of any given sub-library. Our road map to solve the problem is as follows:

Step 1: We will formulate an optimization problem that exploits file popularity to optimize the placement of files at the transmitters, while having receivers that are agnostic to this popularity.

Step 2: We will show that finding the optimal solution to this placement proves to be a hard optimization problem.

Step 3: We will propose a search algorithm—together with a proof that reduces significantly the variable search space—that solves the problem at hand. Our proposed algorithm has complexity $O\left(\log_2 \frac{N}{Q}\right)^Q$, where N is the number of files and Q is the number of sub-libraries.

Step 4: We will provide an upper bound on the number of sub-libraries that yields a feasible solution, and show that a feasible solution will never yield having more than a certain number of sub-libraries, i.e., Q is very small for realistic system parameters. As our algorithm is guaranteed to yield a feasible solution, this upper bound will also hold for any solution that our algorithm provides, hence ensuring our algorithm to deal with a large range of parameters while enjoying its polynomial complexity.

Step 5: We will provide an analytical performance upper bound, which is in fact met by our algorithm in the regime of many receivers.

By solving this problem, we offer a multiplicative performance boost compared to the uniform popularity scenario (cf. (4)), as well as a number of additional significant advantages compared to the state of art.

- A first advantage is that the receiver-side placement remains agnostic to file popularity. This allows the network to easily adapt to possible changes in file popularity, because updating the caches of a modest number of centralized transmitters is much easier than doing so for a large number of distributed receivers.

- Additionally, as we discussed earlier, popularity-aware receiver-side caching requires i) an accurate knowledge of the users that will be active during the placement phase and, ii) creates sub-file asymmetries which reduce the resulting gains.

- Finally, the adopted receiver-side placement strategy does not require the identity of the users to be known during the placement phase.

The work provides interesting insights. While it is beneficial to allocate higher cache redundancy to popular files (so that the majority of requests experience higher DoF performance), this has to be done with caution because after a certain point a law of diminishing returns kicks in. This is particularly true for very popular files, where—as we will see—a minimum redundancy is beneficial.

In the end, a key ingredient in our work is the fact that files do not have unbounded sizes. This may seem like an esoteric detail, but is in fact at the core of many coded caching problems. In our particular problem, having finite file sizes is what makes the impact of transmitter-side redundancy

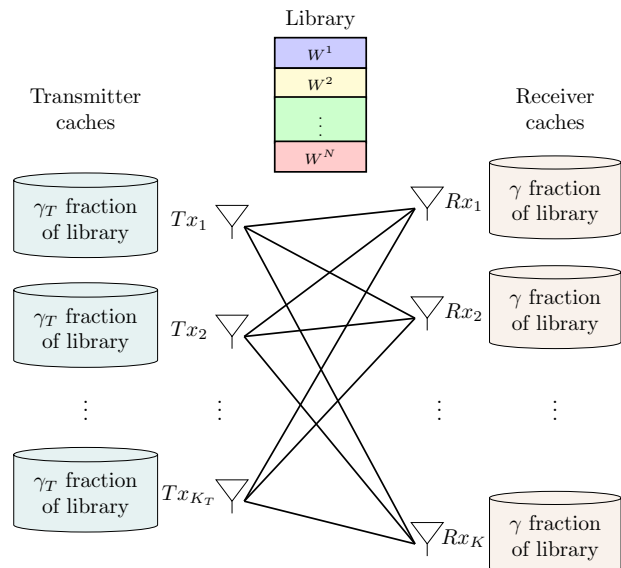


Fig. 1. Wireless network with K_T transmitters and K receivers, where each transmitter and each receiver caches up to γ_T fraction of the library and γ fraction of the library respectively, from a library of N files with different popularity.

so powerful, and thus what motivates us to optimize this redundancy.

Paper outline: In Section II we present the system model and the notation. In Section III we discuss the caching and delivery algorithms as well as the optimization problem that we seek to solve. Further, in Section IV we first provide a proof that reduces significantly the variable search space and then we describe an algorithm that solves the optimization problem. In Section V we calculate a theoretical limit to the performance of our setting, while we prove that the reduced variable search space has the added benefit of providing an increased performance under any choice of variables. Finally, in Section VI we evaluate numerically the algorithm by plotting the multiplicative performance increase, compared to the uniform popularity case, as a function of the Zipf parameter α and for various number of users K .

II. SYSTEM MODEL & NOTATION

We consider the fully-connected, K_T -transmitter cache-aided setting, as depicted in Figure 1, where K_T single-antenna transmitters serve K single-antenna receivers. 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, \dots, W^N , and that each file has size F bits⁷ and is of finite size. We assume that the system operates in the high Signal-to-Noise-Ratio region and 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 one⁸.

⁷This assumption is common in the literature, as non-equally sized files can be handled by making a content chunk the basic caching unit, as in [24].

⁸This requirement holds true in many wireless settings, as well as in wired settings with network-coding capabilities at the intermediate network nodes.

The caches of the transmitters and the receivers are filled with content during the placement phase. During the delivery phase, each user will concurrently request a single file, and we assume that these requests follow a file popularity distribution that is known during the cache placement. In particular, we will focus on file popularity that follows the Zipf distribution [38] with parameter $\alpha > 0$, under which the probability that file W^j is requested, takes the form

$$p_j = \frac{n^{-\alpha}}{\sum_{k=1}^N k^{-\alpha}}, \quad \forall j \in \{1, \dots, N\}. \quad (5)$$

Notation: Symbols $\mathbb{N}, \mathbb{R}, \mathbb{C}$ denote the sets of natural, real and complex numbers, respectively. For $n, k \in \mathbb{N}$, $n \geq k$, we denote the binomial coefficient with $\binom{n}{k}$, while $[k]$ denotes the set $\{1, 2, \dots, k\}$. We use $|\cdot|$ to denote the cardinality of a set. Bold letters are reserved for vectors, while for some vector \mathbf{h} , comprised of Q elements, we denote its elements as h_q , $q \in [Q]$, i.e., $\mathbf{h}^T \triangleq [h_1, h_2, \dots, h_Q]$.

III. CACHING AND DELIVERY POLICIES & MAIN PROBLEM

As suggested above, the caching policy at the transmitter-side is popularity-aware, while the receiver-side placement is not. We begin with the general description of the transmitter-side caching policy and further describe the placement policy at the caches of the receivers. Given these, we continue with the delivery algorithm, which is based on the algorithm of [12]. The last part of this section is dedicated to the presentation of the optimization problem that assigns content to the caches of the transmitters.

A. Caching and delivery policies

1) *Transmitter-side caching policy:* We segment the library into Q non-overlapping sub-libraries. Such segmentation is described via sets $\mathcal{B}_q \subset [N]$, $q \in [Q]$, and signifies that all files belonging to the same library would be assigned the same transmitter-side cache redundancy $L_q \in [1, K_T]$. In other words, each file of sub-library \mathcal{B}_q will be stored at exactly L_q different transmitters. As a consequence, variable L_q is restricted to be in the range $[1, K_T]$. On one end, we force each file to be cached by at least 1 transmitter, hence allowing any request pattern to be satisfied in a finite time. On the other end, the number of transmitters that can store a file is, naturally, limited by the number of different transmitters. The above-described cache-redundancies need to satisfy the collective transmitter side cache-constraint

$$\sum_{q=1}^Q |\mathcal{B}_q| \cdot L_q \leq N \cdot L, \quad (6)$$

where for simplicity we use herein $L \triangleq K_T \gamma_T$.

In addition to the collective cache-constraint of (6), the transmitter-side placement algorithm also needs to satisfy the per-transmitter cache-constraint of our model. In Appendix I we propose an explicit algorithm which, for arbitrary Q, \mathcal{B} and \mathbf{L} satisfying (6), provides a placement which is based solely on the constraint in (6), while also satisfying the individual cache constraint.

Parameters	Description
N	Number of different files
K	Number of users
γ	Fraction of library each user can store
Λ	Number of caches with different content
K_T	Number of single-antenna transmitters
γ_T	Fraction of library each transmitter can store
j	File index
p_j	Probability that file W^j will be requested
α	Zipf parameter
Q	Number of sub-libraries
\mathcal{B}_q	Content of sub-library q
\mathbf{n}	Vector storing the boundaries of the sub-libraries
L_q	Number of transmitters caching file W^j , $\forall j \in \mathcal{B}_q$
\mathbf{L}	Vector storing L_q
K_q	Number of users requesting a file from \mathcal{B}_q
\bar{K}_q	Expected number of users requesting a file from \mathcal{B}_q
$T(Q, \mathbf{n}, \mathbf{L})$	Delay of expected requests as a function of $Q, \mathbf{n}, \mathbf{L}$
T^*	Min. expected delay optimized over all variables
T_Q^*	Min. expected delay optimized over \mathbf{n}, \mathbf{L} . Fixed Q
$T_{Q, \mathbf{n}}^*$	Min. expected delay optimized over \mathbf{L} . Fixed Q, \mathbf{n}
S_Q	Problem search space
π_q	Sum probability of sub-library \mathcal{B}_q

TABLE I
NOTATION SUMMARY

2) *Receiver-side caching policy:* The receivers cache using a modified version of the algorithm of [2]. Specifically, we create a set of $\Lambda < K$ different caches and assign one to each user in a round-robin manner. Variable Λ is chosen such that $\Lambda\gamma$ is an integer and the subpacketization constraint is satisfied, i.e. $\binom{\Lambda}{\Lambda\gamma} \leq F$. Each file W^j , $j \in [N]$, is split into $\binom{\Lambda}{\Lambda\gamma}$ equally-sized subfiles

$$W^j \rightarrow \{W_\tau^j, \tau \subset [\Lambda], |\tau| = \Lambda\gamma\}, \quad (7)$$

thus, each subfile has as index some set τ , which is a $\Lambda\gamma$ -sized subset of set $[\Lambda]$. Then, the ℓ^{th} cache takes the form

$$\mathcal{Z}_\ell = \{W_\tau^j : \ell \in \tau, \forall j \in [N]\}, \quad \forall \ell \in \Lambda, \quad (8)$$

which simply means that cache ℓ consists of all subfiles W_τ^j , whose index τ contains ℓ . The round-robin manner of assigning caches to users results in an approximate $\frac{K}{\Lambda}$ users to be assigned the same exact content.

Example 1. Let us assume a setting comprised of $K = 50$ users, each equipped with a cache of normalized size $\gamma = \frac{1}{10}$, and which users are divided into $\Lambda = 10$ groups. For example, such grouping would yield Group 1 as $\mathcal{G}_1 = \{1, 11, \dots, 41\}$, Group 2 as $\mathcal{G}_2 = \{2, 12, \dots, 42\}$ and so on.

In the placement phase the files are divided into $\binom{\Lambda}{\Lambda\gamma} = 10$ subfiles as $W^j \rightarrow \{W_1^j, W_2^j, \dots, W_{10}^j\}$, $\forall j \in [N]$. Then, the contents of each cache would be

$$\mathcal{Z}_1 = \{W_1^1, \dots, W_1^j\}, \dots, \mathcal{Z}_{10} = \{W_{10}^1, \dots, W_{10}^j\}.$$

In the final step, each user of \mathcal{G}_1 is assigned cache \mathcal{Z}_1 , each user of Group 2 is assigned \mathcal{Z}_2 and so on.

3) *Content delivery policy:* The delivery phase begins with the concurrent request of any single file from each user. The fulfilment of these requests happens in a per-sub-library manner. Specifically, for each set of K_q users, requesting files from sub-library \mathcal{B}_q , we employ the algorithm of [12].

