Fundamental Limits of Topology-Aware Shared-Cache Networks

Emanuele Parrinello, *Member, IEEE,* Antonio Bazco-Nogueras, *Member, IEEE,* and Petros Elia, *Member, IEEE*

***Abstract*—This work studies a well-known shared-cache coded caching scenario where each cache can serve an arbitrary number of users. We analyze the case where there is some knowledge about such number of users (i.e., the topology) during the content placement phase. Under the assumption of regular placement and a cumulative cache size that can be optimized across the different caches, we derive the fundamental limits of performance by introducing a novel cache-size optimization and placement scheme and a novel information-theoretic converse. The converse employs new index coding techniques to bypass traditional uniformity requirements, thus finely capturing the heterogeneity of the problem, and it provides a new approach to handle asymmetric settings. The new fundamental limits reveal that heterogeneous topologies can in fact outperform their homogeneous counterparts where each cache is associated to an equal number of users. These results are extended to capture the scenario of topological uncertainty where the perceived/estimated topology does not match the true network topology. This scenario is further elevated to the stochastic setting where the user-to-cache association is random and unknown, and it is shown that the proposed scheme is robust to such noisy or inexact knowledge on the topology.**

***Index Terms*—Coded Caching, Shared cache, topology-aware, index coding, combinatorics**

I. Introduction

CODED caching is a communications technique proposed by Maddah-Ali and Niesen (MAN) in [2] which exploits cached content in order to reduce congestion in communication networks. This work in [2] revealed that a careful placement of content at the caches that are locally available to the users can substantially boost content delivery rates, by allowing for multicasting opportunities that enable the transmitter to serve multiple users at the same time through a shared link. Specifically, the work in [2] considered a noiseless shared-link broadcast channel (BC) where a server, with access to a library of N files, aims to communicate with K users that are each equipped with a cache that can store a fraction $\gamma \in [0,1]$ of the library. The system consists of two distinct phases; a *cache placement phase* during which — without knowledge of the future requests of the users — portions of the files of the library are pre-stored at the users' cache, and a *delivery phase* in which the users' demands are served. A subdivision of the library files in many sub-files and a meticulous placement of these sub-files at the users' caches allow the server to simultaneously transmit to $K\gamma+1$ users during the delivery phase. This factor of $K\gamma+1$ describes the speedup in delivery rates due to *coded* caching, and it is commonly referred to as the *coding gain*, or as the *global caching gain*, and it matches the well known *Degrees-of-Freedom* (DoF). This DoF — which was shown to be information-theoretically optimal within a gap of 2 in [3] and exactly optimal under the assumption of uncoded cache placement in [4], [5] — scales with the cumulative cache capacity of the network $K\gamma$.

Since the introduction of coded caching, various works have extended the original MAN approach to various interesting scenarios. For example, the works in [6], [7] addressed the average performance under the assumption of a non-uniform popularity distribution of the library files, while the work in [8] explored a *decentralized* scenario where, during cache placement, the server is not aware of the number and identity of the users that will be present during the delivery phase. Similar concepts were also later considered in [9], [10]. Another interesting work can be found in [11], which extended the shared-link BC in [2] to the device-to-device (D2D) setting where users exchange messages in a peer-to-peer fashion to satisfy their requests. Furthermore, advances in coded caching were proved to be also applicable in distributed computing through the development of coded distributed computing [12], [13], [14], [15], [16].

An interesting direction included the study of coded-caching involving a server having multiple (C) transmitting antennas [17] or equivalently the decentralized scenario with multiple transmitting servers [18]. In both cases, the derived DoF of $K\gamma+C$ was shown to be order optimal (cf. [18]) and later to be exactly optimal (cf. [19]) under the assumption of linear one-shot schemes. Subsequently, finite-SNR studies of the multi-antenna setting can be found in [20], [21], [22], [23], the interplay of caching and Channel State Information (CSI) has been also analyzed in [24], [25], as well as the impact of cache-less users [26] or uncacheable traffic [27]. All the above aforementioned works, and in general most research on coded caching, focus on the setting where each user is aided by its own dedicated cache. Recently though, the *shared caches* paradigm has emerged as a much more realistic as well as powerful alternative to the traditional dedicated caches scenario. While in the latter scenario each user can have its own dedicated cache that can be drawn/filled independently of

E. Parrinello and P. Elia are with the Communication Systems Department, EURECOM, France, e-mail: {emanuele.parrinello; petros.elia}@eurecom.fr.

A. Bazco-Nogueras is with the IMDEA Networks Institute, Madrid, Spain, e-mail: antonio.bazco@imdea.org.

This work was supported by the European Research Council (ERC) through the EU Horizon 2020 Research and Innovation Program under Grant 725929 (Project DUALITY), and under Grant 101101031 (Project LIGHT), as well as by the Huawei France-funded Chair towards Future Wireless Networks. The work of A. Bazco-Nogueras was supported by the Regional Government of Madrid through the grant 2020-T2/TIC-20710 for Talent Attraction. This work was presented in part at the 2019 IEEE Information Theory Workshop (ITW) [1].

the other caches, this new shared caches approach asks that each cache can serve — as argued below — multiple users at the same time. As we will see, this shared caches approach captures several scenarios of interest.

A. The shared cache model for coded caching

The information theoretic study of the shared-cache coded caching scenario is strongly motivated by practical considerations and recent trends (see [28] and references therein). Moreover, it directly links our ability to exploit caching at the radio access network to a new ability to employ low-priced storage units at newly deployed macro and micro base stations, each serving various sets of users. In our context, the central server represents a macro base station (MBS), whereas each cache represents a cache-aided micro base station (SBS) that can serve its nearby users at data rates that are significantly higher than those from a distant MBS.

The first extensive information-theoretic study that focuses on this shared-cache model can be found in [29], which considered the scenario where each user has access to one of several caches at zero prefetching cost, and where each cache's occupancy (i.e., the number of users associated to each cache) can be arbitrary but *unknown* during the cache placement phase. For this general setting, the work in [29] characterized the corresponding fundamental limits by deriving the exact optimal (under uncoded cache placement) worst-case normalized delivery time. An interesting subsequent work can be found in [30], which extends the setting in [29] to account for error-prone links. This work in [30] proposes a scheme that employs the "leaders" approach from [8] to extend the scheme in [29] to account for the scenario where multiple users may request the same file. Furthermore, the work in [31] considered the shared-caches scenario with coded placement, which showed that the benefit of coded placement increases as the asymmetry in the number of users per cache increases, while a decentralized version of the scheme in [29] was proposed in [32]. There exist additional interesting works for the scenario where each user can be associated to more than one cache [33], [34], [35], [36], [37], [38], although this latter scenario is not considered in this current work.

It is worth noting that this shared cache model can be applied in a variety of settings that include the idealized model of cache-aided heterogeneous cellular networks, the multiple file request problem [39], [40], [41], and last but not least, this same shared cache approach can be effectively used to account for the omnipresent subpacketization bottleneck of coded caching [9], [42], [43], which in essence forces different users to cache the same content. Interesting works on this latter matter can be found in [9], [43]. In [9], Jin et al. proposed a decentralized shared-cache scheme for the subpacketization-constrained (finite file size) scenario, while [43] revealed for the first time that in the presence of multiple antennas the shared-cache approach can dramatically alleviate the subpacketization problem while simultaneously exploiting both multiplexing and caching gains in their entirety. Recently, [44] analyzed the case in which there exist both dedicated and shared caches, and [45] studied the case where the library files are correlated. Other related works can be found in [46], [47], [48].

Moreover, an additional implication of the shared-cache setting was recently revealed in [49], this time linking this shared-cache approach to the well known worst-user bottleneck, where this bottleneck was previously thought to be fundamental to coded caching. In particular, the work in [49] has provided a new method that exploits the often unavoidable need for shared caches (where users in predefined groups are forced to place identical content in their caches) in a manner than can entirely alleviate this worst-user bottleneck, thus proving that this bottleneck is not fundamental to coded caching. This method was also proved to overcome the worst-user effect in the presence of fast fading and different path losses in [50].

The above jointly suggest that the shared cache scenario can form a pivotal ingredient in any attempt to meaningfully employ coded caching. This motivates our information-theoretic study of this scenario.

B. Memory allocation in cache-aided networks

Optimizing memory allocation in cache-aided systems has been a critical topic of study in many works (see for example [51]). This is particularly true in heterogeneous scenarios like the one we consider here, where one expects to find larger caches that serve the many users of large office buildings coexisting with smaller caches that serve smaller pockets of population. It is worth remarking that deriving fundamental limits for asymmetric settings is a challenging topic, as it has been shown in the recent work [52], where the authors fully characterized the exact capacity of the 2-file/2-user setting simultaneously allowing for heterogeneous files sizes, heterogeneous cache sizes, and user-dependent file popularities.

Various works have explored this problem of memory allocation in cache-aided communication systems. For example, in the context of traditional (non-coded) caching, the works [53], [54] investigated this problem for the scenario of backhaul-limited cache-aided small-cell networks, while the work in [55] explored this problem for the scenario of a cache-enabled heterogeneous small-cell network, for which it proceeded to minimize the average backhaul load by optimizing — subject to a cumulative cache capacity — the distribution of cache sizes across the network. In the context of coded caching, [56] showed that unselfish memory allocation provides significant gains even in selfish settings where each user is interested only in a different subset of the library content.

In this work, we consider the memory allocation problem in a cache-aided broadcast channel in the context of a shared-cache framework, where the size of each cache can be optimized under the assumption of a sum cache size constraint. To the best of our knowledge, this memory allocation problem (i.e., where to allocate the memory resources) was first addressed in the context of coded caching in [57] and [58], both for the setting with dedicated caches (or, equivalently, where each cache serves a single user), while here we consider shared-cache framework. In [57], the authors proposed a delivery scheme with memory allocation optimization for a

wireless backhaul link, which was modeled as a BC where the link rates are fixed and known and the cache sizes can be changed. The work in [58] considers the rate-memory trade-off in general cache-aided degraded broadcast channel scenario. In other words, [58] analyzes a scenario where each user has a distinct cache and under the assumption of having a *degraded* channel, such that users can be sorted from the weakest to the stronger (whereas in this work we consider a (not degraded) broadcast channel where there exist $\Lambda \leq K$ different cache states and each user can store one of such states). That work proposes novel lower and upper bounds for the degraded BC, applicable to e.g. the erasure or the Gaussian BC.

It is interesting to note that, even if the scenario considered in [57], [58] and the one considered here are markedly dissimilar[1], and the placement, delivery schemes, as well as the novel proposed converse result are utterly different, the resulting memory optimization solutions have an analogous shape. Nevertheless, both our work here and the results in [58] highlight the importance of a carefully designed memory allocation for systems with heterogeneous elements.