B. Main optimization problem - Placement at the transmitters

Having described the caching policy at the users and the subsequent delivery policy it remains to design the caches of the transmitters such as to reduce the delivery time. To this end, we need to

- select the number of sub-libraries Q ,
- segment the library into $\mathcal{B}_q \subset [N]$, $q \in [Q]$, and
- associate a cache redundancy L_q with each \mathcal{B}_q .

Since the request pattern is of a stochastic nature we will focus on minimizing the delivery time of the expected demand. In other words, we assume that the number of users requesting a file from sub-library \mathcal{B}_q is $K_q = \bar{K}_q = K\pi_q$, where we denote the cumulative probability of the files of sub-library \mathcal{B}_q , $q \in [Q]$ by $\pi_q \triangleq \sum_{k \in \mathcal{B}_q} p_k$.

Taking the above into account, the delivery time for each sub-library takes the form

$$T_q = \frac{K_q(1-\gamma)}{\min\{L_q(1+\Lambda\gamma), K_q\}}, \quad q \in [Q], \quad (9)$$

where the minimum in the denominator describes that the number of users served in a given time-slot is upper bounded by the number of available users.

In addition, in order to magnify the impact of transmitter-side cache redundancy, we impose a further constraint on the value of L_q . Specifically, we force $L_q \leq \frac{K_q}{\Lambda}$, which ensures that the achieved DoF is always a multiple of L_q , i.e. takes the form $L_q(\Lambda\gamma + 1)$ for any value of L_q (cf. [12]). Beyond this value of L_q the best known results achieve only an additive gain i.e., increasing the DoF by 1 for each increase of L_q by 1, while negatively affecting the subpacketization [40].

Remark 1. *While treating demands in a per sub-library manner is not necessarily optimal we note that, at the time of this writing, no known delivery algorithm can merge demands from multiple libraries in a single transmission. In particular, to date, no known multi-transmitter coded caching algorithm can improve the current performance we achieve, by simultaneously transmitting files that have different transmitter-side redundancy. We believe this to be an interesting open problem.*

Combining the delivery time of each sub-library we get the achievable delay of

$$T = \sum_{q=1}^Q \frac{K\pi_q(1-\gamma)}{\min\{L_q(1+\Lambda\gamma), K\pi_q\}}. \quad (10)$$

We can further improve the delivery time of (10) by considering that a set of ultra popular files may be requested by a significant amount of users, hence these files can be naturally multicasted from a single antenna, i.e. to be communicated sequentially and without employing coded caching techniques. This would allow to serve a significant number of users with minimal transmitter-side resources, since storing each file at a single transmitter would suffice to satisfy such demands. We place these files in sub-library \mathcal{B}_1 , while noting that this additional (natural multicasting) option does not limit the optimization range because \mathcal{B}_1 could be—if indicated by the optimization—empty. Consequently, the cache redundancy assigned to this sub-library is $L_1 = 1$, and the respective

delay is $T_1 = |\mathcal{B}_1|$ and corresponds to broadcasting the whole content of the sub-library.

Combining the above-described delivery delays of each sub-library, and for simplicity refraining from displaying the minimum function, the overall delay achieved takes the form

$$T(Q, \mathcal{B}, \mathbf{L}) = |\mathcal{B}_1| + \sum_{q=2}^Q \frac{K\pi_q(1-\gamma)}{L_q(1+\Lambda\gamma)}. \quad (11)$$

Because (11) is linearly dependent on the number of users requesting a file from each sub-library, we can conclude that the expected delay is equal to the delay of the expected demand, i.e. $\bar{K}_q = K\pi_q$.

Thus, the optimization problem at hand is expressed as

Problem 1 (General Optimization Problem).

$$\min_{Q, \mathcal{B}, \mathbf{L}} \mathbb{E}\{T(Q, \mathcal{B}, \mathbf{L})\} \quad (\text{P1-a})$$

$$\text{s.t. } Q \in [N], \quad (\text{P1-b})$$

$$|\mathcal{B}_1| + \sum_{q=2}^Q L_q |\mathcal{B}_q| \leq LN, \quad (\text{P1-c})$$

$$L_q \in [1, U_q], \quad \forall q \in [Q]. \quad (\text{P1-d})$$

where $U_q \triangleq \min\{K_T, K\pi_q/\Lambda\}$.

IV. DESCRIPTION OF THE OPTIMIZATION ALGORITHM

Retrieving the optimal solution of Problem 1 requires optimizing variables $Q, \mathcal{B}, \mathbf{L}$. The main difficulty we face is that the complexity increases exponentially for non-trivial values of Q . This high complexity is attributed to the need to segment set $[N]$ into Q non-overlapping subsets, which have the property of minimizing the problem at hand. For example, for $Q = 2$ the search space for \mathcal{B} has size 2^N , due to the need to consider every possible subset size for sub-library \mathcal{B}_1 , i.e.

$$\binom{N}{1} + \binom{N}{2} + \dots + \binom{N}{N} = 2^N. \quad (12)$$

In the general case, the size of the search space is exponential in N , as we show in the following proposition.

Proposition 1. *The size of the search space for determining sub-libraries \mathcal{B} in Problem 1, takes the form*

$$|S_1(Q)| = Q^N, \quad Q \in [N]. \quad (13)$$

Proof. The size of the search space is equivalent to the number of ways in which N objects can be partitioned into Q different subsets. We omit the straightforward proof. \square

We employ the following steps to solve Problem 1.

- 1) In Lemma 1 (Section IV-A) we prove that the optimal solution of Problem 1 should be of the form $\mathcal{B}_q = \{n_{q-1}+1, \dots, n_q\}$, $\forall q \in [Q]$, where $n_0 = 0$ and $n_Q = N$. In other words, the first sub-library should be comprised of the n_1 most popular files, the second sub-library would contain files $\{n_1 + 1, \dots, n_2\}$, and so on. From this point on we refer to a library segmentation using vector

$$\mathbf{n} \triangleq \{n_1, \dots, n_Q = N\}. \quad (14)$$

We can easily deduce the size of the reduced search space

$$|S_2(Q)| = \binom{N}{Q} \approx \left(\frac{N}{Q}\right)^Q, \quad (15)$$

which considerably prunes the search space from exponential in N to polynomial in N , without sacrificing optimality.

- 2) We provide an algorithm that searches $S_2(Q)$ requiring complexity at most $\left(\log_2 \frac{N}{Q}\right)^Q$.
- 3) We reformulate the objective function as a set of nested problems, as follows

$$\min_{Q(\mathbf{n}^*, \mathbf{L}^*)} \min_{\mathbf{n}(\mathbf{L}^*)} \min_{\mathbf{L}} \mathbb{E}\{T(Q, \mathbf{n}, \mathbf{L})\},$$

which effectively means that for each search of the outmost variables Q and \mathbf{n} we optimize the innermost variables $\mathbf{n}(\mathbf{L})$ and \mathbf{L} , respectively hence, maintaining the optimality of the solution [41].

- 4) As we show in Section IV-B, calculating the optimal \mathbf{L} can be achieved via the use of the Karush-Kuhn-Tucker (KKT) conditions. In other words, the innermost problem has an analytical solution, conditional on the values of Q and \mathbf{n} , which can be used directly for the outer optimization of these variables. Furthermore, the continuous relaxation of \mathbf{L} required by the application of the KKT conditions would result in a small performance degradation⁹, which we show in Lemma 4 is at most 12%.
- 5) Finally, we prove that the objective function, when optimized over both \mathbf{L} and \mathbf{n} is monotonically decreasing when $Q \in [1, Q^*]$ and monotonically increasing for $Q \in [Q^*, N]$. Thus, the function has a single minimum point which we calculate using a bisection algorithm.

A. Reduced sub-library search space

In order to show the optimality of the solution when considering the reduced sub-space in (14)-(15), we begin with a corollary that describes the relationship between the cache-allocation among any two arbitrary sub-libraries.

Corollary 1. *For two arbitrary sub-libraries $\mathcal{B}_q, \mathcal{B}_r \subset [N]$ for which $\pi_q > \pi_r$, their respective optimal cache-redundancy allocations satisfy $L_q^* > L_r^*$.*

Proof. The proof is relegated to Appendix II-A. \square

With this in place, we proceed with the lemma that establishes the optimality of the consecutively indexed library segmentation.

Lemma 1. *For arbitrary number of sub-libraries Q , the sub-libraries producing the optimal delay are those whose files have consecutive indices.*

Proof. The proof is relegated to Appendix II-B. \square

⁹The performance boost offered is already hundreds of percent even for modest system parameters.

Consequently, using Lemma 1 we can simplify the objective function as

$$\mathbb{E}\{T(Q, \mathbf{n}, \mathbf{L})\} = n_1 + \sum_{q=2}^Q \frac{\bar{K}_q(1-\gamma)}{L_q(1+\Lambda\gamma)}, \quad (16)$$

and the optimization problem takes the following form.

Problem 2 (Main Optimization Problem).

$$\min_{Q, \mathbf{n}, \mathbf{L}} \mathbb{E}\{T(Q, \mathbf{n}, \mathbf{L})\} \quad (\text{P2-a})$$

$$\text{s.t. } Q \in [N], \quad (\text{P2-b})$$

$$n_1 + \sum_{q=2}^Q L_q(n_q - n_{q-1}) \leq LN, \quad (\text{P2-c})$$

$$L_q \in [1, U_q], \quad \forall q \in [Q]. \quad (\text{P2-d})$$

The constraints of Problem 2 are those of Problem 1, with the notable difference being constraint (P2-c) which substitutes (P1-c), to yield a substantially reduced search space without loss of optimality.

As we discuss in Section V, the library segmentation of Problem 2 has an added benefit, on top of reducing the search space, compared to the general library segmentation of Problem 1. We show that *any* library segmentation as the one proposed in Problem 2, and under the optimal allocation of cache-redundancies L_q , would outperform the uniform popularity setting. On the other hand, the general library segmentation of Problem 1 does not share this property (see discussion in Corollary 2 and Remark 6).

B. Optimizing cache redundancies L_q

We begin this section with the following lemma.

Lemma 2. *The objective function is convex in variables \mathbf{L} for fixed Q and \mathbf{n} .*

Proof. The proof is relegated to Appendix II-C. \square

Hence, applying the KKT condition would provide the optimal vector \mathbf{L} . The Lagrangian takes the form

$$\begin{aligned} \mathcal{L} = & n_1 + \sum_{q=2}^Q \frac{K\pi_q(1-\gamma)}{L_q(1+\Lambda\gamma)} + \lambda \left(n_1 + \sum_{q=2}^Q L_q(n_q - n_{q-1}) - LN \right) \\ & + \sum_{q=2}^Q \mu_q(-L_q + 1) + \sum_{q=2}^Q \nu_q(L_q - U_q), \end{aligned} \quad (17)$$

where $L_q, \mu_q, \nu_q \geq 0, \forall q \in [Q]$ and $\lambda \in \mathbb{R}$.

Lemma 3. *The optimal cache-allocation vector \mathbf{L} for fixed Q, \mathbf{n} is given by*

$$L_q = \begin{cases} 1, & q \in \phi \cup \{1\} \\ U_q, & q \in \psi \\ \sqrt{\frac{\pi_q}{n_q - n_{q-1}} \frac{LN - n_1 - \Phi_S - \Psi_S}{\sum_{r \in \chi} \sqrt{\pi_r(n_r - n_{r-1})}}}, & q \in \chi \end{cases} \quad (18)$$

where $\Phi_S = \sum_{q \in \phi} (n_q - n_{q-1})$, $\Psi_S = \sum_{q \in \psi} U_q \cdot (n_q - n_{q-1})$, and $\phi \cup \chi \cup \psi \cup \{1\} = [Q]$.

Proof. The proof is relegated to Appendix II-D. \square

Theorem 1. *The expected delay optimized over \mathbf{L} takes the form*

$$T^*(Q, \mathbf{n}) = n_1 + \frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{q \in \phi} \pi_q + |\psi| \frac{\Lambda(1-\gamma)}{1+\Lambda\gamma} + \frac{K(1-\gamma)}{1+\Lambda\gamma} \frac{\left(\sum_{q \in \chi} \sqrt{\pi_q(n_q - n_{q-1})}\right)^2}{LN - n_1 - \Phi_S - \Psi_S}. \quad (19)$$

Proof. The proof is direct by inserting the calculated values L_q from (18) into the expression of the expected delay (16). \square

Lemma 4. *The continuous relaxation of \mathbf{L} requires the use of memory sharing (cf. [2]). This would result in a performance loss that is bounded by a multiplicative factor of 1.12.*

Proof. The optimal solution provided by Lemma 3 may produce non-integer L_q . In order for the algorithm of [12] to handle such non-integer values, we apply memory sharing as in [2]. Specifically, each file with a non-integer cache-redundancy L_q would be split into two parts, one part is cached with redundancy $\lceil L_q \rceil$, and the other part with $\lfloor L_q \rfloor$. If we denote with $p \in [0, 1]$ the fraction of the file stored with redundancy $\lceil L_q \rceil$ we can calculate its value through

$$p \lceil L_q \rceil + (1-p) \lfloor L_q \rfloor = L_q. \quad (20)$$

The memory sharing approach invariably results in some loss in performance, but as we show promptly it remains small.

Assuming that the target non-integer cache redundancy of sub-library \mathcal{B}_q is $L_q + r$, $r < 1$, i.e., $p = r$ by (20), focusing on the performance loss between the theoretical (non-integer) L_q compared to the one achieved by memory sharing, we have

$$\frac{\frac{p}{L_q+1} + \frac{1-p}{L_q}}{\frac{1}{L_q}} = 1 + \frac{r(1-r)}{L_q(L_q+1)}. \quad (21)$$

We can see that the biggest gap in (21) occurs when $r = \frac{1}{2}$. It follows that the maximum difference between the delivery time achieved without memory sharing and after we apply the technique would be for $\lfloor L_q \rfloor = 1$ amounting to $< 12\%$, while for $\lfloor L_q \rfloor = 2$ this would be $< 4\%$. Similar calculations show that for sub-libraries with even higher L_q the performance loss due to memory sharing becomes negligible.

Taking into consideration that only one sub-library can have cache-redundancy $\lfloor L_q \rfloor = 1$, it follows that the overall loss due to memory sharing is strictly less than 12% ¹⁰. \square

Remark 2. *Equation (19) can be simplified when $\phi = \psi = \emptyset$ to the following*

$$T^*(Q, \mathbf{n}) = n_1 + \frac{K(1-\gamma)}{1+\Lambda\gamma} \frac{\left(\sum_{q=2}^Q \sqrt{\pi_q(n_q - n_{q-1})}\right)^2}{LN - n_1}. \quad (22)$$

¹⁰Note that the memory sharing needs to be applied only at the very last step, hence this bound for the loss is valid for the final overall average delay.

C. Optimizing \mathbf{n}

Using the objective function in (16), i.e. optimized over variables \mathbf{L} , we can proceed to minimize it with respect to \mathbf{n} for some instance of Q . To this end, we propose an algorithm (Algorithm 1), which recursively optimizes each of the elements of \mathbf{n} .

Remark 3. *Numerical evaluation of (16) suggests that it is discrete convex [42]. This suggests that applying Algorithm 1 yields the optimal result. We defer the formal proof of this statement to future work, due to considerable technical difficulty.*

Algorithm 1: update(n_q)

Input: $n_1, n_2, \dots, n_{q-1}, Q$
1 Initialize: $S_q = \{n_{q-1} + 1, N + q - Q\}$ (Search space)
2 while $S_q(1) \neq S_q(2)$ **do**
3 (Calculate delay using first search space point)
 $n_q = S_q(1)$
 $\mathbf{n}^*(q+1:Q) = \text{update}(n_{q+1})$
 $T_{S_q(1)} = T(n_1, \dots, n_{q-1}, S_q(1), \mathbf{n}^*(q+1:Q))$
4 (Calculate delay using second search space point)
 $n_q = S_q(2)$
 $\mathbf{n}^*(q+1:Q) = \text{update}(n_{q+1})$
 $T_{S_q(2)} = T(n_1, \dots, n_{q-1}, S_q(2), \mathbf{n}^*(q+1:Q))$
5 (Update search space with mid-point)
 $s_a = \text{round}\left(\frac{S_q(1) + S_q(2)}{2}\right)$
 $s_b = \arg \min_{s \in S_q} T_s$
 $S_q = \{\min\{s_a, s_b\}, \max\{s_a, s_b\}\}$ (23)
Output: $\mathbf{n}(q:Q) = \{S_q(1), n_{q+1}^*, \dots, n_Q^*\}$

a) *Intuition behind the algorithm:* The main idea behind our algorithm is based on the observation that one can easily compare the delivery time achieved by two vectors $\mathbf{n}(1)$ and $\mathbf{n}(2)$, where i) the first $q-1$ elements of these vectors are the same, ii) they differ in the q -th element, and iii) the remaining $Q-q$ elements are chosen such that the delivery time is minimized, given the first $Q-q$ elements. In other words, we are interested in comparing the following vectors

$$\mathbf{n}(1) = \{n_1, \dots, n_{q-1}, n_q(1), n_{q+1}^*(1), \dots, n_Q^*(1)\},$$

$$\mathbf{n}(2) = \{n_1, \dots, n_{q-1}, n_q(2), n_{q+1}^*(2), \dots, n_Q^*(2)\},$$

where n_r^* denotes the r -th element that, conditioned on all previous elements, produces the lowest delivery time.

Hence, by fixing the first $q-1$ elements and, at the same time, for each value of n_q having access to the values of elements $\{q+1, \dots, Q\}$ that produce the lowest delivery time, we can apply the bisection algorithm to optimize the value

of n_q , by searching the discrete space between $n_{q-1} + 1$ and $N - Q + q$.

b) *Explanation of algorithm:* Our algorithm consists of a single recursive function that begins from the maximum search space for n_1 , i.e. points 0 and $N - Q$.

Initially, the algorithm creates a *While* loop which stops when the search space is reduced to a single element. Inside the *While* loop, the algorithm sets n_1 equal to the lower boundary of the search space and proceeds to calculate the optimal remaining $Q - 1$ elements. To achieve this, it recursively calls **update**(n_2).

In the same spirit, **update**(n_2) starts searching for the optimal n_2 , conditioned on the value of n_1 that is given as input. To this end, the algorithm sets $n_2 = n_1 + 1$ and recursively calls **update**(n_3). The recursive call of function **update** continues in the same manner until the last element, n_Q , is reached. At this point, since all previous elements are set (elements 1, ..., $Q - 1$) the algorithm can perform a bisection in the discrete space and produce the optimal n_Q .

The bisection procedure for n_Q , given fixed n_1, \dots, n_{Q-1} , is done by calculating the delivery time achieved using the lower boundary point (Step 3), and then by calculating the delivery time achieved by the highest boundary point (Step 4). Then, the boundaries of the new search space would include the boundary of the previous search space that produced the smallest delay as well as the middle point of the old boundary.

When the optimal n_Q is produced, the algorithm returns that value to **update**(n_{Q-1}), which continues with the calculation of the delay for point n_{Q-1} . Further, the algorithm seeks to calculate the delivery time when n_{Q-1} is equal to the other boundary of its search space. Similarly to before, the algorithm needs to first optimize n_Q , and as a result calls **update**(n_Q). After this operation has produced the optimal n_Q the algorithm calculates the delivery time corresponding to the higher boundary point of search space S_{Q-1} and now is able to update the boundaries of the search space. The new boundaries of the search space are the middle point of the old search space and the boundary of the old search space which has produced the lowest delivery time. Due to the convexity of each point n_q , given that all previous points are the same, and that all following points are optimized, we can conclude that the new search space is reducing the delivery time.

Theorem 2. *The worst-case complexity of Algorithm 1 is polynomial in N , and specifically is upper bounded by $\left(\log_2 \frac{N}{Q}\right)^Q$.*

Proof. Let us denote with x_q the search space for each sub-library. It follows that $\sum_q x_q = N$, while the searching complexity of our algorithm takes the form

$$\prod_q \log_2 x_q. \quad (24)$$

Then, the search space values x_q which maximize (24) are $x_1 = \dots = x_Q = \frac{N}{Q}$, hence the complexity of this part of the algorithm is upper-bounded by

$$\left(\log_2 \frac{N}{Q}\right)^Q. \quad (25)$$

□

D. Optimizing the number of sub-libraries Q

Equipped with Algorithm 1, which outputs the optimal library boundaries for an arbitrary Q , we need to search for Q^* such that

$$Q^* = \arg \min_{Q \in [N]} \mathbb{E}\{T_{\mathbf{n}, \mathbf{L}}(Q)\}. \quad (26)$$

As we show in the following lemma, function $T(Q, \mathbf{n}^*)$ is monotonous decreasing in the absence of (P2-d).

Lemma 5. *The objective function of Problem 2, in the absence of (P2-d), is monotonous decreasing with respect to Q .*

Proof. Let us assume some arbitrary Q , for which the optimal delivery time, optimized over $\mathbf{n}^*, \mathbf{L}^*$ takes the form

$$T_Q(\mathbf{n}_Q^*, \mathbf{L}_Q^*) = n_1 + \sum_{q=2}^Q \frac{K_q(1-\gamma)}{L_q(1+\Lambda\gamma)}. \quad (27)$$

We can transition to $Q + 1$ sub-libraries and split the last sub-library into two sub-libraries, i.e.

$$\mathbf{n}_{Q+1} = \{n_1^*(Q), \dots, n_{Q-1}^*(Q), n_Q(Q+1), n_{Q+1}(Q+1)\}$$

and $\mathbf{L}_{Q+1} = \{L_1^*(Q), L_1^*(Q), \dots, L_Q^*(Q), L_Q^*(Q)\}$. The above choice of variables $Q + 1$, \mathbf{n}_{Q+1} , \mathbf{L}_{Q+1} produces the same delivery time as Q , \mathbf{n}_Q^* , \mathbf{L}_Q^* , i.e.

$$T_{Q+1}(\mathbf{n}_{Q+1}, \mathbf{L}_{Q+1}) = T_Q(\mathbf{n}_Q^*, \mathbf{L}_Q^*). \quad (28)$$

Since increasing the number of sub-libraries leads to at least the same delivery time, it follows that the objective function is monotonous decreasing with respect to Q , when optimized over variables \mathbf{n} and \mathbf{L} . □

Theorem 3. *The globally optimum average delay takes the form $T_{Q^*}(\mathbf{n}_{Q^*}^*, \mathbf{L}_{Q^*}^*)$, and is achieved with Q^* sub-libraries, with boundaries $\mathbf{n}_{Q^*}^*$ and redundancy $\mathbf{L}_{Q^*}^*$.*

Proof. Lemma 5 shows the monotonicity of the objective function in the absence of constraint (P2-d). Conversely, by re-introducing the constraint we can guarantee that the objective function is monotonous increasing after point Q^* . Therefore, there exists an optimal number of sub-libraries Q^* that will give the global minimum for the average delay. □

Using the result of Lemma 5 we can see that a simple bisection algorithm, as illustrated in Algorithm 2, in the discrete search space allows to successfully retrieve Q^* .