C. Contributions

In this work we consider a shared-cache setting where each cache serves an arbitrary number of users and where the size of each such cache can be optimized — under a sum cache-size constraint — as a function of the topology.[2] We focus on a topology-aware scenario where the cache occupancy is known during the memory-allocation phase. In particular:

- We construct a novel converse bound based on index coding that allows us to overcome the challenges posed by the considered heterogeneous cache-aided setting. This converse bound is able to capture the complex influence of the asymmetry in the setting through a non-trivial and entangled combination of index coding bounds. This comes at a time when most of the converse bounds for cache-aided networks build on mathematical machinery that depends heavily on symmetry properties. Such bounding techniques, while working well for idealized symmetric settings, can suffer in the presence of heterogeneity, which is known to cause additional challenges in achieving converse tightness. We believe that our proposed converse can shed light on this type of bounds and on their applicability on heterogeneous settings.
- For this setting, we will show that our proposed memory allocation, cache placement, and delivery scheme achieve the information theoretic optimal performance under some regularity assumptions.
- As a by-product, we prove that asymmetry can be beneficial and asymmetric settings can outperform the well-studied symmetric setting. As a matter of example, for a given number of caches, total number of users and total cache size, we show that the uniform cache occupancy (where each cache serves the same number of users) is the one that results in the highest delivery time, whereas the lowest delivery time is achieved for the case where all caches but one serve a single user each, and the remaining cache serves the rest of the users.
- Subsequently, motivated by the possibility of having a cache occupancy that varies with time, we consider a scenario of partial topology awareness where there is a mismatch between the perceived/estimated cache occupancy during the memory allocation phase and the true cache occupancy experienced during the actual delivery phase. This prior estimate of the topology, which will define the memory allocation, may for example reflect long-term statistical knowledge of this occupancy. For this scenario of imperfect knowledge of topology, we will show that, under a specific memory allocation and cache placement that are both designed according to the *expected* number of users connected to each cache, the proposed delivery scheme is exactly optimal. Finally, in addition to providing exact information-theoretic optimality expressions, we also proceed to elevate our problem to the stochastic setting by studying the case where the number of users connected to each cache follows a random distribution, for which we compare our derived results with state-of-the-art schemes, namely the topology-agnostic scheme in [29] and the scheme from [59], which allows us to stress the fundamental importance of optimized memory allocation in such heterogeneous cache-aided networks.

D. Outline and Notation

The rest of the paper is organized as follows. Section II describes the system model, the formal problem definition, and various mathematical preliminaries, while Section III provides an illustrative example of the proposed scheme for the topology-aware scenario. After that, we present our main results in Section IV. For the topology-aware scenario, the achievable scheme is described in Section V, while the converse bound can be found in Section VI. Subsequently, we present in Section VII the achievable scheme and converse bound for the scenario with imperfect knowledge of topology, as well as various numerical evaluations. Finally, our conclusions are discussed in Section VIII, while key mathematical derivations are presented in the appendices.

Notation: For $\Lambda \in \mathbb{N}$, we use $[\Lambda] \triangleq \{1, 2, \ldots, \Lambda\}$ and $[\Lambda]_0 \triangleq \{0, 1, 2, \ldots, \Lambda\}$. For any set $\mathcal{T}$, we define the set of all k-combinations of $\mathcal{T}$ as $C_k^{\mathcal{T}} \triangleq \{\tau : \tau \subseteq \mathcal{T}, |\tau| = k\}$, while we denote the powerset of $\mathcal{T}$ as $2^{\mathcal{T}}$. For any $n \in \mathbb{N}$, we use $\mathcal{S}_n$ to denote the symmetric group of all permutations of $[n]$, i.e., $\mathcal{S}_n$ is the set of all bijections $\sigma : [n] \to [n]$. For an ordered set τ, we will refer to the j-th element of τ as $\tau(j)$. The XOR operation is here denoted by the symbol $\bigoplus$. We denote by $\mathrm{round}()$ the function rounding a real number to the nearest integer. Furthermore, for a set $\mathcal{X}$ and $k \in \mathcal{X}$, and for $\{\mathrm{a}_k\}$ being a real-valued discrete sequence, we will use $\mathrm{Conv}_{k\in\mathcal{X}}(\mathrm{a}_k)$ to denote the (real and continuous) lower convex envelope of the points $\{(k, \mathrm{a}_k) | k \in \mathcal{X}\}$.

[1] In such works, the main parameter that defines the scenario is the relative channel strength between different users. In our case of study, the main parameter is the amount of users associated to each cache.

[2] As it will become evident from the formal description of the system model, our derived schemes and results directly apply to the isomorphic coded caching problem where users having access to their own dedicated cache can request any number of files.

II. System Model

We present the system model for the two operating scenarios, as well as the corresponding performance metric. The system model is fully described for the topology-aware scenario, while for the scenario corresponding to imperfect knowledge of topology the description focuses on highlighting the differences with respect to the previous scenario.

A. Topology-aware scenario

We consider a cache-aided network where a transmitter (TX) with access to a library of N unit-sized files $W^{(1)}, W^{(2)}, \dots, W^{(N)}$ is connected via a shared-link broadcast channel to K users ($N \geq K$), and each of these users is connected to *one* of Λ different caches. The size of each cache $\lambda \in \{1, 2, \dots, \Lambda\}$ is a design parameter denoted by $M_\lambda \in (0, N]$ (in units of file), adhering to a cumulative (sum) cache-size constraint defined by $M_\Sigma \triangleq \sum_{\lambda=1}^{\Lambda} M_\lambda$. We define the normalized size of cache λ as $\gamma_\lambda \triangleq \frac{M_\lambda}{N}$, such that the corresponding cache redundancy takes the form of the normalized sum cache-size constraint

$$t \triangleq \sum_{\lambda=1}^{\Lambda} \gamma_\lambda = \frac{M_\Sigma}{N}. \tag{1}$$

We consider the general scenario where the channel capacity is normalized to one file per unit of time, and we assume that users can access the content of their associated cache at zero cost.

Each cache λ is connected to a disjoint set of users $\mathcal{U}_\lambda \subset [K] \triangleq \{1, \dots, K\}$, where these sets form a partition $\mathcal{U} = \{\mathcal{U}_1, \dots, \mathcal{U}_\Lambda\}$ of $[K]$. The *occupancy* of each cache λ describes the number of users connected to this cache, and it is denoted by $L_\lambda = |\mathcal{U}_\lambda| \in \mathbb{N}$. The corresponding *cache occupancy* vector is denoted as

$$\boldsymbol{L} \triangleq (L_1, \dots, L_\Lambda), \tag{2}$$

where naturally $\sum_{\lambda=1}^{\Lambda} L_\lambda = K$, and where we assume without loss of generality that $L_1 \geq L_2 \geq \cdots \geq L_\Lambda$. With a slight abuse of notation, whenever needed, we will use $\boldsymbol{L}$ in its set form, to represent $\{L_\lambda\}_{\lambda=1}^{\Lambda}$. Figure 1 provides a schematic representation of our setting. Hereinafter, we will denote this setting as the $(t, \boldsymbol{L})$ *shared-cache BC network*, where t and $\boldsymbol{L}$ has been defined in (1) and (2), respectively.

The system works in three different phases.

1) A *memory allocation phase* during which the knowledge of the cache occupancy vector $\boldsymbol{L}$ is used to allocate the total memory M_Σ to the caches, yielding the allocated size set $\{\gamma_\lambda\}_{\lambda=1}^{\Lambda}$.
2) A *cache placement phase* during which each cache λ — of allocated size γ_λ — is filled with content $\mathcal{Z}_\lambda$ from the library, according to a certain strategy $\boldsymbol{\mathcal{Z}}_{\boldsymbol{L}} = (\mathcal{Z}_1, \dots, \mathcal{Z}_\Lambda)$, where the lower index $\boldsymbol{L}$ highlights the dependency of the cache placement on the cache occupancy vector $\boldsymbol{L}$. Hereinafter, we will often omit the lower index $\boldsymbol{L}$ whenever there is no possible ambiguity. In this work, as is often common, we focus only on *uncoded cache placement schemes* where each packet stored in a cache can be traced back to the library (i.e., comes directly from the library, without any coding). Finally, let us again recall that $\boldsymbol{L}$ is known during this phase.
3) A *delivery phase* that starts with each user $k \in [K]$ requesting a single library file $W^{(d_k)}$. Once the demand vector $\mathbf{d} \triangleq (d_1, d_2, \dots, d_K)$ of requested file indices is known, the server begins to deliver each requested file to its corresponding user.

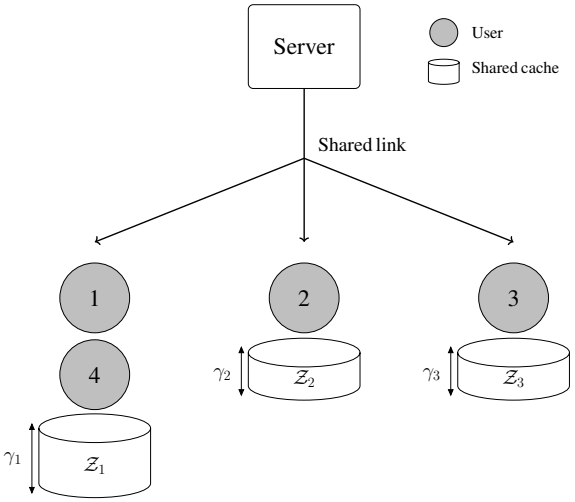


Fig. 1: The shared-cache setting with $\boldsymbol{L} = (2, 1, 1)$.

B. Scenario with imperfect knowledge of topology

In this scenario, we consider a similar setting to the $(t, \boldsymbol{L})$ shared-cache BC network described before, with the only difference being that the number of users that will be connected to each cache during delivery phase is not known during the memory allocation phase and the subsequent cache placement phase. Instead of the knowledge of the true vector $\boldsymbol{L}$, we assume that the first two phases are designed according to another (perhaps estimated) cache occupancy vector $\bar{\boldsymbol{L}} = (\bar{L}_1, \bar{L}_2, \dots, \bar{L}_\Lambda)$ that is generally not equal to $\boldsymbol{L}$. We will denote this scenario as the $(t, \bar{\boldsymbol{L}}, \boldsymbol{L})$ shared-cache BC network. For any cache $\lambda \in [\Lambda]$, the value of $\bar{L}_\lambda$ can represent an imperfect prediction or an expectation of the number of users that will be connected to this cache during the delivery phase. We also allow the sum $\sum_{\lambda=1}^{\Lambda} \bar{L}_\lambda$ to be arbitrary[3] and not necessarily equal to $\sum_{\lambda=1}^{\Lambda} L_\lambda = K$.

C. Problem definition

We consider the standard *rate* metric that has been commonly used in coded caching literature [2], [5], [60], which we hereinafter refer to as the *delivery time*, and which we denote by T. For any given *uncoded cache placement* scheme $\boldsymbol{\mathcal{Z}}$, any cache occupancy vector $\boldsymbol{L}$, and any given demand $\mathbf{d}$, we define $T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L})$ as the minimum delivery time (minimized over all delivery schemes) that guarantees delivery of the desired files $W^{(d_k)}$ to all users $k \in [K]$. Under the assumption of uncoded cache placement, our goal is to characterize the

[3]We note that the integers $\bar{L}_\lambda, \lambda \in [\Lambda]$, are strictly positive.

minimum worst-case delivery time over all memory-allocation strategies and all placement-and-delivery schemes, i.e., we aim to characterize

$$T^*(t,\boldsymbol{L}) \triangleq \min_{\boldsymbol{\mathcal{Z}}} \max_{\mathbf{d}} T^*(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L}) \tag{3}$$

as a function of t and $\boldsymbol{L}$. We omit hereinafter the dependence of T^* on $(t,\boldsymbol{L})$ when there is no possible ambiguity. For any cache occupancy vector $\boldsymbol{L}$, we also define the optimal cache placement $\boldsymbol{\mathcal{Z}}^*_{\boldsymbol{L}}$ as $\boldsymbol{\mathcal{Z}}^*_{\boldsymbol{L}} \triangleq \operatorname{argmin}_{\boldsymbol{\mathcal{Z}}} \max_{\mathbf{d}} T^*(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L})$. Next, we present the definition of what is often referred to as regular cache placement.

Definition 1. A cache placement scheme is said to be *regular* if each bit of the library is repeated the same number of times throughout the different caches.

It is easy to see that any regular cache placement naturally implies that $t \in [\Lambda]$. In the context of non-uniform $\boldsymbol{L}$ and heterogeneous $\{\gamma_\lambda\}_{\lambda=1}^{\Lambda}$, the concept of the sum-DoF in cache-aided networks [24] naturally generalizes to

$$\mathrm{DoF} \triangleq \frac{K - \sum_{\lambda=1}^{\Lambda} \gamma_\lambda L_\lambda}{T}, \tag{4}$$

reflecting the rate of delivery of the non-cached desired information.

For the scenario with imperfect topology knowledge, where the memory allocation and cache placement is done on the basis of the available information $\bar{\boldsymbol{L}}$, we will characterize the optimal delivery time

$$T^*(t,\boldsymbol{L},\bar{\boldsymbol{L}}) \triangleq \max_{\mathbf{d}} T^*(\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}},\mathbf{d},\boldsymbol{L}) \tag{5}$$

as a function of t, $\boldsymbol{L}$ and $\bar{\boldsymbol{L}}$.

D. Mathematical preliminaries

Before presenting our main results, we include some mathematical preliminaries that will be useful for the derivation of our results. We first recall the well-known *elementary symmetric functions* [61], also known as elementary symmetric polynomials. These functions have already appeared in some works about cache allocation for coded caching [57], [58] and will be used extensively in our proofs. We recall that we denote the set of all k-combinations of a set $\mathcal{X}$ as $C_k^{\mathcal{X}} \triangleq \{\tau : \tau \subseteq [\mathcal{X}], |\tau| = k\}$.

Definition 2 (Elementary symmetric functions)**.** For any n-set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and any $k \in \{1,\ldots,n\}$, the k-th *elementary symmetric function* $e_k(\mathcal{X})$ is defined as

$$e_k(\mathcal{X}) \triangleq \sum_{q \in C_k^{\mathcal{X}}} \prod_{j=1}^{k} x_{q(j)} \tag{6}$$

where $e_0(\mathcal{X}) \triangleq 1$.

Next, we present some known (or otherwise here derived) basic properties of such elementary symmetric functions.

Property 1. *For any elementary symmetric function $e_k(\mathcal{X})$, $k \in \{1,2,\ldots,n\}$, on any set $\mathcal{X} \triangleq \{x_1,\ldots,x_n\}$, it holds that*

$$e_k(\mathcal{X}) = e_k(\mathcal{X} \setminus \{x_i\}) + x_i e_{k-1}(\mathcal{X} \setminus \{x_i\}), \tag{7}$$

Proof. The proof of Property 1 is straightforward and follows from the definition of the elementary symmetric functions (cf. Def. 2) [61], since (7) just represents that the sum of all products of k distinct elements in set $\mathcal{X}$ (i.e., $e_k(\mathcal{X})$) can be split in two terms: (i) The sum of all such products of k distinct elements that *do not include* x_i (i.e., $e_k(\mathcal{X} \setminus \{x_i\})$), and (ii) the sum of all products of k distinct elements that *do include* x_i, where the latter can be written as $x_i e_{k-1}(\mathcal{X} \setminus \{x_i\})$. □

Property 2. *For any elementary symmetric function $e_k(\mathcal{X})$, $k \in \{1,2,\ldots,n\}$, on any set $\mathcal{X} \triangleq \{x_1,\ldots,x_n\}$, the following equality holds for any subset $\phi \subset \mathcal{X}$.*

$$\sum_{x_i \in \{\mathcal{X}\setminus\phi\}} x_i \cdot e_{k-1}(\mathcal{X} \setminus \{x_i\}) = \sum_{q \in C_k^{\mathcal{X}}} \prod_{j=1}^{k} x_{q(j)} \cdot |\{q \setminus \phi\}|, \tag{8}$$

Proof. The proof is relegated to Appendix I. □

Corollary 1. *For any elementary symmetric function $e_k(\mathcal{X})$, $k \in \{1,2,\ldots,n\}$, on any set $\mathcal{X} \triangleq \{x_1,\ldots,x_n\}$, it holds that*

$$\sum_{i \in [|\mathcal{X}|]} x_i e_{k-1}(\mathcal{X} \setminus \{x_i\}) = k \cdot e_k(\mathcal{X}), \tag{9}$$

Proof. Corollary 1 follows directly from Property 2 after setting $\phi = \emptyset$. □

III. Memory-Allocation, Placement and Delivery: an Illustrative Example

Before presenting the main results, we provide an example that illustrates the main ideas behind the proposed general scheme in the topology-aware scenario.

We consider a particular instance of the $(t,\boldsymbol{L})$ shared-cache BC network system presented in Section II and represented in Fig. 1. In particular, we consider an instance with $N = 6$ files, $\Lambda = 3$ caches, and $K = 6$ users associated to the different caches according to the cache occupancy vector $\boldsymbol{L} = (3,2,1)$. We assume that a sum memory of $M_\Sigma = 12$ units of file — corresponding to $t = 2$ times the size of the library — is available to be allocated across the caches. In the first phase, we use the knowledge of $\boldsymbol{L}$ to allocate fractions

$$\gamma_1 = \frac{L_1L_2 + L_1L_3}{L_1L_2 + L_1L_3 + L_2L_3} = \frac{9}{11} \tag{10}$$

$$\gamma_2 = \frac{L_1L_2 + L_2L_3}{L_1L_2 + L_1L_3 + L_2L_3} = \frac{8}{11} \tag{11}$$

$$\gamma_3 = \frac{L_1L_3 + L_2L_3}{L_1L_2 + L_1L_3 + L_2L_3} = \frac{5}{11} \tag{12}$$

of the library to caches $1,2,3$, respectively. Seeing that $\frac{9}{11} + \frac{8}{11} + \frac{5}{11} = t = 2$ allows us to verify that the cumulative cache size is not exceeded. In the subsequent caching phase, we split each file $W^{(n)}$ ($n \in [6]$) into 11 equally-sized subfiles which we label as $W^{(n)}_{\tau,m_\tau}$, where each pair (τ,m_τ) is taken from the following set:[4]

$$\big\{(12,1),\ (12,2),\ (12,3),\ (12,4),\ (12,5),\ (12,6),\\ (13,1),\ (13,2),\ (13,3),\ (23,1),\ (23,2)\big\}. \tag{13}$$

[4]For the sake of being concise, we have used a compact notation in (13) and throughout this example, such that we use for example $(12,1)$ instead of the less concise notation $(\{1,2\},1)$.

In the above subfile labeling, the first index τ represents the set of caches that will store the associated subfile. On the other end, the second index m_τ is a mere counter that helps us differentiate subfiles with the same first index τ. Hence, for example, $W^{(n)}_{12,4}$ is the fourth subfile out of the subfiles of $W^{(n)}$ that are exclusively stored in caches 1 and 2 (6 subfiles in this case). Then, for each $n \in [6]$, each cache λ stores those subfiles whose first index τ includes λ. Consequently, in our example, the content of each cache is:

$$\begin{aligned}\mathcal{Z}_1 = \{&W^{(n)}_{12,1}, W^{(n)}_{12,2}, W^{(n)}_{12,3}, W^{(n)}_{12,4}, W^{(n)}_{12,5}, W^{(n)}_{12,6},\\ &W^{(n)}_{13,1}, W^{(n)}_{13,2}, W^{(n)}_{13,3} : n \in [6]\},\\ \mathcal{Z}_2 = \{&W^{(n)}_{12,1}, W^{(n)}_{12,2}, W^{(n)}_{12,3}, W^{(n)}_{12,4}, W^{(n)}_{12,5}, W^{(n)}_{12,6},\\ &W^{(n)}_{23,1}, W^{(n)}_{23,2} : n \in [6]\},\\ \mathcal{Z}_3 = \{&W^{(n)}_{13,1}, W^{(n)}_{13,2}, W^{(n)}_{13,3}, W^{(n)}_{23,1}, W^{(n)}_{23,2} : n \in [6]\},\end{aligned}$$

which adheres to the aforementioned memory allocation $\gamma_1 = \frac{9}{11}, \gamma_2 = \frac{8}{11}, \gamma_3 = \frac{5}{11}$.

In the delivery phase, we consider the demand vector $\mathbf{d} = (1, 2, 3, 4, 5, 6)$ where $W^{(1)}, W^{(4)}$ and $W^{(6)}$ are each requested by one of the three users associated to cache 1, $W^{(2)}$ and $W^{(5)}$ by the two users associated to cache 2, and $W^{(3)}$ by the user associated to cache 3. For the sake of a more understandable exposition, we re-denote files $\{W^{(n)}\}_{n=1}^{6}$ as $\{A, B, C, D, E, F\}$. We can see that each set $\mathcal{R}_\lambda$ of uncached subfiles wanted by the users of cache λ takes the form

$\mathcal{R}_1$	$\mathcal{R}_2$	$\mathcal{R}_3$
$A_{23,1}$	$B_{13,1}$	$C_{12,1}$
$D_{23,1}$	$E_{13,1}$	$C_{12,2}$
$F_{23,1}$	$B_{13,2}$	$C_{12,3}$
$A_{23,2}$	$E_{13,2}$	$C_{12,4}$
$D_{23,2}$	$B_{13,3}$	$C_{12,5}$
$F_{23,2}$	$E_{13,3}$	$C_{12,6}$

We notice that as a consequence of the proposed heterogeneous memory allocation and cache placement — which we will present in detail in Section V — the number of equi-sized subfiles jointly requested from all the users of each cache remains the same. This fact is key to allowing the transmission of all data during the delivery phase in the form of multicast messages serving $t+1$ users at a time. In our example, these messages will contain information for $t+1 = 3$ different users at a time. The subsequently transmitted 6 XORs that deliver all requested subfiles take the form

$$X_{123}(1) = D_{23,1} \oplus B_{13,1} \oplus C_{12,1} \tag{14}$$

$$X_{123}(2) = A_{23,1} \oplus E_{13,1} \oplus C_{12,2} \tag{15}$$

$$X_{123}(3) = A_{23,2} \oplus B_{13,2} \oplus C_{12,3} \tag{16}$$

$$X_{123}(4) = F_{23,1} \oplus E_{13,2} \oplus C_{12,4} \tag{17}$$

$$X_{123}(5) = F_{23,2} \oplus B_{13,3} \oplus C_{12,5} \tag{18}$$

$$X_{123}(6) = D_{23,2} \oplus E_{13,3} \oplus C_{12,6} \tag{19}$$

and each XOR can be easily decoded in the classical manner described in [2]. Consequently, the total delay is $T = \frac{6}{11}$, which will be shown to be exactly optimal under the assumption of uncoded cache placement. On the other hand, if we had forced equal-sized caches, the best possible performance under the same assumptions would be $T = 1$ (cf. [29]), which almost doubles the delay of the new scheme. We recall that the work in [29] has shown that, without knowledge of the topology during the caching phase, the heterogeneity in the cache occupancy numbers results in an unavoidable reduction in the multicasting gain, which reduces below $t+1$ as the skewness of the cache occupancy vector increases. On the other hand, the current knowledge of the topology allows our scheme to optimize the cache sizes, yielding a symmetry that in turn allows for a constantly full multicasting gain of $t+1$ while also allowing for a local caching gain that, as we will explore later on, interestingly increases with the skewness of the cache occupancy vector.