Algorithm 2: A simple bisection algorithm

- 1 **Input:** $N, K, \gamma, \Lambda, K_T, \gamma_T, \alpha;$
 - 2 **Initialize:** $T_0^* = \infty, T_1^* = \min\{N, \frac{K(1-\gamma)}{L(1+\Lambda\gamma)}\}, Q = 1;$
 - 3 **while** $T_Q^* < T_{Q-1}^*$ **do**
 - 4 $Q = Q + 1;$
 - 5 Compute T_Q^* by solving Problem 2 and using Algorithm 1 for the search range of $\mathbf{n};$
 - 6 $Q^* = Q - 1;$
 - 7 **Output:** $T_{Q^*}^*;$
-

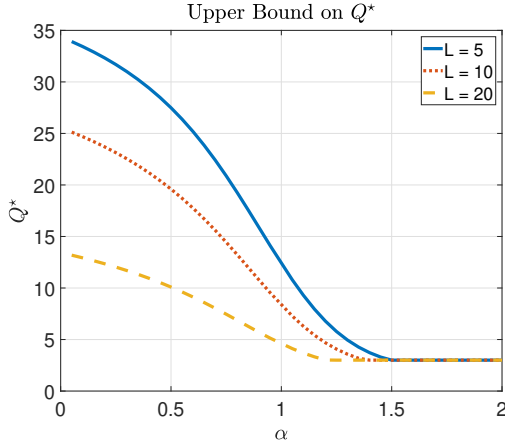


Fig. 2. The bound on the optimal number of sub-libraries, as a function of α for different L values, $K = 2000$, $\Lambda = 40$, $\gamma = 0.1$ and $N = 6000$.

Theorem 4. *The optimal number of sub-libraries will not exceed $\kappa \frac{K}{\Lambda}$, where $\kappa \leq 1$, i.e., $Q^* = O\left(\frac{K}{\Lambda}\right)$.*

Proof. The proof is relegated to Appendix II-E. \square

In Figure 2, we plot the upper bound on the optimal number of sub-libraries, as a function of α for different L values. We set $K = 2000$, $\Lambda = 40$, $\gamma = 0.1$ and $N = 6000$. It is clear that the upper bound on Q^* is less than $\frac{K}{\Lambda} = 50$ and decreases drastically as both α and L increases.

Remark 4. *For most VoD services, typical α values lie between 0.8 and 2 [43]. Moreover, it is also fair to assume that K scales with Λ as modifying the number of users would also require to tune the transmitter side cache fraction γ , eventually tuning Λ to satisfy the subpacketization constraint (2). Finally, Q^* does not scale with the library size N . Therefore, for a wide range of realistic system parameters, Algorithm 1 ensures its polynomial complexity.*

Remark 5. *It is also interesting to note that throughout the simulations (see Section VI) under realistic assumptions, the observed optimal value of the number of sub-libraries Q^* remains very small, and upper bounded by $Q^* = 3$, thus further revealing that the overall complexity of designing the caches of the transmitters remains computationally feasible.*

E. Problem 2's relation to biconvex minimization problems

Before moving on to the analysis of the performance of our proposed method, we would like to discuss the relationship between Problem 2 and the biconvex minimization problems. We begin by proving the biconvexity of our problem.

Lemma 6. *Function (P2-a) is biconvex in \mathbf{L}, \mathbf{n} for fixed Q .*

Proof. The proof is detailed in Appendix II-C. \square

There are various methods and algorithms in the literature for solving biconvex minimization problems through exploitation of the biconvex structure of the problem [44]. For instance, Alternate Convex Search (ACS) is a minimization method, where the variable set is divided into disjoint

blocks [45]–[47], and only one set of variables is optimized while the others remain fixed in each step. ACS does not provide any global optimality guarantee and the final solution may reach a local optimum or a saddle point. The Global Optimization Algorithm (GOA), proposed in [48], aims to take advantage of the biconvex structure of the problem using a primal-relaxed dual approach, which can provide an upper bound and a lower bound to the optimal solution, thus further leading to a finite ϵ -convergence to the global optimum.

Even though our objective function (P2-a) is a biconvex function, constraint (P2-d) is not convex when \mathbf{n} are optimized for fixed Q and \mathbf{L} . Therefore, Problem 2 does not satisfy Conditions(A) provided in [48], which points to the reason why our problem cannot be solved by GOA. Further, using GOA to calculate a bound of our problem, would require the discarding of constraint (P2-d). As we show in the next section (Section V), discarding constraint (P2-d) allows us to reach an analytical solution for the performance of our setting.

We need to note here that a setting where constraint (P2-d) is always satisfied can be interpreted as one with a very high number of users, or more accurately a very high ratio $\frac{K}{\Lambda}$, and very high number of transmitters K_T . In such a setting, as it will also become evident from the simulations (Section VI), the achieved delay and the upper bound performance are becoming narrowly smaller.

V. PERFORMANCE ANALYSIS

In this section we provide a bound on the expected achieved delivery time, and further prove that *any* sub-library segmentation, as described by our main problem (Problem 2), would yield a decreased expected delivery time compared to the uniform popularity case.

The bound is achieved by utilizing the outcome of Lemma 5, describing the monotonicity of the objective function over variable Q , as well as expression (29), obtained in the following lemma, describing the form of the objective function optimized with respect to \mathbf{L} .

Lemma 7. *The optimal allocation of the cache-redundancy vector \mathbf{L} for each sub-library, under the assumption that constraint (P2-d) is satisfied away from the boundaries, results in the objective function*

$$T(Q, \mathbf{n}) = n_1 + \frac{K(1-\gamma)}{1+\Lambda\gamma} \frac{\left(\sum_{q=2}^Q \sqrt{\pi_q(n_q - n_{q-1})}\right)^2}{LN - n_1}. \quad (29)$$

Proof. Inserting the optimal cache-allocation calculated in (18) for $\phi = \psi = \emptyset$ into (16) yields the result. \square

The main idea behind the performance bound is to utilize the monotonicity of the objective function with respect to Q , in the absence of constraint (P2-d), which leads to the conclusion that the expected delay is minimized when $Q = N$.

Lemma 8. *The minimum expected delivery time $\mathbb{E}\{T^*\}$ under the assumption of file popularity following a Zipf distribution with parameter α is lower bounded by*

$$\mathbb{E}\{T^*\} \geq \frac{K(1-\gamma)}{LN(1+\Lambda\gamma)} \frac{\left(\sum_{q=1}^N q^{-\alpha/2}\right)^2}{\sum_{q=1}^N q^{-\alpha}}. \quad (30)$$

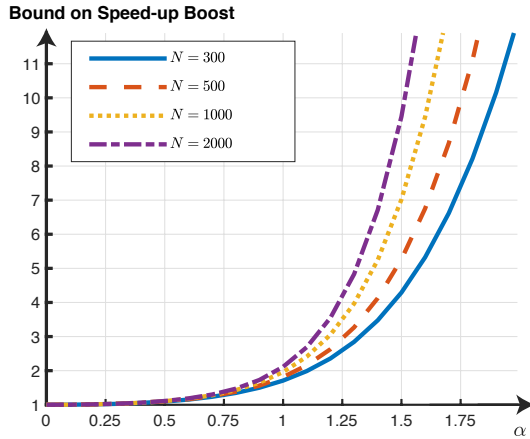


Fig. 3. The bound on the multiplicative boost, (31), as a function of α .

Consequently, the maximum multiplicative ratio G_{\max} that can be achieved by the optimal expected delay T^* compared to the delay of the uniform popularity case is bounded as

$$G_{\max} \leq N \frac{\sum_{q=1}^N q^{-\alpha}}{\left(\sum_{q=1}^N q^{-\alpha/2}\right)^2}. \quad (31)$$

Proof. In order to bound the optimal expected delivery time we remove constraint (P2-d) and constraint $L_q \geq 1$.

Then, it follows from Lemma 5 that the minimum value of (P2-a) is achieved for $Q^* = N$, which implies $\mathbf{n} = [N]$, i.e. each sub-library is comprised of a single file. By incorporating the result of (29) we can write the expectation of the objective function for $Q = N$ and $\mathbf{n} = [N]$ as

$$\mathbb{E}\{T(N, [N])\} = \frac{K(1-\gamma)}{LN(1+\Lambda\gamma)} \left(\sum_{q=1}^N \sqrt{p_q}\right)^2. \quad (32)$$

Using that the fact that the file popularity follows the Zipf distribution with parameter α , we can rewrite (32) as

$$\mathbb{E}\{T(N, [N])\} = \frac{K(1-\gamma)}{LN(1+\Lambda\gamma)} \frac{\left(\sum_{q=1}^N \frac{1}{q^{\alpha/2}}\right)^2}{\sum_{q=1}^N \frac{1}{q^\alpha}}. \quad (33)$$

The ratio between the above result and the uniform-popularity case, where the delivery time is $T_u = \frac{K(1-\gamma)}{L(1+\Lambda\gamma)}$, yields the result of (31). \square

As we can see, the gain achieved is not depend on the number of users K . This is due to the lack of constraint (P2-d) which would otherwise enforce each cache-allocation variable $L_q \leq \min\{\frac{K_q}{\Lambda}, K_T\}$. In Figure 4 we compare the theoretical result from (31) with the numerical results of Section VI. It is interesting to note that as the number of users increases, the gains achieved in the simulations are moving closer to the theoretical bound. This can be attributed to the fact that as the number of users increases the cache-allocation variables L_q are allowed to increase, in conjunction with constraint (P2-d), hence the L_q variables move closer to their optimal values.

We continue with a result that shows that any library segmentation, as long as constraint (P2-d) is satisfied, would

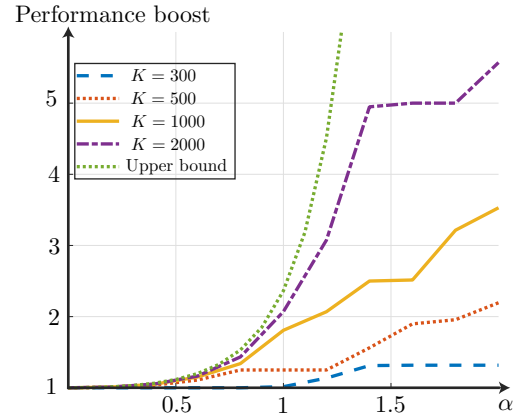


Fig. 4. The multiplicative boost of the expected performance achieved by our algorithm compared to the setting with uniform file popularity. The comparison is displayed here as a function of the Zipf parameter and for various K . The number of files across the examples is $N = 6000$.

lead to a lower or equal delay compared to the uniform popularity case.