IV. Main Results

We present in this section our main contributions. Let us start by presenting an information-theoretic converse (lower bound) on the delivery time under uncoded cache placement for the topology-aware scenario described before, which will be explained later on.

Theorem 1. *Under the assumption of uncoded cache placement, the optimal normalized delivery time of the $(t, \boldsymbol{L})$ shared-cache BC network satisfies*

$$T^*(t, \boldsymbol{L}) \geq \mathcal{T}^{(\bar{t}, \boldsymbol{L})}_{low}(t) \tag{20}$$

where $\bar{t} \triangleq \text{round}(t)$, and $\mathcal{T}^{(\bar{t}, \boldsymbol{L})}_{low}(x)$ is defined as

$$\mathcal{T}^{(\bar{t}, \boldsymbol{L})}_{low}(x) \triangleq \text{Conv}_{j \in [\Lambda]_0}\left(\frac{\sum\limits_{q \in C^{[\Lambda]}_{\bar{t}+1}} \frac{\bar{t}+1-|q \cap \tau^\star_j|}{j+1-|q \cap \tau^\star_j|} \prod\limits_{i=1}^{\bar{t}+1} L_{q(i)}}{\sum_{\ell \in C^{[\Lambda]}_{\bar{t}}} \prod\limits_{i=1}^{\bar{t}} L_{\ell(i)}} \right) \tag{21}$$

where $\tau^\star_j$ is given by

$$\tau^\star_j = \begin{cases} \{\emptyset\} & \text{if } j = 0 \\ \{\Lambda - j + 1,\ \Lambda - j + 2,\ \ldots,\ \Lambda\} & \text{if } 1 \leq j < \bar{t} \\ \{1,\ 2,\ \ldots,\ j\} & \text{if } j \geq \bar{t} \end{cases}$$

Proof. The proof is presented in Section VI. □

The heterogeneity of the cache occupancy vector $\boldsymbol{L}$ and the fact that this vector is known during the placement phase jointly introduce a new important challenge in the derivation of the converse bound. As briefly demonstrated in [1], a direct application of the traditional index coding techniques (see. [60], or equivalently see the genie-aided approach of [5]) would result in very loose bounds. The reason for which the known bounds do obtain a loose result is because they generally rely on derivations that require some *symmetry* in the topology of the scenario. What the bound in (21) achieves is to render unnecessary the previously generally employed symmetries, thanks to involved combinatorial derivations, thus allowing us to capture the heterogeneity of the system.

In conjunction with the developed achievable coded caching scheme of Section V, this new converse becomes exactly tight under the basic assumptions of regular and uncoded cache

placement (cf. Definition 1), which are common properties of the content placement for most of the known coded caching schemes [2], [5], [43], [62], [63]. Let us now present the converse result for the case where we restrict ourselves to the common *regular* placement from Definition 1.

Theorem 2. *Under the assumption of regular and uncoded cache placement, the optimal normalized delivery time of the $(t, \boldsymbol{L})$ shared-cache BC network satisfies*

$$T^*(t, \boldsymbol{L}) \geq \mathcal{T}^{(t,\boldsymbol{L})}_{low,\, reg}(t) \tag{22}$$

where $\mathcal{T}^{(t,\boldsymbol{L})}_{low,\, reg}(x)$ is defined as

$$\mathcal{T}^{(t,\boldsymbol{L})}_{low,reg}(x) \triangleq \frac{\sum_{q\in C^{[\Lambda]}_{t+1}} \prod_{j=1}^{t+1} L_{q(j)}}{\sum_{q\in C^{[\Lambda]}_{t}} \prod_{j=1}^{t} L_{q(j)}} = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}. \tag{23}$$

Proof. The proof is presented in Section VI. □

Note that for the standard scenario with dedicated caches, where $L_i = 1$ for any $i \in [\Lambda]$ and $\Lambda = K$, (23) reduces to the optimal delivery time from [5]. Next, we present the achievable delivery time of our proposed scheme, which will be described in detail in Section V.

Lemma 1. *For the $(t, \boldsymbol{L})$ shared-cache BC network with Λ shared caches, normalized sum-cache constraint t, and cache occupancy vector $\boldsymbol{L}$, the worst-case delivery time*

$$T(t, \boldsymbol{L}) = \mathrm{Conv}_{t\in[\Lambda]_0}\left(\frac{\sum_{\lambda=1}^{\Lambda} L_\lambda(1-\gamma_\lambda)}{t+1}\right) \tag{24}$$

is achievable, where the memory allocation $\{\gamma_\lambda\}_{\lambda=1}^{\Lambda}$ is given by

$$\gamma_\lambda = \frac{L_\lambda \cdot e_{t-1}(\boldsymbol{L}\setminus\{L_\lambda\})}{e_t(\boldsymbol{L})}. \tag{25}$$

Proof. The placement and delivery schemes are presented in Section V. □

We immediately notice that the delay in (24) directly implies that $\mathrm{DoF} = t+1$, which is an improvement over the case where $\boldsymbol{L}$ is unknown to the placement phase. In fact, we know from [29] that, without knowledge of $\boldsymbol{L}$ during the (uncoded) cache placement, $\mathrm{DoF} = t+1$ can be achieved only in the uniform case where we have $\frac{K}{\Lambda}$ users per cache, and that any non-uniformity in $\boldsymbol{L}$ strictly forces a DoF penalty. The above lemma shows that knowledge of the profile $\boldsymbol{L}$ allows for a redesigned and skewed memory allocation that, in turn, simultaneously allows a better local caching gain and a higher sum-DoF. To clarify this, a strategy that allocates more memory to more loaded caches automatically allows for higher local caching gains than a uniform memory allocation across the caches. At the same time, such heterogeneous allocation allows for multicasting messages that always serve $t+1$ users at a time. These observations lead to the surprising fact that, for a fixed number of users K and Λ non-empty caches, the uniform cache occupancy vector $\boldsymbol{L} = (\frac{K}{\Lambda}, \frac{K}{\Lambda}, \ldots, \frac{K}{\Lambda})$ is the one that results in the highest delivery time, while the lowest delivery time is achieved for $\boldsymbol{L} = (K-\Lambda+1, 1, 1, \ldots, 1)$.

Observation 1. *It is interesting to observe that, for any given normalized total cache size t, the memory allocation $\{\gamma_\lambda\}_{\lambda=1}^{\Lambda}$ given in Lemma 1 coincides with the one of the scheme proposed in [57] (see also [58]) for a cache-aided setting with fixed unequal channel capacities. In particular, the work in [57] by Tang et al. considers a wireless broadcast channel where the link between the transmitter and cache-aided receiver λ has normalized capacity R_λ, and where these capacities are known during the cache allocation and placement phases. Assuming that each user requests only one file, the authors proposed a scheme which requires an unequal cache size allocation that coincides with the one in (25), where L_λ would be replaced by the inverse of the rate of user λ, i.e. $R_\lambda = \frac{1}{L_\lambda}, \forall \lambda \in [\Lambda]$. Despite the different nature of these two problems, we can observe that in the unequal link rate setting a user i connected to the server through a link of capacity R_i is reminiscent of the cache serving L_i users in our shared-cache setting. Similar analogies could be done between our contribution and the aforementioned work in [58], where they studied the cache-aided degraded broadcast channel that includes the setting with fixed unequal channel capacities of [57].*

We have presented the proposed lower bound and achievable scheme. In the next theorem, we show that they match and that the achievable delivery time is exactly optimal under the assumption of regular placement.

Theorem 3. *For the $(t, \boldsymbol{L})$ shared-cache BC network, the achievable delivery time $T(t, \boldsymbol{L})$ in (24) is exactly optimal under the assumption of uncoded and regular cache placement.*

Proof. First, we note that we can write the lower-bound in Theorem 2 as $\mathcal{T}^{(t,\boldsymbol{L})}_{low,\, reg}(t) = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}$. Then, it follows that

$$\begin{aligned} T^*(t, \boldsymbol{L}) &\geq \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} \\ &\overset{(a)}{=} \frac{1}{t+1}\frac{\sum_{\lambda=1}^{\Lambda} L_\lambda \cdot e_t(\boldsymbol{L}\setminus\{L_\lambda\})}{e_t(\boldsymbol{L})} \\ &\overset{(b)}{=} \frac{1}{t+1}\frac{\sum_{\lambda=1}^{\Lambda} L_\lambda \cdot (e_t(\boldsymbol{L}) - L_\lambda e_{t-1}\boldsymbol{L}\setminus\{L_\lambda\})}{e_t(\boldsymbol{L})} \\ &\overset{(c)}{=} \frac{\sum_{\lambda=1}^{\Lambda} L_\lambda(1-\gamma_\lambda)}{t+1} = T(t, \boldsymbol{L}), \end{aligned} \tag{26}$$

where (a) follows from using Corollary 1, (b) follows from employing Property 1, and (c) is because of (25). $T(t, \boldsymbol{L})$ represents the achievable delivery time of Lemma 1, which concludes the proof of Theorem 3. □

These optimality results are extended beyond the assumption of regular placement in Section VI.

The previous results show how the knowledge of the cache occupancy vector $\boldsymbol{L}$ considerably impacts the performance of coded caching, and that we can derive optimal schemes that leverage this knowledge to improve considerably both local and global caching gains. Of course, there may be scenarios where having perfect knowledge of the topology during the placement phase is not feasible, and instead one has to rely on some noisy, imperfect, or average information about the

said topology. Because of that, we now shift the focus to the scenario in which such topology knowledge is imperfect or noisy.

For this setting, the next theorem describes the optimal delivery time for the scenario where, during the memory allocation and cache placement phases, the assumed cache occupancy vector does not match the actual vector that materializes during the subsequent delivery phase.