Corollary 2. Any library segmentation \mathbf{n} that respects constraint (P2-d) and is optimized over vector \mathbf{L} improves upon the delivery time of the uniform popularity case i.e.

$$\mathbb{E}\{T(Q, \mathbf{n}, \mathbf{L})\} < \frac{K(1-\gamma)}{L(1+\Lambda\gamma)}, \quad (34)$$

$$L_q \leq U_q, \quad \forall q \in [Q], \quad \mathbf{n} \in [N]^Q : n_i < n_j, \quad i < j.$$

Proof. We consider some arbitrary library segmentation \mathbf{n} which respects constraint (P2-d), and the objective function in (29), i.e. after optimized over vector \mathbf{L} .

$$\mathbb{E}\{T(Q, \mathbf{n})\} = \frac{K(1-\gamma)}{L(1+\Lambda\gamma)} \frac{\left(\sum_{q=1}^Q \sqrt{\pi_q(n_q - n_{q-1})}\right)^2}{N} \quad (35)$$

$$\leq \frac{K(1-\gamma)}{L(1+\Lambda\gamma)} \frac{\sum_{q=1}^Q (\sqrt{\pi_q})^2 \sum_{q=1}^Q (\sqrt{n_q - n_{q-1}})^2}{N} \quad (36)$$

$$= \frac{K(1-\gamma)}{L(1+\Lambda\gamma)}. \quad (37)$$

The transition from (35) to (36) makes use of the Cauchy-Schwartz inequality, where the first summation in (36) is equal to 1, while the second summation is equal to N . Thus, any library segmentation \mathbf{n} , under the optimal cache-redundancy allocation, is upper bounded by the delivery time of the uniform popularity setting.

Further, we can deduce the choices of \mathbf{n} that do not improve the delivery time, compared to the uniform case. Specifically, we can view $\mathbb{E}\{T(Q, \mathbf{n})\}$ as the dot product of vectors $\boldsymbol{\pi}_{1/2} \triangleq (\sqrt{\pi_1}, \dots, \sqrt{\pi_Q})$ and $\mathbf{n}_{1/2} \triangleq (\sqrt{n_1}, \sqrt{n_2 - n_1}, \dots, \sqrt{n_Q - n_{Q-1}})$.

In order for the equality to hold in the Cauchy-Schwartz inequality, since neither $\boldsymbol{\pi}_{1/2}$ nor $\mathbf{n}_{1/2}$ can be the all zero vector, it is required that the two vectors are linearly dependent, i.e. $\boldsymbol{\pi}_{1/2} = \lambda \mathbf{n}_{1/2}$, $\lambda \in \mathbb{R}$, [49]. In other words,

$$\lambda^2(n_q - n_{q-1}) = \pi_q, \quad \forall q \in [Q]. \quad (38)$$

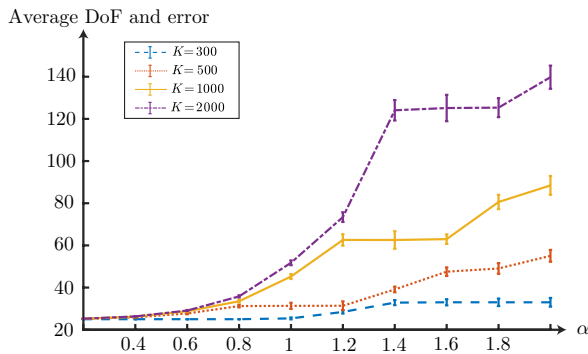


Fig. 5. Expected DoF and standard deviation achieved under the placement dictated by the algorithm of Section IV. User preferences are drawn according to the Zipf distribution. The deviation from the mean is remains less than 1 DoF for small values of α , while for bigger values of α and number of users the deviation is approximately 5 – 10% of the achieved DoF.

Summing (38) over all q yields $\lambda^2 \cdot N = 1$. Thus, for π_1 it must hold that $\pi_1 = \frac{n_1}{N}$, which cannot be satisfied regardless of the sub-library segmentation when $\alpha > 0$ and $Q > 1$. Hence, any choice of \mathbf{n} which satisfies constraint (P2-d), would lead to an improved delivery time compared to the uniform-popularity case. \square

Remark 6. Based on the result of Corollary 2, we can see that this improvement would not necessarily hold in the general library segmentation considered in Problem 1. Specifically, we can easily see that there are many library segmentations that satisfy $\pi_{1/2} = \frac{1}{N} \cdot \mathbf{n}_{1/2}$.

VI. NUMERICAL EVALUATION

To illustrate the performance of our proposed placement, we consider two scenarios that differ on the library sizes and the caching capabilities of the transmitters and the users. The first scenario focuses on a library with TV series, comprised of many files, but each of relatively small size, thus allowing a higher percentage of the library to be stored at the receivers. On the second scenario, we have a library of movies which, although has much fewer individual files, nevertheless each file has higher size.

Scenario 1: We consider a typical, dense multiple transmitter setting [24], [50]–[52], where a set of $K_T = 50$ single-antenna transmitters are connected to K receivers. The content library is comprised of $N = 6000$ TV series episodes, such as typically found in the Netflix catalogue of European countries [53]. The size of each such episode is assumed to be 100MB, i.e. of standard definition quality, while its duration approximately 45min. Each transmitter and each receiver can store 10% of the whole library, i.e. $\gamma_T = \gamma = \frac{1}{10}$ which amounts to 60GB. For a packet size of 1KB the subpacketization is constrained to be $F \leq 10^5$, thus the maximum number of different caches allowed is $\Lambda = 40$ ($\binom{40}{4} \approx 9 \cdot 10^4$).

In Figure 4 we plot, for varying number of users $K \in \{300, 500, 1000, 2000\}$, the ratio of the delivery time of the non-uniform setting over the expected delivery time of our scheme as well as the upper bound calculated in Section V, as a function of parameter α . We observe that for $\alpha = 0.8$

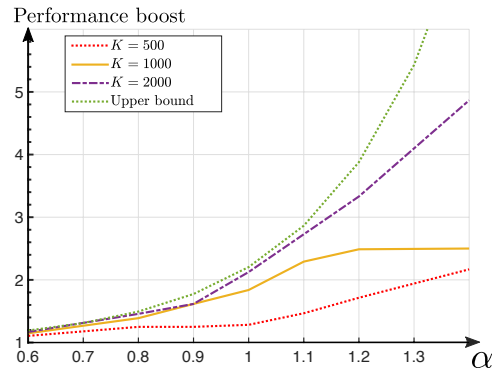


Fig. 6. Scenario 2. The multiplicative boost of the expected performance achieved by our algorithm compared to the setting with uniform file popularity. The comparison is displayed here as a function of the Zipf parameter, for various numbers of users K . The library has $N = 3000$ files.

the delay reduction, compared to the uniform popularity case, ranges between 25% (factor 1.3) and 45% (factor 1.8) for $K = 500$ and $K = 2000$, respectively and further increases to a multiplicative factor of 2.8 for $\alpha = 1.2$. Another important point is that for $\alpha \leq 1.2$ the proposed scheme remains close to the upper bound.

Further, in Figure 5 we plot the average DoF performance as a function of α for all the values of K of our example, as well as the deviation from the mean produced by 10^3 simulations. We note that for practical values of parameter α ($\alpha \leq 1.2$), the DoF performance varies slightly from the mean value.

The optimal sub-library boundaries \mathbf{n}^* and the optimal cache-allocation values \mathbf{L}^* for each of the parameters of Scenario 1 are displayed in Table II.

Scenario 2: Let us now consider another network that aims to serve content from a library of $N = 3000$ movies, typical of a Netflix catalogue [53], each of size of 1GB, of average duration 1.5h, and of standard definition quality. User demands are satisfied by a set of $K_T = 20$ single-antenna transmitters. Due to the much higher per-file size, the normalized cache of a user is $\gamma = \frac{1}{50}$, while we consider that each transmitter's cache is $\gamma_T = \frac{1}{10}$. Hence, a user dedicates 60GB for caching, while a transmitter dedicates 300GB. Assuming, as before, that the minimum packet size is 1KB, translates to a maximum supacketization of $F \leq 10^6$ packets thus, the maximum number of different caches allowed is $\Lambda = 150$ ($\binom{150}{3} \approx 5,5 \cdot 10^5$).

The performance boost, compared to the non-popularity case, is displayed in Figure 6 for varying number of users $K = \{500, 1000, 2000\}$. It is interesting to note that as the number of users increases, one can get close to the bound.

VII. FINAL REMARKS AND CONCLUSIONS

Our work showed for the first time how one can leverage file popularity in order to optimize the cached content at multiple transmitters and achieve multiplicative increase in the performance of coded caching systems. This performance increase can occur even when the file popularity is not very skewed, and it can occur in the subpacketization-constrained regime, where it is indeed needed the most. While most single antenna

α	$K = 300$		$K = 500$		$K = 1000$		$K = 2000$	
	\mathbf{n}^*	\mathbf{L}	\mathbf{n}^*	\mathbf{L}	\mathbf{n}^*	\mathbf{L}	\mathbf{n}^*	\mathbf{L}
0.2	[0]	[5.0000]	[0, 2174]	[5.5415, 4.6928]	[0, 1004, 2183]	[5.9728, 5.1550, 4.6962]	[0, 466, 2138]	[6.4407, 5.2914, 4.6998]
0.4	[0]	[5.0000]	[0, 1923]	[6.2933, 4.3900]	[0, 817, 2530]	[7.4944, 5.2183, 4.3049]	[0, 353, 1907]	[8.9654, 5.7116, 4.3878]
0.6	[0]	[5.0000]	[0, 1678]	[7.3849, 4.0740]	[0, 634, 2262]	[9.8119, 5.4960, 3.9679]	[0, 251, 1652]	[13.1689, 6.3736, 4.0858]
0.8	[0]	[5.0000]	[0, 1431]	[8.8101, 3.6899]	[0, 498, 2036]	[13.3912, 5.8530, 3.6148]	[0, 191, 1439]	[20.4944, 7.1939, 3.7507]
1	[1]	[5.0007]	[0, 1582]	[10.7041, 1.7959]	[0, 490]	[18.2515, 3.8216]	[0, 157, 1278]	[30.3803, 8.1446, 3.4096]
1.2	[1]	[5.0007]	[0, 816]	[11.3591, 1.1409]	[0, 250]	[20.8692, 4.1308]	[0, 233]	[41.1398, 3.6763]
1.4	[1]	[5.0007]	[3]	[5.0020]	[0, 28]	[20.2321, 4.7679]	[0, 212]	[46.6118, 3.3188]
1.6	[1]	[4.2058]	[2]	[5.0013]	[5]	[5.0033]	[0, 98]	[47.8672, 2.1325]
1.8	[1]	[3.5129]	[2]	[3.9464]	[4]	[4.9572]	[0, 15]	[46.3237, 3.6761]
2	[1]	[2.9401]	[1]	[4.9001]	[3]	[4.3115]	[5]	[5.0033]

TABLE II

OPTIMAL SUB-LIBRARY BOUNDARIES \mathbf{n}^* AND OPTIMAL ANTENNA ALLOCATIONS \mathbf{L}^* . THE NUMBER OF SUB-LIBRARIES IS GIVEN BY $Q = 1 + |\mathbf{n}^*|$. WHEN THE FIRST VALUE OF \mathbf{n}^* IS 0 IT POINTS TO AN EMPTY \mathcal{B}_1 SUB-LIBRARY.

cache-aided systems exhibit success in increasing the “usable” part of a user’s cache when exploiting file-popularity at the receiver side, we have showed here how multi-transmitter environments can provide multiplicative gains by becoming popularity-aware, while not affecting the structural symmetry in which coded caching thrives.