Theorem 4. *For the $(t, \bar{\boldsymbol{L}}, \boldsymbol{L})$ scenario with imperfect topology knowledge, the delivery time*

$$T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L}) = \frac{\sum_{q \in C_{t+1}^{[\Lambda]}} \max_{\lambda \in q} L_\lambda \prod_{j=1}^{t} \bar{L}_{\tau_\lambda(j)}}{e_t(\bar{\boldsymbol{L}})}, \quad (27)$$

is exactly optimal under the assumption that the memory allocation and content placement is applied under the premise that the topology during the delivery phase remains the same as the available knowledge $\bar{\boldsymbol{L}}$, where τ_λ is defined as $\tau_\lambda \triangleq \{q \backslash \{\lambda\}\}$.

Proof. The achievable delivery scheme and the converse bound are presented in Section VII. □

Theorem 4 refers to the case in which, during the first 2 phases, the server assumes that users will be connected to the caches in the delivery phase according to cache occupancy vector $\bar{\boldsymbol{L}}$, but it turns out that the actual cache occupancy vector will finally be $\boldsymbol{L}$. It is easy to conclude that such delivery time is higher than the optimal delivery time for the scenario where users actually show up according to $\bar{\boldsymbol{L}}$ (as expected by the server) when the total number of users is the same. This results is stated in the following corollary.

Corollary 2. *For the case in which $\sum_{\lambda=1}^{\Lambda} \bar{L}_\lambda = \sum_{\lambda=1}^{\Lambda} L_\lambda$, the scenario where $\boldsymbol{L} = \bar{\boldsymbol{L}}$ achieves the lowest delivery time, i.e., $T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L}) \geq T^*(t, \boldsymbol{L})$.*

The above corollary is a direct derivation from the definitions of $T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L})$ and $T^*(t, \boldsymbol{L})$ in the system model.

Corollary 3. *For a given $\bar{\boldsymbol{L}}$ where $\bar{L}_1 \geq \bar{L}_2 \geq \cdots \geq \bar{L}_\Lambda$, the worst occupancy vector during delivery phase is given by*

$$\underset{\boldsymbol{L}}{\operatorname{argmax}} \; T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L}) = \{1, 1, \ldots, 1, K - \Lambda + 1\}, \quad (28)$$

i.e., $L_\Lambda = K - \Lambda + 1$ and $L_i = 1$ for any $i \in [\Lambda - 1]$.

Proof. The proof is relegated to Appendix III-A. □

Theorem 4 refers to the delivery time for a specific value $\boldsymbol{L}$ of the occupancy vector at delivery phase. We can easily derive the expectation of the delivery time for the case where $\boldsymbol{L}$ follows a certain probability distribution, as stated in the following corollary. Let us define the Λ-dimensional space of non-negative numbers as $\mathcal{N}_\Lambda = \{n_1, n_2, \ldots, n_\Lambda \mid n_i \in \{\mathbb{N}_0\}, \; \forall i \in [\Lambda]\}$, where $\mathbb{N}_0$ denotes the set of Natural numbers including 0, and let $\mathrm{Prob}(\boldsymbol{L})$ represent the probability of a specific $\boldsymbol{L} \in \mathcal{N}_\Lambda$ to be realized during delivery phase.

Corollary 4. *In the scenario where the occupancy vector at delivery phase $\boldsymbol{L}$ follows a probability distribution, the optimal expected delivery time when the memory allocation and content placement are applied on the basis of the placement-phase information $\bar{\boldsymbol{L}}$ is given by*

$$E_{\boldsymbol{L}}[T^*(t, \bar{\boldsymbol{L}})] = \sum_{\boldsymbol{L} \in \mathcal{N}_\Lambda} \mathrm{Prob}(\boldsymbol{L}) T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L}). \quad (29)$$

Note that, in scenarios where only long-term statistical information is available, e.g., when expected values are known, it is common to consider the strategy of acting as if this information was perfect and true for any realization. In Section VII-D, we will show through some numerical evaluations how $T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L})$ behaves on average when $\boldsymbol{L}$ is a realization of Λ independent random variables. Finally, we note that the scheme achieving the performance in Theorem 4 creates XORs that do not always serve $t+1$ users, which will be explained in Section VII. This is clearly a drawback of the fact that the memory allocation and cache placement phases are designed according to a cache occupancy vector that is different from the one that materializes in the delivery phase.

So far, we have optimized the memory allocation and cache placement to minimize the delivery time for a given occupancy vector $\boldsymbol{L}$. To conclude, we analyze the delivery time for the worst-case occupancy vector, i.e., instead of solving (3), we consider

$$T_{wc}^*(t) \triangleq \min_{\boldsymbol{\mathcal{Z}}} \max_{\boldsymbol{L}} \max_{\mathbf{d}} T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L}) \quad (30)$$

where $T_{wc}^*(t)$ denotes the worst-case occupancy vector delivery time.

Lemma 2. *For the $(t, \boldsymbol{L})$ shared-cache scenario, under the assumption that $\sum_{i=1}^{\Lambda} L_i = K$ and $\Lambda \mid K$, the delivery time for the worst-case occupancy vector ($T_{wc}^*(t)$) is lower bounded by the delivery time of the standard symmetric case where $L_i = \frac{K}{\Lambda}$ for all $i \in [\Lambda]$.*

Proof. The proof is relegated to Appendix III-B. □

Note that Lemma 2 is intuitive from the fact that the optimal memory allocation *adapts* to the asymmetry of the occupancy vector: Caches serving more users compensate that fact by increasing the size of its memory. Thus, the case minimizing the worst occupancy vector case is the symmetric case, which minimizes the maximum possible distance w.r.t. the most extreme occupancy vectors. This result is also intuitive from Corollary 3.

Comparison with related scenarios

The aforementioned analogies with other settings (cf. Observation 1), such as the cache-aided degraded BC or the BC where each user can demand several files, raise the questions of whether our results can be applied to such scenarios and, conversely, whether there is any overlap of results. In the following, we provide a brief discussion on this topic.

First, the shared-cache scenario here considered is a one-to-one mapping to the standard dedicated-cache BC as in [2] where users are allowed to demand more than one file and no file is requested twice. Thus, all results here presented apply to that scenario. To prove this, we refer to Fig. 1 and we note that, in our setting, two users sharing the same cache content will never be served simultaneously, as that would create

interference and both users could not decode their packets. Because of that, it also holds that each user is able to obtain the files of all the other users with which it shares the cache content. Consequently, the transmission to the subset of users connected to cache i, $\mathcal{U}_i$, who demand files $\{W^{(d_u)}\}_{u\in\mathcal{U}_i}$, is equivalent to the transmission to a single user with dedicated cache i that requested all files in $\{W^{(d_u)}\}_{u\in\mathcal{U}_i}$, and the lower and upper bounds here presented can be applied to that setting just by defining L_i as the number of files that user i requests.

With respect to the unequal link strength scenario from [57], the authors present an achievable scheme whose memory allocation matches the one of our achievable scheme (as mentioned in Observation 1), although the process and design of the algorithms considerably differs. The converse result in [57] is a loose bound, whose gap is proportional to $12\frac{R_{\max}}{R_{\min}}$, where $R_{\max}, R_{\min}$ denote the maximum and the minimum link capacities, respectively. Applying our converse approach to such scenario would close this gap; however, the derivation of the converse is not direct, as we require to prove that the models describing both scenarios are analogous.

The most interesting work is [58]. They consider the cache-aided degraded BC with dedicated caches. In fact, the model in [58] contains the scenario analyzed in [57]. The authors in [58] derive lower and upper bounds for the rate-memory trade-off. While a similar memory allocation strategy arises in the achievable scheme, the converse results and derivation are utterly different from our contributions. Furthermore, their generic bound does not have a close-form solution, in the sense that it is required to find the optimal choice of auxiliary variables to find the best bound, for each possible subset of coefficients. Both converse results are also difficult to compare, since [58] measures rate/capacity while we measure delivery time. The found analogies motivate further analysis on whether the tools derived in this work are applicable to such unequal-rate BC scenarios, although this fails out of the scope of this manuscript.

To conclude, we take a look at how to export the results here presented to multi-antenna scenarios. We note that this work is dedicated to single antenna BC channels, and we provide this result on multi-antenna transmission to show that the method here derived can be applied in different settings.

Let us thus consider the setting involving a server with multiple (C) transmitting antennas, where now, unlike the work in [29], the cache sizes can be optimized, thus revealing an even more powerful interplay between multiplexing and coded caching gains.

Lemma 3. *For the $(t, \boldsymbol{L}, C)$ multi-antenna shared-cache BC network where $L_\lambda \geq C,\ \forall\lambda\in[\Lambda]$, the worst-case delivery time*

$$T(t,\boldsymbol{L},C) = \frac{1}{C}\,\mathrm{Conv}_{t\in[\Lambda]_0}\left(\frac{\sum_{\lambda=1}^{\Lambda} L_\lambda(1-\gamma_\lambda)}{t+1}\right) \tag{31}$$

is optimal under the assumptions of uncoded and regular cache placement.

Proof. The converse bound directly follows from the converse bound developed in Section VI where Lemma 4 is adapted to the multi-antenna scenario by means of the bound in Lemma 1 of [29]. Similarly, the achievable scheme is an adaptation of the one developed in Section V, with the difference that in the delivery phase each subfile is further divided in C chunks, thus allowing to serve $C(t+1)$ users simultaneously through the use of zero-forcing precoding, as in the achievable scheme developed in [29]. The details of both converse bound and achievability are omitted for the sake of being concise. □

V. Achievable Scheme

In this section, we present our caching and delivery scheme, and we provide an analysis of its performance. This analysis allows us to prove Lemma 1 from the characterization of the achievable delivery time. We recall that the said achievable delivery time is in turn proven optimal in Theorem 3.

In this section, we present the scheme for integer values of t, $t \in \{1, 2 \ldots, \Lambda\}$, while the case with non-integer t is optimally handled by memory-sharing (cf. [2]), and it is presented in Appendix II-A.

A. Memory Allocation and Cache Placement

We first split each file $W^{(n)}, n \in [N]$, into

$$S = e_t(\boldsymbol{L}) = \sum_{\tau\in C_t^{[\Lambda]}} \prod_{j=1}^{t} L_{\tau(j)}, \tag{32}$$

subfiles of equal size, which corresponds to the subpacketization requirement of the achievable scheme, such that $W^{(n)}$ is partitioned as

$$W^{(n)} = \left\{W^{(n)}_{\tau,1}, W^{(n)}_{\tau,2}, \ldots, W^{(n)}_{\tau,|A_\tau|} \mid \tau \in C_t^{[\Lambda]}\right\}$$

where $A_\tau \triangleq \{1, 2, \ldots, \prod_{j=1}^{t} L_{\tau(j)}\}$. Afterwards, each cache $\lambda \in [\Lambda]$ stores in its memory all subfiles $W^{(n)}_{\tau,m_\tau}, m_\tau \in A_\tau$, whose first subscript τ includes λ, which results in the following cache content.

$$\mathcal{Z}_\lambda = \left\{W^{(n)}_{\tau,m_\tau} \mid W^{(n)}_{\tau,m_\tau} \in W^{(n)}, \tau \ni \lambda, m_\tau \in A_\tau, n \in [N]\right\}.$$

This automatically yields the memory allocation

$$\gamma_\lambda = \frac{L_\lambda \cdot e_{t-1}(\boldsymbol{L}\setminus\{L_\lambda\})}{e_t(\boldsymbol{L})}, \quad \lambda \in [\Lambda]. \tag{33}$$

A detailed explanation on how to obtain (33) is presented in Appendix II-B.