REFERENCES

- [1] B. Serbetci, E. Lampiris, T. Spyropoulos, and P. Elia, “Augmenting multiple-transmitter coded caching using popularity knowledge at the transmitters,” in *Proc. Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, June 2020.
- [2] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Trans. Inf. Theory*, vol. 60, pp. 2856–2867, May 2014.
- [3] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Characterizing the rate-memory tradeoff in cache networks within a factor of 2,” *IEEE Trans. Inf. Theory*, vol. 65, pp. 647–663, Jan 2019.
- [4] 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, 2020.
- [5] 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, pp. 1281–1296, Feb 2018.
- [6] C. Shangquan, Y. Zhang, and G. Ge, “Centralized coded caching schemes: A hypergraph theoretical approach,” *IEEE Trans. Inf. Theory*, vol. 64, pp. 5755–5766, Aug 2018.
- [7] H. H. S. Chittoor, P. Krishnan, K. V. S. Sree, and B. Mamillapalli, “Subexponential and linear subpacketization coded caching via projective geometry,” *IEEE Trans. Inf. Theory*, vol. 67, pp. 6193–6222, 2021.
- [8] Q. Yan, M. Cheng, X. Tang, and Q. Chen, “On the placement delivery array design for centralized coded caching scheme,” *IEEE Trans. Inf. Theory*, vol. 63, pp. 5821–5833, Sep. 2017.
- [9] L. Tang and A. Ramamoorthy, “Coded caching schemes with reduced subpacketization from linear block codes,” *IEEE Trans. Inf. Theory*, vol. 64, pp. 3099–3120, April 2018.
- [10] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, “Coded caching with linear subpacketization is possible using Ruzsa-Szeméredi graphs,” in *IEEE Int. Symp. on Inf. Theory (ISIT)*, pp. 1237–1241, June 2017.
- [11] 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, pp. 5524–5537, Oct 2016.
- [12] E. Lampiris and P. Elia, “Adding transmitters dramatically boosts coded-caching gains for finite file sizes,” *IEEE J. Sel. Areas Commun. (JSAC)*, vol. 36, pp. 1176–1188, June 2018.
- [13] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, “Fundamental limits of cache-aided interference management,” *IEEE Trans. Inf. Theory*, vol. 63, pp. 3092–3107, May 2017.
- [14] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, “Multi-server coded caching,” *IEEE Trans. Inf. Theory*, vol. 62, pp. 7253–7271, 2016.
- [15] E. Piovano, H. Joudeh, and B. Clerckx, “On coded caching in the overloaded MISO broadcast channel,” in *IEEE Int. Symp. on Inf. Theory (ISIT)*, pp. 2795–2799, June 2017.
- [16] J. Zhang and P. Elia, “Fundamental limits of cache-aided wireless BC: Interplay of coded-caching and CSIT feedback,” *IEEE Trans. Inf. Theory*, vol. 63, pp. 3142–3160, May 2017.
- [17] E. Parrinello, A. Unsal, and P. Elia, “Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching,” *IEEE Trans. Inf. Theory*, vol. 66, pp. 2252–2268, 2019.
- [18] S. Borst, V. Gupta, and A. Walid, “Distributed caching algorithms for content distribution networks,” in *IEEE Conf. Comp. Commun. (INFOCOM)*, pp. 1–9, March 2010.
- [19] L. Saino, I. Psaras, E. Leonardi, and G. Pavlou, “Load imbalance and caching performance of sharded systems,” *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 112–125, 2020.
- [20] G. Zhang, Y. Li, and T. Lin, “Caching in information centric networking: A survey,” *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013. Information Centric Networking.
- [21] D. Carra, G. Neglia, and P. Michiardi, “Elastic provisioning of cloud caches: A cost-aware ttl approach,” *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1283–1296, 2020.
- [22] A. O. Al-Abbasi, V. Aggarwal, and M. Ra, “Multi-tier caching analysis in CDN-based over-the-top video streaming systems,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 835–847, 2019.
- [23] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, “The role of caching in future communication systems and networks,” *IEEE J. Sel. Areas Commun. (JSAC)*, vol. 36, pp. 1111–1125, June 2018.
- [24] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, “Femtocaching: Wireless content delivery through distributed caching helpers,” *IEEE Trans. Inf. Theory*, vol. 59, pp. 8402–8413, 2013.
- [25] A. Tuholukova, G. Neglia, and T. Spyropoulos, “Optimal cache allocation for femto helpers with joint transmission capabilities,” in *IEEE Int. Conf. Commun. (ICC)*, pp. 1–7, May 2017.
- [26] W. C. Ao and K. Psounis, “Distributed caching and small cell cooperation for fast content delivery,” in *ACM Int. Symp. Mob. Ad Hoc Netw. Comp. (MobiHoc)*, pp. 127–136, 2015.
- [27] U. Niesen and M. A. Maddah-Ali, “Coded caching with nonuniform demands,” *IEEE Trans. Inf. Theory*, vol. 63, pp. 1146–1158, Feb 2017.
- [28] J. Zhang, X. Lin, and X. Wang, “Coded caching under arbitrary popularity distributions,” *IEEE Trans. Inf. Theory*, vol. 64, pp. 349–366, 2018.
- [29] J. Hachem, N. Karamchandani, and S. Diggavi, “Content caching and delivery over heterogeneous wireless networks,” in *IEEE Conf. Comp. Commun. (INFOCOM)*, pp. 756–764, 2015.
- [30] H. Ding and L. Ong, “An improved caching scheme for nonuniform demands and its optimal allocation,” in *IEEE Int. Conf. Comp. Commun. (ICCC)*, pp. 389–393, Dec 2017.
- [31] E. Ozfatura and D. Günduz, “Uncoded caching and cross-level coded delivery for non-uniform file popularity,” in *IEEE Int. Conf. Commun. (ICC)*, pp. 1–6, May 2018.
- [32] S. Jin, Y. Cui, H. Liu, and G. Caire, “Uncoded placement optimization for coded delivery,” in *Proc. Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, pp. 1–8, 2018.
- [33] 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.
- [34] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, “Order-optimal rate of caching and coded multicasting with random demands,” *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3923–3949, 2017.
- [35] A. Ramakrishnan, C. Westphal, and A. Markopoulou, “An efficient delivery scheme for coded caching,” in *Int. Teletraffic Cong.*, 2015.
- [36] Y. Deng and M. Dong, “Subpacketization level in optimal placement for coded caching with nonuniform file popularities,” in *53rd Asilomar Conf. Signals, Systems and Computers*, pp. 1294–1298, 2019.

- [37] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, pp. 1029–1040, Aug 2015.
- [38] M. E. J. Newman, "Power laws, pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, pp. 323–351, 2019.
- [39] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, pp. 1479–1494, 2011.
- [40] E. Parrinello, P. Elia, and E. Lampiris, "Extending the optimality range of multi-antenna coded caching with shared caches," in *IEEE Int. Symp. on Inf. Theory (ISIT)*, June 2020.
- [41] S. Boyd and L. Vandenberghe, *Convex Optimization*. USA: Cambridge University Press, 2004.
- [42] K. Murota, "Discrete convex analysis," *Mathematical Programming*, vol. 83, pp. 313 – 371, 1998.
- [43] S. Gupta and S. Moharir, "Request patterns and caching for vod services with recommendation systems," in *Int. Conf. Commun. Systems and Networks (COMSNETS)*, pp. 31–38, 2017.
- [44] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Mathematical Methods of Operations Research*, vol. 66, pp. 373–407, 2007.
- [45] R. E. Wendell and A. P. Hurter, Jr, "Minimization of non-separable objective function subject to disjoint constraints," *Operations Research*, vol. 24, no. 4, pp. 643–657, 1976.
- [46] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming - Theory and Algorithms (Second ed.)*. John Wiley & Sons Inc., 1993.
- [47] J. De Leeuw and G. Michailidis, "Block relaxation algorithms in statistics," *Information systems and data analysis*, pp. 308–325, 1999.
- [48] C. Floudas and V. Visweswaran, "A global optimization algorithm (gop) for certain classes of nonconvex nlp's—i. theory," *Computers & Chemical Engineering*, vol. 14, no. 12, pp. 1397–1417, 1990.
- [49] T. Apostol, "Mathematical analysis, reading, mass.," *London—Don Mills, Ont.: Addison-Wesley Publishing Co*, 1974.
- [50] L. Liu, V. Garcia, L. Tian, Z. Pan, and J. Shi, "Joint clustering and inter-cell resource allocation for comp in ultra dense cellular networks," in *IEEE Int. Conf. Commun. (ICC)*, pp. 2560–2564, 2015.
- [51] V. Garcia, Y. Zhou, and J. Shi, "Coordinated multipoint transmission in dense cellular networks with user-centric adaptive clustering," *IEEE Trans. Wireless Commun.*, vol. 13, no. 8, pp. 4297–4308, 2014.
- [52] K. Poularakis, G. Iosifidis, I. Pefkianakis, L. Tassiulas, and M. May, "Mobile data offloading through caching in residential 802.11 wireless networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, pp. 71–84, 2016.
- [53] M. Batikas, E. Gomez-Herrera, and B. Martens, "Film availability in netflix country stores in the eu," *Tech. Rep. 2015/11, Inst. Prosp. Tech. Studies Dig. Econ. Working Paper*, Seville, 2015.

APPENDIX I

TRANSMITTER-SIDE CACHING POLICY

To implement the cache-redundancy allocation of our algorithm, for any values of parameters $Q, \mathbf{n}, \mathbf{L}$, we extend the approach of [12] to account for multiple sub-libraries with different redundancies. We note that the objective is to place each file of sub-library \mathcal{B}_q in exactly L_q different transmitters. The proposed placement paired with the delivery process we described in the main part of the document can produce the DoF performance of (9), as illustrated in [12].

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 $1 + (2L_1 - 1 \bmod K_T)$, and so on for the remaining files of \mathcal{B}_1 . 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 from 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 $1 + (n_1 \cdot L_1 \bmod K_T)$, 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 \mathcal{B}_q , using the corresponding L_q , starting every time from the transmitter after the one last used.

Overall, the above process stores each file of sub-library \mathcal{B}_q in exactly L_q distinct transmitters. Further, through this cyclic assignment of files into transmitters we can guarantee that the cache-size constraint is satisfied, i.e. that each transmitter stores exactly $\gamma_T N$ files.

APPENDIX II

PROOFS OF SECTION IV

A. Proof of Corollary 1

The delay required to satisfy solely the demands corresponding to sub-libraries q, r can be written, after normalization by $\frac{K(1-\gamma)}{1+\Lambda\gamma}$, as

$$T_p(L_q, L_r) = \frac{\pi_q}{L_q} + \frac{\pi_r}{L_r}. \quad (39)$$

Using an equal cache-allocation, $L_q = L_r = \tilde{L}$, yields

$$T_p(\tilde{L}, \tilde{L}) = \frac{\pi_q}{\tilde{L}} + \frac{\pi_r}{\tilde{L}}. \quad (40)$$

In contrast, if we assume that the two cache-allocations differ by ℓ such that $\ell < \tilde{L}$, we have

$$T_p(\tilde{L} + \ell, \tilde{L} - \ell) = \frac{\pi_q}{\tilde{L} + \ell} + \frac{\pi_r}{\tilde{L} - \ell} \quad (41)$$

$$\begin{aligned} &= \frac{(\pi_q - \pi_r)\tilde{L}}{\tilde{L}^2 - \ell^2} - \frac{(\pi_q - K_r)\ell}{\tilde{L}^2 - \ell^2} \\ &< \frac{(\pi_q - \pi_r)}{\tilde{L}} - \frac{(\pi_q - \pi_r)\ell}{\tilde{L}^2} < T_p(\tilde{L}, \tilde{L}), \end{aligned} \quad (42)$$

which shows that it is always a better strategy to allocate higher cache redundancy to sub-libraries with higher cumulative probability. \square

B. Proof of Lemma 1

We assume that $\{\mathcal{B}_q\}_{q=1}^Q$ is an optimal library segmentation where, without loss of generality, $\pi_q \geq \pi_{q+1}, \forall q \in [Q-1]$. In a different case we can rename the sub-libraries such that $\pi_q \geq \pi_{q+1}, \forall q \in [Q-1]$. We pick two files, W^{r_a}, W^{r_b} , with corresponding popularity $p_{r_a} > p_{r_b}$, such that $W^{r_a} \in \mathcal{B}_a$ and $W^{r_b} \in \mathcal{B}_b$, while $\pi_a < \pi_b$, else we wouldn't have anything to prove. Further, assuming that L_a and $L_b > L_a$ correspond to the optimal cache-allocation of \mathcal{B}_a and \mathcal{B}_b , respectively we can calculate the expected delay, T_1 , of this sub-library segmentation and cache-allocation using (11).