This same placement also assures that each subfile is cached in exactly t caches (because each τ satisfies $|\tau| = t$), which guarantees the sum memory constraint in (1). This memory constraint can also be verified by noting that

$$\sum_{\lambda=1}^{\Lambda}\gamma_\lambda = \sum_{\lambda=1}^{\Lambda}\frac{L_\lambda \cdot e_{t-1}(\boldsymbol{L}\setminus\{L_\lambda\})}{e_t(\boldsymbol{L})} = t \tag{34}$$

where the last step follows directly from Property 1 of the elementary symmetric functions.

Observation 2. *Let us shed light on the structure of the memory allocation. If we want to obtain a DoF of $t+1$, we need that every subfile that is stored in cache λ is also stored in other $t-1$ caches, and this must be true for any*

file. Let us consider a specific set of caches $\mathcal{T} \in [\Lambda]$, $|\mathcal{T}| = t$. The previous observation must hold for all users. Thus, the subset of caches $\mathcal{T}$, which in the dedicated-cache setting refers to a unique subset of users, now refers to many different user subsets. Specifically, since there are L_i users in cache i, the number of possible user combinations for $\mathcal{T}$ is $\prod_{i\in\mathcal{T}} L_i$. Thus, we need a different subfile for every combination, since each user is going to request a different file. To allocate memory to a specific cache λ, we need to consider all the possible subsets $\mathcal{T}$ containing the said cache λ. This superset is $\{\tau \in C_t^{[\Lambda]} : \lambda \in \tau\}$, and therefore we can write the total number of subfiles stored in cache λ as

$$\sum_{\tau\in C_t^{[\Lambda]}:\lambda\in\tau} \prod_{i\in\tau} L_i = L_\lambda \sum_{\tau\in C_{t-1}^{[\Lambda]\setminus\lambda}} \prod_{i\in\tau} L_i = L_\lambda \cdot e_{t-1}(\boldsymbol{L}\setminus\{L_\lambda\})$$

which corresponds to the numerator of (33), which sets the memory allocation. The denominator of (33) is just obtained as the sum of all subfiles, so we satisfy the constraint in (34).

This memory allocation and placement yields an interesting property — described in the following proposition — that will be instrumental in the design and performance of the delivery phase.

Proposition 1. *For any $(t+1)$-tuple $\mathcal{Q} \subset [\Lambda]$, and for any specific cache $\lambda \in \mathcal{Q}$, the total number of subfiles with first subscript $\tau = \mathcal{Q}\setminus\{\lambda\}$ that are missing from all the users associated to cache λ is the same for any $\lambda \in \mathcal{Q}$ and it equals*

$$P_{\mathcal{Q}} \triangleq \prod_{j=1}^{t+1} L_{\mathcal{Q}(j)}. \tag{35}$$

Proof. For any $(t+1)$-tuple $\mathcal{Q} \subset [\Lambda]$, consider cache $\lambda \in \mathcal{Q}$ and let $\tau = \mathcal{Q}\setminus\{\lambda\}$. There are L_λ requested files from the users $\mathcal{U}_\lambda$ of cache λ, each having $\prod_{j=1}^{t} L_{\tau(j)}$ subfiles with first index τ. This means that the total number of subfiles that need to be sent to serve users in $\mathcal{U}_\lambda$ is $L_\lambda \prod_{j=1}^{t} L_{\tau(j)} = \prod_{j=1}^{t+1} L_{\mathcal{Q}(j)}$, which does not depend on which $\lambda \in \mathcal{Q}$ is selected. □

B. Delivery phase

For ease of presentation, we will use $\mathbf{d}_\lambda$ to denote the vector of indices of the files requested by the users in $\mathcal{U}_\lambda$. For a fixed $(t+1)$-tuple $\mathcal{Q}$ and any $\lambda \in \mathcal{Q}$, consider the set of subfiles

$$\{W_{\tau,m}^{(\mathbf{d}_\lambda(j))} : j \in [L_\lambda], m \in A_\tau\}$$

with first subscript $\tau = \mathcal{Q}\setminus\{\lambda\}$, where these subfiles are desired by the users in $\mathcal{U}_\lambda$. Recalling from Proposition 1 that the cardinality of this set is $P_{\mathcal{Q}}$ (cf. (35)), we relabel the subfiles of the set as

$$\{F_{\tau,j}^{(\lambda)} : j \in [P_{\mathcal{Q}}]\}.$$

Because of the design of the cache placement phase in Section V-A, we note that, for any $(t+1)$-tuple $\mathcal{Q}$ and any $j \in [P_{\mathcal{Q}}]$, the set of subfiles

$$F_{\mathcal{Q}\setminus\{\lambda\},j}^{(\lambda)},\ \forall \lambda \in \mathcal{Q}, \tag{36}$$

forms a clique of $t+1$ nodes. By Proposition 1, for any $(t+1)$-tuple $\mathcal{Q} \in [\Lambda]$, we have $P_{\mathcal{Q}}$ cliques as in (36), all containing $t+1$ nodes. Consequently, we transmit, for each $(t+1)$-tuple $\mathcal{Q} \subseteq [\Lambda]$, the following $P_{\mathcal{Q}}$ XORs:

$$X_{\mathcal{Q}}(j) = \bigoplus_{\lambda\in\mathcal{Q}} F_{\mathcal{Q}\setminus\{\lambda\},j}^{(\lambda)},\quad \forall j \in [P_{\mathcal{Q}}], \tag{37}$$

whose structure allows for clique-based decoding as in [2].

C. Performance of the scheme

The fact that there are $P_{\mathcal{Q}}$ XORs for each $(t+1)$-tuple Q implies a total of

$$\sum_{\mathcal{Q}\in C_{t+1}^{[\Lambda]}} P_{\mathcal{Q}} = \sum_{\mathcal{Q}\in C_{t+1}^{[\Lambda]}} \prod_{j=1}^{t+1} L_{\mathcal{Q}(j)} = e_{t+1}(\boldsymbol{L})$$

transmissions, and a corresponding delivery time of

$$T(t,\boldsymbol{L}) = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}, \tag{38}$$

where the denominator $e_t(\boldsymbol{L})$ is due to (32). In the proof of Theorem 3 (cf. (26)), we have seen that the above achievable delivery time in (38) can be written in the more standard form

$$T(t,\boldsymbol{L}) = \frac{\sum_{\lambda=1}^{\Lambda} L_\lambda(1-\gamma_\lambda)}{t+1}, \tag{39}$$

where $\{\gamma_\lambda\}_{\lambda=1}^{\Lambda}$ is the memory allocation obtained in (33) and that leads to (25) in Lemma 1.

VI. Information Theoretic Converse

In this section, we present a converse bound on the optimal delivery time $T^*(t,\boldsymbol{L})$, which will serve as a proof for Theorem 1. We will also prove Theorem 2 by restricting the cache placement scheme to be regular as in Definition 1, which implies that t is integer.

This converse result builds on a different approach with respect to previous bounds. We remark that, in the scenario here considered, we are deriving the optimal cache placement *and* the optimal memory allocation, i.e., the size of each cache memory. This diverges from previous results, where the optimal placement was derived for a homogeneous memory allocation [29]. Indeed, as we will prove in the following, the results are against the intuition from [29] that more homogeneous profiles $\boldsymbol{L}$ would lead to better performances, as it turns out to be the opposite.

In what follows, we denote the set of demand vectors having distinct file requests by $\mathcal{D}_{wc}$, such that $\mathcal{D}_{wc} \triangleq \{\mathbf{d} \in [N]^K : d_j \neq d_i,\ \forall\ i \neq j\}$. Finally, we denote the part of file $W^{(i)}$ exclusively stored in the caches in set τ as $W_\tau^{(i)}$.

A. Lower bounding $T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L})$

We first present a *generic* lower bound on the delivery time $T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L})$ as a function of the cache permutation $\sigma \in \mathcal{S}_\Lambda$, where $\mathcal{S}_n$ denotes the symmetric group of all permutations of $[n]$. This result was first stated in [29] (cf. equation (51)). However, in [29], this partial result was not stated as a lemma

and/or proposition, and we reproduce it here for the sake of completeness.

Lemma 4. *Consider the delivery phase of a shared-cache network with a cache placement described by $\boldsymbol{\mathcal{Z}}$, demand vector $\mathbf{d}$ and cache occupancy vector $\boldsymbol{L}$. Then, under the assumption of uncoded cache placement, the optimal delivery time $T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L})$ can be lower bounded by the quantity*

$$T_{lb,\sigma}(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L}) \triangleq \sum_{\lambda=1}^{\Lambda} \sum_{\ell=1}^{L_{\sigma(\lambda)}} \sum_{\tau_\lambda \subseteq [\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}} |W_{\tau_\lambda}^{(\mathbf{d}_{\sigma(\lambda)}(\ell))}| \quad (40)$$

where σ denotes an arbitrary permutation of the cache set $[\Lambda]$.

Proof. The proof[5] builds on index coding arguments and is an adaptation of Corollary 1 in [64] to the considered caching problem, in a similar manner as how it has been done in [60]. As described in [29], for any cache placement $\boldsymbol{\mathcal{Z}}$, any demand vector $\mathbf{d}$, and any cache occupancy vector $\boldsymbol{L}$, the caching problem considered here can be converted into an index coding problem and its associated *side information graph*. Then, Lemma 1 in [29] can be used to obtain a lower bound on $T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L})$ by identifying any acyclic subgraph of the side-information graph. Next, in the Appendix Section VII-B of [29] it is proved that, for any cache permutation $\sigma \in \mathcal{S}_\Lambda$, an acyclic subgraph can be identified and used in conjunction with Lemma 1 in [29] to prove our above lemma. □

Next, an *adaptable* lower bound on $T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L})$ can be constructed as a weighted average of the $\Lambda!$ possible lower bounds that stem from (40). Thus, it holds that

$$T^*(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L}) \geq \sum_{\sigma \in \mathcal{S}_\Lambda} w_\sigma T_{lb,\sigma}(\boldsymbol{\mathcal{Z}}, \mathbf{d}, \boldsymbol{L}), \quad (41)$$

where the weights $\{w_\sigma\}$ satisfy $\sum_{\sigma\in\mathcal{S}_\Lambda} w_\sigma = 1$. Any of such possible sets of weights $\{w_\sigma\}$ provides a valid lower-bound for our problem.