Now, we can proceed to calculate the delay, T_2 of a similar system with the same cache-allocation as before, but now files W^{r_a}, W^{r_b} are swapped, i.e. $W^{r_a} \in \mathcal{B}_b$ and $W^{r_b} \in \mathcal{B}_a$.

The difference of the two delays then takes the form

$$T_1 - T_2 = \frac{p_{r_a}}{L_a} + \frac{p_{r_b}}{L_b} - \left(\frac{p_{r_b}}{L_a} + \frac{p_{r_a}}{L_b} \right) \quad (43)$$

$$= (p_{r_a} - p_{r_b}) \frac{L_b - L_a}{L_b \cdot L_a} > 0. \quad (44)$$

Using the above result, and beginning from some arbitrary segmentation of the library we can select pairs of files which belong in different sub-libraries such that the probability of one file is higher than the probability of the other, while the more popular file resides in the less popular sub-library and swap them. As we showed, performing this task will always transition the system to a lower delivery time.

Continuing to perform this task would result in a library segmentation where each sub-library has files of consecutive indices. \square

C. Convexity of (16) for fixed Q, \mathbf{n}

Let us define the Hessian of T_Q^* with respect to \mathbf{L} by \mathbf{H}_1 , which is given by

$$\mathbf{H}_1 = \begin{bmatrix} \frac{\partial^2 T_Q^*(Q)}{\partial L_1^2} & \frac{\partial^2 T_Q^*(Q)}{\partial L_1 L_2} & \cdots & \frac{\partial^2 T_Q^*(Q)}{\partial L_1 L_Q} \\ \frac{\partial^2 T_Q^*(Q)}{\partial L_2 L_1} & \frac{\partial^2 T_Q^*(Q)}{\partial L_2^2} & \cdots & \frac{\partial^2 T_Q^*(Q)}{\partial L_2 L_Q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 T_Q^*(Q)}{\partial L_Q L_1} & \frac{\partial^2 T_Q^*(Q)}{\partial L_Q L_2} & \cdots & \frac{\partial^2 T_Q^*(Q)}{\partial L_Q^2} \end{bmatrix}. \quad (45)$$

Focusing on the q -th diagonal element of \mathbf{H}_1 we have

$$\frac{\partial^2 T_Q^*}{\partial L_q^2} = \frac{\partial^2}{\partial L_q^2} \left(n_1 + \sum_{r=2}^Q \frac{K(1-\gamma)\pi_r}{L_r(1+\Lambda\gamma)} \right) = 2 \frac{K\pi_q(1-\gamma)}{L_q^3(1+\Lambda\gamma)} > 0.$$

Similarly, we can show that the non-diagonal elements of \mathbf{H}_1 are equal to 0. Let us consider arbitrary element (q, s) , $q \neq s$ for which we have

$$\begin{aligned} \frac{\partial^2 T_Q^*}{\partial L_q \partial L_s} &= \frac{\partial^2}{\partial L_q \partial L_s} \left(n_1 + \sum_{r=2}^Q \frac{K\pi_r(1-\gamma)}{L_r(1+\Lambda\gamma)} \right) \\ &= \frac{\partial}{\partial L_s} \left(-\frac{K\pi_q(1-\gamma)}{L_q^2(1+\Lambda\gamma)} \right) = 0. \end{aligned}$$

We can now conclude that \mathbf{H} is positive semi-definite since its diagonal elements are positive, while its non-diagonal elements are zero. Thus, function $T_{Q,\mathbf{n}}^*$ is convex in \mathbf{L} .

Now, we continue with proving the convexity of $T^*(Q)$ in \mathbf{n} for fixed \mathbf{L} . Since p_j is defined only for discrete j , we replace the Zipf distribution with a continuous Pareto distribution,

$$f(j) = j^{-\alpha} \underbrace{(H(N, \alpha))^{-1}}_{\triangleq C},$$

where $H(N, \alpha)$ is the generalized Harmonic number.

Let us define the Hessian of $T^*(Q)$ with respect to \mathbf{n} as \mathbf{H}_2 , which is given similarly to (45).

Following similar arguments used in showing the positive semi-definiteness of \mathbf{H}_1 , we will show that \mathbf{H}_2 is also positive semi-definite. It is trivial to show that first diagonal element of \mathbf{H}_2 is always non-negative. Let us consider a different diagonal element $q > 1$, for which we get

$$\frac{\partial^2 T_Q^*}{\partial n_q^2} = -\frac{K(1-\gamma)\alpha n_2^{-(\alpha+1)}}{C(1+\Lambda\gamma)} \left[\frac{1}{L_q} - \frac{1}{L_{q+1}} \right] \geq 0,$$

due to the fact that $L_q \geq L_{q+1}$, $\forall q \in [2, Q]$.

Since function (16) is a linear combination of terms, where each term is solely dependent on one of the L_q variables

it follows that a double partial differentiation over different L_q, L_k would produce 0. Therefore, we conclude that \mathbf{H}_2 is positive semi-definite and function T_Q^* is convex in \mathbf{n} for fixed \mathbf{L} . Therefore, the function in (29) is biconvex.

Finally, all the constraints are affine. Thus, Problem 2 for fixed Q is a biconvex problem. \square

D. Proof of Lemma 3

The output of the KKT conditions provides three different, and non-overlapping, subsets of $[Q]$. The first set, ϕ , is comprised of those $q \in [Q]$ for which $L_q = 1$ (apart from $q = 1$ for which, by definition, $L_1 = 1$). The second set, ψ , is comprised of those q for which $L_q = U_q$. Finally, the remaining q are contained in set χ and for these we need to calculate the cache-allocation variable L_q .

Hence, the expected delay can be written as

$$\begin{aligned} T^*(Q, \mathbf{n}) &= n_1 + \frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{q \in \phi} \pi_q + |\psi| \frac{\Lambda(1-\gamma)}{1+\Lambda\gamma} \\ &\quad + \frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{q \in \chi} \frac{\pi_q}{L_q}. \end{aligned} \quad (46)$$

The cache capacity constraint becomes

$$\begin{aligned} \sum_{q \in \chi} L_q (n_q - n_{q-1}) &= \\ LN - \left[n_1 + \sum_{q \in \phi} (n_q - n_{q-1}) + \sum_{q \in \psi} U_q (n_q - n_{q-1}) \right]. \end{aligned} \quad (47)$$

Taking the derivative of (17) with respect to L_q such that $q \in \chi$, and equating it to 0 yields

$$\frac{\partial \mathcal{L}}{\partial L_q} = -\frac{K\pi_q(1-\gamma)}{(1+\Lambda\gamma)L_q^2} + \lambda(n_q - n_{q-1}) = 0. \quad (48)$$

Separating L_q from the remaining terms in (48) yields

$$L_q = \frac{1}{\sqrt{\lambda}} \sqrt{\frac{K(1-\gamma)\pi_q}{(1+\Lambda\gamma)(n_q - n_{q-1})}} \quad (49)$$

$$L_q (n_q - n_{q-1}) = \frac{1}{\sqrt{\lambda}} \sqrt{\frac{K(1-\gamma)\pi_q (n_q - n_{q-1})}{(1+\Lambda\gamma)}}. \quad (50)$$

Further, from (48) keeping on one side terms $L_q, (n_q - n_{q-1}), \lambda$ we get

$$L_q (n_q - n_{q-1}) \lambda = \frac{K(1-\gamma)\pi_q}{(1+\Lambda\gamma)L_q}. \quad (51)$$

Summing (50) and (51) over all $q \in \chi$

$$\stackrel{(50)}{\Rightarrow} \sum_{q \in \chi} L_q (n_q - n_{q-1}) = \frac{1}{\sqrt{\lambda}} \sum_{q \in \chi} \sqrt{\frac{K(1-\gamma)\pi_q (n_q - n_{q-1})}{(1+\Lambda\gamma)}} \quad (52)$$

$$\stackrel{(51)}{\Rightarrow} \sum_{q \in \chi} L_q (n_q - n_{q-1}) = \frac{1}{\lambda} \frac{K(1-\gamma)}{(1+\Lambda\gamma)} \sum_{q \in \chi} \frac{\pi_q}{L_q}. \quad (53)$$

Replacing λ from (53) to (52) yields

$$\sum_{q \in \mathcal{X}} L_q(n_q - n_{q-1}) = \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{q \in \mathcal{X}} \sqrt{\pi_q(n_q - n_{q-1})}}{\frac{K(1-\gamma)}{(1+\Lambda\gamma)} \sum_{q \in \mathcal{X}} \frac{\pi_q}{L_q}}$$

$$\frac{K(1-\gamma)}{(1+\Lambda\gamma)} \sum_{q \in \mathcal{X}} \frac{\pi_q}{L_q} = \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma} \sum_{q \in \mathcal{X}} \sqrt{\pi_q(n_q - n_{q-1})}}{\sum_{q \in \mathcal{X}} L_q(n_q - n_{q-1})}. \quad (54)$$

Replacing the left-hand-side of (54) with (46) and the denominator of (54) from (47) yields the result of Theorem 1. \square

E. Proof of Theorem 4

We will show that for any given set of system parameters, a solution to Problem 2 for fixed \mathbf{n} and fixed Q is feasible only when $Q \leq \kappa \frac{K}{\Lambda}$, where $\kappa < 1$.

Since Problem 2 aims to minimize the objective function (P2-a), for any fixed \mathbf{n} and Q , the solution obtained by KKT conditions will spend the *cumulative budget* LN in (P2-c) as much as possible, i.e., the left hand side of (P2-c) will be maximized as increasing the number of transmitters allocated to sub-libraries decreases the objective function (P2-a).

Let $\hat{\mathbf{L}}$ be a feasible solution to Problem 2 consisting of \hat{Q} sub-libraries. We also set $n_1 = 0$, i.e., we assume that the first sub-library is empty, and will not spend any collective transmitter side cache capacity budget. Since we are going to provide an upper bound on Q^* , we will later consider the possibility of this sub-library being non-empty¹¹.

We will only focus on the case $K_T \geq \frac{K\pi_q}{\Lambda}$, $\forall q \in [\hat{Q}]$ as $K_T \leq \frac{K\pi_q}{\Lambda}$ case is trivial. Consequently, $U_q = \frac{K\pi_q}{\Lambda}$, $\forall q \in [\hat{Q}]$, and we assume that the following ordering is satisfied¹²,

$$\hat{L}_2 \geq \hat{L}_3 \geq \dots \geq \hat{L}_{\hat{Q}}.$$

For $\alpha > 0$, it is easy to verify from (5) that $p_1 > p_2 > \dots > p_N$. We also know from (P2-d) that

$$\frac{K\pi_q}{\Lambda} \geq 1, \forall q \in [\hat{Q}] \quad (55)$$

must hold since $U_q \geq 1$ for any feasible solution.