Remark 1. *Generally, the approach to construct lower bounds on the delivery time for coded caching problems in works that follow the index coding approach originally proposed in [60] (and the similar genie-aided approach used in [5]) is based on generating symmetry, e.g., by averaging over all possible permutations, or based on a certain cache permutation σ (see for example [5], [65], [27], [29]). This method has been shown to work well in settings that are uniform in terms of number of users per cache and sizes of the caches. However, whenever the system model is affected by some heterogeneity, this approach can easily fail to meet the goal. For the scenario here considered, we have shown in [1] that the uniform average (i.e. $w_\sigma = \frac{1}{\Lambda!}$) leads to a loose bound, which proved our achievable performance to be optimal within a gap that scales linearly with the normalized total cache size t.*

A key contribution of this work is to show that the limitations of index coding bounds in heterogeneous settings are not fundamental and can be overcome by an asymmetric combination of lower bounds, where the combination depends on the topology of the setting. For that, the use of the weighted average and a careful choice of the weights in (41) is crucial to the construction of a tight bound. This approach is utterly different from previous solutions inasmuch as before the goal was generally to avoid *asymmetry, as it was thought that otherwise the exponential complexity induced by the combinatorial nature of the problem would make unfeasible to find a solution. Conversely, we take the opposite direction and we* enforce *asymmetry, but a structured asymmetry that allows us to map the heterogeneity of the setting to the bound. We believe that this approach can be helpful to derive lower bounds for other generic heterogeneous coded caching problems.*

In our derivation, the weights w_σ depend on a parameter p. In particular, for any $p \in [\Lambda]_0$, the choice of the weights w_σ is taken as

$$w_\sigma^{(p)} \triangleq \frac{\prod_{j=1}^{p} L_{\sigma(\Lambda-p+j)}}{\sum_{\sigma'\in\mathcal{S}_\Lambda} \prod_{j=1}^{p} L_{\sigma'(\Lambda-p+j)}}, \quad (42)$$

where we have used the upper index $^{(p)}$ to highlight the dependency of the value of the weights on the choice of the parameter p, and we recall that $\sigma(n)$ denotes the n-th element of any ordered set σ. For $p = 0$ we define $w_\sigma^{(0)} \triangleq \frac{1}{|\mathcal{S}_\Lambda|}$.

Intuition on the value of the weights in (42): We provide some interpretation to the choice of the weights in (42). First, let us consider the more intuitive case were $p = t$. The denominator acts just as normalization to ensure that the sum of all weights is equal to one. Then, in the numerator, we have that the weight is proportional to the product of the last t elements of the permutation σ. Thus, if these last values are big (i.e., the number of users in those caches is high), the weight will be greater. Conversely, the smaller the last t values of the permutation, the smaller the weight.

This seems in contradiction with the result in [29], where the only required permutation was the one satisfying that $L_1 \geq L_2 \geq \ldots \geq L_\Lambda$, which in (42) would take the minimum weight. This difference comes from the key fact that in [29] the cache size was fixed, while here we can optimize it. Indeed, the permutation in [29] was chosen because it provides the longest acyclic graph, thus becoming the tighter bound. However, we now have the freedom to allocate the memory to reduce the worst delivery time. Actually, if we allocate more memory to the cache with more users, which acts as limiting cache in the equal-cache-size case, the delivery time for that cache will be reduced (since there is less content to transmit). The most crucial aspect is to understand how to balance the different bounds, since each of them could the tightest for a specific memory allocation. What the weights in (42) accomplish is that the weighted bounds for all permutations are *equally tight*, which is proven by the fact that we obtain the optimal delivery time. The reason for which we give more weight to the a priori looser bounds is because with memory allocation we are able to tighten them while they provide a shorter acyclic graph. The exact value of the numerator is selected due to the fact that it is the number of different combination of t users that can be simultaneously served for the last t caches of the permutation.

[5]The proof of this lemma is fully presented in [29]. Therefore, we omit in this work the detailed derivation and restrict ourselves to provide the main aspects of the proof; we refer to Section V of [29] for a detailed proof.

B. Lower bound on T^*

We now proceed to derive the lower bound on the optimal delivery time T^*. In this respect, we start by bounding from below the worst-case delay for a fixed cache placement $\boldsymbol{\mathcal{Z}}$, where the bound is obtained as the average rate over all demands with distinct requests as

$$T^*(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) \triangleq \max_{\mathbf{d}} \; T^*(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L}) \geq \frac{1}{|\mathcal{D}_{wc}|}\sum_{\mathbf{d}\in\mathcal{D}_{wc}} T^*(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L}).$$

Combining the above inequality and (41) yields

$$T^*(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) \geq \frac{1}{|\mathcal{D}_{wc}|}\sum_{\mathbf{d}\in\mathcal{D}_{wc}}\sum_{\sigma\in\mathcal{S}_\Lambda} w_\sigma^{(p)} T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L}) \tag{43}$$

$$\overset{(a)}{\geq} \underbrace{\sum_{\mathbf{d}\in\mathcal{D}_{wc}}\sum_{\sigma\in\mathcal{S}_\Lambda}\frac{w_\sigma^{(p)}}{|\mathcal{D}_{wc}|}\sum_{\lambda=1}^{\Lambda}\sum_{\ell=1}^{L_{\sigma(\lambda)}}\sum_{\tau_\lambda\subseteq[\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}}|W_{\tau_\lambda}^{(\mathbf{d}_{\sigma(\lambda)}(\ell))}|}_{\triangleq T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})} \tag{44}$$

for any $p\in[\Lambda]_0$, where in (a) we have used (40).

Next, we rewrite the right-hand side of (44), which we denote by $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$, in the more compact form

$$T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) = \sum_{n=1}^{N}\sum_{\tau\in2^{[\Lambda]}} c_{\tau,n}^{(p)}\frac{|W_\tau^{(n)}|}{N}, \tag{45}$$

where the value of $c_{\tau,n}^{(p)}$ is expressed in the following lemma. Before presenting the lemma, let us introduce the notation $\dot{\boldsymbol{L}}_q \triangleq \prod_{j=1}^{|q|} L_{q(j)}$ for any subset $q\subseteq[\Lambda]$ for the sake of readability.

Lemma 5. *The value of $c_{\tau,n}^{(p)}$ does not depend on the file index n and it takes the form*

$$c_\tau^{(p)} = \frac{1}{\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}\Bigg(\sum_{q\in C_{p+1}^{[\Lambda]}}\dot{\boldsymbol{L}}_q\frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|} + \frac{1}{|\tau|+1}\dot{\boldsymbol{L}}_\tau\sum_{s\in C_{p-j}^{[\Lambda]\setminus\{\tau\}}}\Big(\dot{\boldsymbol{L}}_s\sum_{i=1}^{p-j}L_{s(i)}\Big)\Bigg). \tag{46}$$

Proof. The proof is presented in Appendix IV. □

We can tighten the bound on $T^*(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$ by selecting the most restricting p, such that

$$T^*(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) \geq \max_{p\in[\Lambda]_0} T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L}). \tag{47}$$

From the definition of the optimal delay $T^*(t,\boldsymbol{L})$ in (3), and from (45) and (47), we get

$$T^*(t,\boldsymbol{L}) = \min_{\boldsymbol{\mathcal{Z}}} T^*(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) \tag{48}$$

$$\geq \min_{\boldsymbol{\mathcal{Z}}}\max_{p\in[\Lambda]_0}\sum_{n=1}^{N}\sum_{\tau\in2^{[\Lambda]}} c_\tau^{(p)}\frac{|W_\tau^n|}{N}$$

$$= \min_{\boldsymbol{\mathcal{Z}}}\max_{p\in[\Lambda]_0}\sum_{\tau\in2^{[\Lambda]}} c_\tau^{(p)} a_\tau, \tag{49}$$

where we have introduced the notation $a_\tau \triangleq \frac{1}{N}\left(|W_\tau^{(1)}|+|W_\tau^{(2)}|+\ldots+|W_\tau^{(N)}|\right)$. Now, by considering the library size and the sum cache size constraints, a lower bound on the optimal delay $T^*(t,\boldsymbol{L})$ can be obtained from the solution of the following linear program

$$\begin{aligned}\min_{a_\tau}\max_{p\in[\Lambda]_0}\;&\sum_{\tau\in2^{[\Lambda]}} c_\tau^{(p)} a_\tau\\ \text{subject to}\;&\sum_{\tau\in2^{[\Lambda]}} a_\tau = 1,\\ &\sum_{\tau\in2^{[\Lambda]}}|\tau| a_\tau = t,\\ &a_\tau\geq 0,\quad \forall\tau\in2^{[\Lambda]}.\end{aligned} \tag{50}$$

Let us now focus on the proof of the general lower bound in Theorem 1, and later we will consider the proof for the case with regular placement of Theorem 2, to conclude with an optimality result for the case where we are not restricted to regular placement.

C. Proof of Theorem 1

In what follows, we further lower-bound the constructed lower bound in (50). First of all, let us introduce some useful notation. We first define $\tilde{c}_\tau^{(p)}$ as

$$\tilde{c}_\tau^{(p)} \triangleq \frac{\sum_{q\in C_{p+1}^{[\Lambda]}}\dot{\boldsymbol{L}}_q\frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|}}{\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}.$$

Then, we define the subset of cardinality j that minimizes $\tilde{c}_\tau^{(p)}$ as $\tau_j^\star$, i.e., $\tau_j^\star \triangleq \operatorname{argmin}_{\tau\in2^{[\Lambda]}:|\tau|=j}\tilde{c}_\tau^{(p)}$, and we define $\bar{a}_j$ as $\bar{a}_j \triangleq \sum_{\tau\in2^{[\Lambda]}:|\tau|=j} a_\tau$. Then, it holds that

$$\sum_{\tau\in2^{[\Lambda]}} c_\tau^{(p)} a_\tau \overset{(a)}{\geq} \sum_{\tau\in2^{[\Lambda]}}\tilde{c}_\tau^{(p)} a_\tau \overset{(b)}{\geq} \sum_{j=0}^{\Lambda}\tilde{c}_{\tau_j^\star}^{(p)}\bar{a}_j, \tag{51}$$

where in (a) we have applied the fact that $\tilde{c}_\tau^{(p)} \leq c_\tau^{(p)}$ (cf. (46)), and in (b) we have used the definitions of $\tau_j^\star$ and $\bar{a}_j$.

We now provide a lemma that provides the value of the optimal $\tau_j^\star$, for any $p\in[\Lambda]_0$ and $j\in[\Lambda]_0$.

Lemma 6. *Let us consider that the caches are sorted such that $L_1\geq L_2\geq\cdots\geq L_\Lambda$. Then, for any cardinality $|\tau|=j$, $j\in[\Lambda]$, it holds that $\tau_j^\star \triangleq \operatorname{argmin}_{\tau\in2^{[\Lambda]}:|\tau|=j}\tilde{c}_\tau^{(p)}$ satisfies*

$$\tau_j^\star \triangleq \begin{cases}\{\emptyset\} & \text{if } j=0\\ \{\Lambda-j+1,\;\Lambda-j+2,\;\ldots,\;\Lambda\} & \text{if } 1\leq j<p\\ \{1,\;2,\;\ldots,\;j\}=[j] & \text{if } j\geq p.\end{cases}$$

Note that for $j=p$, $\tilde{c}_\tau^{(p)}$ is the same for every τ such that $|\tau|=j$. This means that $\tilde{c}_{\tau_j^\star}^{(p)} = \tilde{c}_\tau^{(p)}\;\forall\tau:|\tau|=j$.