Now, we will find the maximum number of groups that yields a feasible solution for the case where the cumulative budget in (P2-c) is spent as much as possible —so that (P2-a) is minimized as discussed above—.

Let us set $L_{\hat{Q}} = 1$, and $L_{\hat{Q}-1} = 1 + \epsilon_{\hat{Q}-1}$, where $\epsilon_{\hat{Q}-1}$ is arbitrarily small. Then, (55) must hold with equality for $q = \hat{Q}$. Similarly, for $q = \hat{Q} - 1$, (55) holds with equality where the right hand side is equal to $1 + \epsilon_{\hat{Q}-1} \approx 1$.

Since $p_1 > p_2 > \dots > p_N$, it is clear that in order to have $\pi_{\hat{Q}} \approx \pi_{\hat{Q}-1}$, $|\mathcal{B}_{\hat{Q}}| > |\mathcal{B}_{\hat{Q}-1}|$ for any $\alpha > 0$.

Following the same argument, a feasible solution that leads to the maximum possible number of sub-libraries \hat{Q} and simultaneously satisfies (P2-c) and (P2-d) would then need to spend as minimum budget as possible for the sub-libraries

¹¹Note that if $n_1 \neq 0$, then $L_1 = 1$ for the files within this sub-library. By setting $n_1 = 0$, we restrict to the case where all files are served by coded caching and no files to be broadcasted, eventually minimizing the cache capacity budget that is spent by the first sub-library.

¹²Any feasible solution will satisfy a similar ordering with a possible permutation $[2, \hat{Q}] \rightarrow [2, \hat{Q}]$.

\mathcal{B}_3 to $\mathcal{B}_{\hat{Q}}$ and spend the remaining budget for the sub-library \mathcal{B}_2 to satisfy (P2-c), yielding

$$\hat{\mathbf{L}} = [U_2, 1 + \epsilon_3, \dots, 1 + \epsilon_{\hat{Q}-1}, 1], \quad (56)$$

where $U_2 = \frac{K\pi_2}{\Lambda}$.

Next, we will show that \hat{Q} will not exceed $\kappa \frac{K}{\Lambda}$. Following a similar argument that we used in Appendix II-C, we will replace the Zipf distribution with the continuous Pareto distribution. We will consider the cases of $\alpha \neq 1$ and $\alpha = 1$ separately.

a) *Case 1: $\alpha \neq 1$:* In this case, we have

$$\pi_q = \frac{n_q^{1-\alpha} - n_{q-1}^{1-\alpha}}{C(1-\alpha)}.$$

Then, for the feasible solution given in (56), for $\epsilon_q \rightarrow 0$, $\forall q \in [3, \hat{Q} - 1]$, (P2-c) becomes

$$\frac{K}{\Lambda} \left(1 - \frac{N^{1-\alpha} - n_2^{1-\alpha}}{C(1-\alpha)} \right) n_2 + (N - n_2) \leq LN$$

$$\left(\frac{K}{\Lambda} - \frac{KN^{1-\alpha}}{\Lambda C(1-\alpha)} - 1 \right) n_2 + \frac{K}{\Lambda C(1-\alpha)} n_2^{2-\alpha} - (L-1)N \leq 0, \quad (57)$$

where $C \triangleq H(N, \alpha)$ is the generalized Harmonic number. Also, from (P2-d), we have

$$\frac{K}{\Lambda} \left(\frac{N^{1-\alpha} - n_2^{1-\alpha}}{C(1-\alpha)} \right) \geq 1. \quad (58)$$

Let ν_1 and ν_2 be the solutions to (57) and (58) respectively with equality, i.e.,

$$\left(\frac{K}{\Lambda} - \frac{KN^{1-\alpha}}{\Lambda C(1-\alpha)} - 1 \right) \nu_1 + \frac{K}{\Lambda C(1-\alpha)} \nu_1^{2-\alpha} - (L-1)N = 0, \quad (59)$$

and

$$\nu_2 = \left(N^{1-\alpha} - \frac{C(1-\alpha)\Lambda}{K} \right)^{\frac{1}{1-\alpha}}. \quad (60)$$

To maximize the left hand side of (P2-c), we need to find the maximum n_2 , henceforth denoted by \hat{n}_2 , that satisfies both (57) and (58). Therefore, we have $\hat{n}_2 = \min\{\nu_1, \nu_2\}$.

The remaining *budget* must be distributed among $\hat{Q} - 2$ sub-libraries¹³ evenly —since for $\epsilon_q \rightarrow 0$, $\forall q \in [3, \hat{Q} - 1]$, the cache redundancies of these sub-libraries are equal— to obtain the feasible solution given in (56), yielding

$$\frac{K}{\Lambda} \left(\frac{N^{1-\alpha} - \hat{n}_2^{1-\alpha}}{C(1-\alpha)} \right) \frac{1}{\hat{Q} - 2} \geq 1$$

$$\hat{Q} \leq \frac{K}{\Lambda} \left(\frac{N^{1-\alpha} - \hat{n}_2^{1-\alpha}}{C(1-\alpha)} \right) + 2. \quad (61)$$

Finally, since the optimum number of sub-libraries Q^* must be less than or equal to the maximum number of sub-libraries that yields to the feasible solution spending the cumulative budget as much as possible, $Q^* \leq \hat{Q}$, and Q^* is then upper bounded by

$$Q^* \leq \frac{K}{\Lambda} \left(\frac{N^{1-\alpha} - \hat{n}_2^{1-\alpha}}{C(1-\alpha)} \right) + 2. \quad (62)$$

¹³Note that we set $n_1 = 0$ in the beginning. As we are providing an upper bound, we are adding the possibility of \mathcal{B}_1 being non-empty.

b) *Case 2: $\alpha = 1$:* We follow the same arguments as in Case 1, and we skip the details due to space restrictions. In this case, we have

$$\pi_q = \frac{\log n_q - \log n_{q-1}}{\log N}.$$

Then, for the feasible solution given in (56), for $\epsilon_q \rightarrow 0$, $\forall q \in [3, \hat{Q} - 1]$, (P2-c) becomes

$$\frac{K \log n_2}{\Lambda \log N} + N - n_2 \leq LN, \quad (63)$$

and from (P2-d), we have

$$\frac{K}{\Lambda} \left(1 - \frac{\log n_2}{\log N} \right) \geq 1. \quad (64)$$

Consequently, n_2 must satisfy both (63) and (64). Let ν_1 and ν_2 be the solutions to (63) and (64) respectively with equality. Then, we have

$$\nu_1 = \frac{(L-1)\Lambda N \log N}{KW \left(\frac{(L-1)\Lambda N \log N}{K} e^{\frac{\Lambda \log N}{K}} \right)}, \quad (65)$$

where $W(\cdot)$ is the Lambert W function, and

$$\nu_2 = N^{1 - \frac{\Lambda}{K}}. \quad (66)$$

To maximize the left hand side of (P2-c), we need to find the maximum n_2 , henceforth denoted by \hat{n}_2 , that satisfies both (63) and (64). Therefore, we have $\hat{n}_2 = \min\{\nu_1, \nu_2\}$.

Finally, following the same arguments as in the $\alpha \neq 1$ case, we have

$$Q^* \leq \frac{K}{\Lambda} \left(1 - \frac{\log \hat{n}_2}{\log N} \right) + 2. \quad (67)$$

It is easy to verify that for any $\alpha > 0$, Q^* is less than $\kappa \frac{K}{\Lambda}$, where $\kappa < 1$, and κ decreases as both α and L increase. As a consequence, Q^* does not scale with the library size N . \square



Berkans Serbetci received the B.Sc. degree in Electrical and Electronics Engineering from Middle East Technical University in 2009, the M.Sc. degree in Electrical and Electronics Engineering from Bogazici University in 2012, and the Ph.D. degree in Applied Mathematics from the University of Twente in 2018. He is currently a postdoctoral fellow with the Communication Systems Department, EURECOM. His research interests include caching, wireless networks, optimization theory, stochastic processes, stochastic geometry, information theory

and machine learning.



Eleftherios Lampiris received the B.Sc. degree in Physics and the M.Sc. degree in Radio-electrology and Electronics both from the National & Kapodistrian University of Athens, Greece. He further obtained a Ph.D. degree in Electrical Engineering from Sorbonne University, France while employed at EURECOM. Since, he held post-doctoral researcher positions at the Technical University of Berlin, and at EURECOM. His latest obsessions include Artificial Intelligence and Art, Computer Vision, Machine Learning, and their implementation. He is the founder of ArtificialCanvas, a place merging traditional art with technology.



Thrasyvoulos Spyropoulos received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and a Ph.D degree in Electrical Engineering from the University of Southern California. He was a post-doctoral researcher at INRIA and then, and a senior researcher with the Swiss Federal Institute of Technology (ETH) Zurich. Between 2010 and 2022 he was a Professor at EURECOM, Sophia-Antipolis. He is currently a Professor at the Technical University of Crete, Greece. He is the co-recipient of the best paper award in IEEE SECON 2008, and IEEE WoWMoM 2012, as well as the runner-up for ACM MobiHoc 2011, IEEE WoWMoM 2015 and IEEE WoWMoM 2021.



Giuseppe Caire (S '92 – M '94 – SM '03 – F '05) was born in Torino in 1965. He received a B.Sc. in Electrical Engineering from Politecnico di Torino in 1990, an M.Sc. in Electrical Engineering from Princeton University in 1992, and a Ph.D. from Politecnico di Torino in 1994. He has been a post-doctoral research fellow with the European Space Agency (ESTEC, Noordwijk, The Netherlands) in 1994-1995, Assistant Professor in Telecommunications at the Politecnico di Torino, Associate Professor at the University of Parma, Italy, Professor with the Department of Mobile Communications at the Eurecom Institute, Sophia-Antipolis, France, a Professor of Electrical Engineering with the Viterbi School of Engineering, University of Southern California, Los Angeles, and he is currently an Alexander von Humboldt Professor with the Faculty of Electrical Engineering and Computer Science at the Technical University of Berlin, Germany.

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 in 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 best IEEE JSAC paper in 2019, the IEEE Communications Society Edwin Howard Armstrong Achievement Award in 2020, and he is a recipient of the 2021 Leibniz Prize of the German National Science Foundation (DFG). Giuseppe Caire is a Fellow of IEEE since 2005. He has served in the Board of Governors of the IEEE Information Theory Society from 2004 to 2007, and as officer from 2008 to 2013. His main research interests are in the field of communications theory, information theory, channel and source coding with particular focus on wireless communications.



Petros Elia received the B.Sc. degree from the Illinois Institute of Technology, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Southern California (USC), Los Angeles, in 2001 and 2006 respectively. He is now a professor with the Department of Communication Systems at EURECOM in Sophia Antipolis, France. His latest research deals with the intersection of coded caching and feedback-aided communications in multiuser settings. He has also worked in the area of complexity-constrained communications, MIMO, queueing theory and cross-layer design, coding theory, information theoretic limits in cooperative communications, and surveillance networks. He is a Fulbright scholar, the co-recipient of the NEWCOM++ distinguished achievement award 2008-2011 for a sequence of publications on the topic of complexity in wireless communications, the recipient of the ERC Consolidator Grant 2017-2022 on cache-aided wireless communications, and the recipient of the ERC-PoC 2022-2024.