Proof. The proof is relegated to Appendix V. □

Now, jointly employing (51) in (49) and using the max-min inequality yields

$$\min_{\boldsymbol{\mathcal{Z}}}\max_{p\in[\Lambda]_0}\sum_{\tau\in2^{[\Lambda]}} c_\tau^{(p)} a_\tau \geq \max_{p\in[\Lambda]_0}\min_{\boldsymbol{\mathcal{Z}}}\sum_{j=0}^{\Lambda}\tilde{c}_{\tau_j^\star}^{(p)}\bar{a}_j \tag{52}$$

which implies that

$$T^*(t,\boldsymbol{L}) \geq \max_{p\in\{0,1,\ldots,\Lambda\}} \min_{\bar{a}_j} \sum_{j=0}^{\Lambda} \tilde{c}^{(p)}_{\tau^\star_j} \bar{a}_j \quad \text{subject to} \quad \sum_{j=0}^{\Lambda} \bar{a}_j = 1, \quad \sum_{j=0}^{\Lambda} j\bar{a}_j = t, \quad \bar{a}_j \geq 0, \quad \forall j \in [\Lambda]_0. \tag{53}$$

We present now a result that will be instrumental in establishing the following step in the derivation.

Proposition 2. *The sequence* $\left\{\tilde{c}^{(p)}_{\tau^\star_j}\right\}$ *is a decreasing sequence in* $j \in [\Lambda]_0$.

Proof. The proof is relegated to Appendix VI. □

We now focus on the inner optimization problem in (53) for any fixed $p \in [\Lambda]_0$, and we follow the same steps as in [5] to solve this problem analytically. In this respect, we know from Proposition 2 that $\left\{\tilde{c}^{(p)}_{\tau^\star_j}\right\}$ is a decreasing sequence in $j \in [\Lambda]_0$, and thus its convex envelope is a decreasing and convex sequence. Thus, applying Jensen inequality, we obtain

$$T^*(t,\boldsymbol{L}) \geq \max_{p\in[\Lambda]_0} \mathcal{T}^{(p,\boldsymbol{L})}_{low}(t) \tag{54}$$

where

$$\mathcal{T}^{(p,\boldsymbol{L})}_{low}(t) \triangleq \mathrm{Conv}_{j\in[\Lambda]_0}\left(\tilde{c}^{(p)}_{\tau^\star_j}\right). \tag{55}$$

Then, Theorem 1 simply follows from the fact that

$$\max_{p\in\{0,1,\ldots,\Lambda\}} \mathcal{T}^{(p,\boldsymbol{L})}_{low}(t) \geq \mathcal{T}^{(\bar{t},\boldsymbol{L})}_{low}(t), \tag{56}$$

where $\bar{t} = \mathrm{round}(t)$. Hence, we have proved Theorem 1.

D. Proof of Theorem 2

Since under the regular assumption of Definition 1 it holds that t is integer (i.e., $t \in [\Lambda]_0$), we first note that $\bar{t} = t$. Furthermore, this assumption implies that $\bar{a}_t = 1$ and $\bar{a}_j = 0$ for any $j \in \{[\Lambda]_0 \backslash t\}$. We can lower bound (53) by fixing p to $p = t$, which reduces (53) to

$$\min_{\bar{a}_j} \quad \tilde{c}^{(t)}_{\tau^\star_t}\bar{a}_t \quad \text{subject to} \quad \bar{a}_t = 1. \tag{57}$$

It is easy to verify that, for all $\tau \in [\Lambda] : |\tau| = t$, with $p = t$, it holds that $\tilde{c}^{(t)}_{\tau} = \tilde{c}^{(t)}_{\tau^\star_t}$ and that

$$\tilde{c}^{(t)}_{\tau^\star_t} = \frac{\sum_{q\in C^{[\Lambda]}_{t+1}} \prod_{j=1}^{t+1} L_{q(j)}}{\sum_{q\in C^{[\Lambda]}_{t}} \prod_{j=1}^{t} L_{q(j)}} = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} = \mathcal{T}^{(t,\boldsymbol{L})}_{low,\,reg}(t), \tag{58}$$

where the last step follows from the definition of $\mathcal{T}^{(t,\boldsymbol{L})}_{low,\,reg}(t)$ in (23). This, together with (57), directly results in

$$T^*(t,\boldsymbol{L}) \geq \tilde{c}^{(t)}_{\tau^\star_t} = \mathcal{T}^{(t,\boldsymbol{L})}_{low,\,reg}(t), \tag{59}$$

which concludes the proof of Theorem 2. □

E. Optimality beyond regular placement

Apart from the optimality results presented in Theorem 3 for the case with regular placement, we present in the following theorem a new tight bound for the case where no assumption on regular placement is taken, and for which the achievable delivery time in (24) is exactly optimal.

Theorem 5. *For integer values of* $t \in [\Lambda]_0$, *the achievable delivery time* $T(t,\boldsymbol{L})$ *in* (24) *is exactly optimal under the assumption of uncoded cache placement when the sequence* $\{\tilde{c}^{(t)}_{\tau^\star_j}\}_{j\in[\Lambda]_0}$ *is convex in* j, *where*

$$\tilde{c}^{(t)}_{\tau^\star_j} \triangleq \frac{\sum_{q\in C^{[\Lambda]}_{t+1}} \frac{t+1-|q\cap\tau^\star_j|}{j+1-|q\cap\tau^\star_j|} \prod_{i=1}^{t+1} L_{q(i)}}{\sum_{\ell\in C^{[\Lambda]}_{t}} \prod_{i=1}^{t+1} L_{\ell(i)}}. \tag{60}$$

Proof. From (60), we can write $\mathcal{T}^{(t,\boldsymbol{L})}_{low}(x)$ (defined in (21)) as

$$\mathcal{T}^{(t,\boldsymbol{L})}_{low}(x) = \mathrm{Conv}_{j\in[\Lambda]_0}\left(\tilde{c}^{(t)}_{\tau^\star_j}\right). \tag{61}$$

For any convex sequence, its lower convex envelope contains all its elements. Thus, if $\{\tilde{c}^{(t)}_{\tau^\star_j}\}_{j\in[\Lambda]_0}$ is convex, then $\mathcal{T}^{(t,\boldsymbol{L})}_{low}(t) = \tilde{c}^{(t)}_{\tau^\star_t}$ for any integer point $t \in [\Lambda]_0$.

Moreover, as explained in (58) for the proof of Theorem 2, it holds that, for $j = t$, $\tilde{c}^{(t)}_{\tau^\star_t} = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}$, and also that (cf. (26))

$$T^*(t,\boldsymbol{L}) \geq \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} = \frac{\sum_{\lambda=1}^{\Lambda} L_\lambda(1-\gamma_\lambda)}{t+1} = T(t,\boldsymbol{L}), \tag{62}$$

which concludes the proof of the optimality of Theorem 5. □

Next, we present an example of a setting for which our scheme is optimal under the constraint of uncoded cache placement without assuming regular placement.

Example 1. *Consider the cache-aided network of the example in Section III with* $\boldsymbol{L} = (3,2,1)$ *and* $t = 2$. *The sequence* $\{\tilde{c}^{(2)}_{\tau^\star_j}\}_{j\in\{0,1,2,3\}}$ *takes the values* $\{18/11, 12/11, 6/11, 0\}$, *which is a convex sequence. From Theorem 5, this implies that for the considered example in Section III the achievable delivery time* $T^*(2,(3,2,1)) = 6/11$ *is information-theoretically optimal under the assumption of uncoded cache placement.*

VII. The Scenario with Imperfect Topology Knowledge

In this section, we present the achievable scheme and the matching converse for the scenario with imperfect topology knowledge previously described. We remind the reader that our setting entails only partial knowledge about the cache occupancy vector $\boldsymbol{L}$, and that this available knowledge is denoted by $\bar{\boldsymbol{L}}$. This analysis allows us to characterize the impact of the degree of knowledge about the network topology on the performance of coded caching. In this scenario, we recall that the memory allocation and placement phases assume that the future cache occupancy vector during the delivery phase matches the available information $\bar{\boldsymbol{L}}$, although eventually this cache occupancy vector turns out to be $\boldsymbol{L}$ at the delivery phase.

We first present the proposed scheme for this scenario and its performance, followed by the converse analysis to provide a bound. We then prove that the presented scheme attains the

converse result to prove Theorem 4, and finally we provide some numerical examples to better illustrate the impact of the topology knowledge.

A. Achievable scheme

With the knowledge of the cache occupancy vector $\bar{\boldsymbol{L}}$ at hand — which as we recall can represent the expected cache occupancy vector in the delivery phase — the server designs the memory allocation $\{\gamma_\lambda\}_{\lambda=1}^{\Lambda}$ and cache placement $\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}$ as described in Section V-A. For the subsequent delivery phase with topology described by $\boldsymbol{L}$, the following fact holds.

Proposition 3. *For any $(t+1)$-tuple $\mathcal{Q} \subset [\Lambda]$, any scalar $\lambda \in \mathcal{Q}$ and tuple $\tau = \mathcal{Q} \setminus \{\lambda\}$ ($|\tau| = t$), the total number of subfiles of the form $W^{(n)}_{\tau,m_\tau}$ that are missing from all users associated to any specific cache $\lambda \in \mathcal{Q}$ is equal to*

$$P_{\lambda_\mathcal{Q}} = L_\lambda \prod_{j=1}^{t} \bar{L}_{\tau(j)}. \tag{63}$$

Proof. Considering cache λ, there are L_λ users requesting L_λ files. For each of these files there are $|A_\tau| = \prod_{j=1}^{t} \bar{L}_{\tau(j)}$ subfiles with first index $\tau = \mathcal{Q} \setminus \{\lambda\}$. □

In what follows, we will use $\mathbf{d}_\lambda$ to denote the vector of indices of the files requested by the users in $\mathcal{U}_\lambda$. Similarly to the delivery scheme in section V-B, for a fixed $(t+1)$-tuple $\mathcal{Q}$ and any $\lambda \in \mathcal{Q}$, let us consider the set of subfiles with first subscript $\tau = \mathcal{Q} \setminus \{\lambda\}$ that are requested from users in $\mathcal{U}_\lambda$, i.e.,

$$\{W^{(\mathbf{d}_\lambda(j))}_{\tau,m} : j \in [L_\lambda], m \in A_\tau\},$$

where we recall that $A_\tau = \{1, 2, \ldots, \prod_{j=1}^{t} \bar{L}_{\tau(j)}\}$. From Proposition 3, we know that the cardinality of this set is $P_{\lambda_\mathcal{Q}}$, and thus we can relabel the set of these subfiles with successive integer indexes as

$$\mathcal{F}_{\lambda_\mathcal{Q}} = \{F^{(\lambda)}_{\tau,j} : j \in [P_{\lambda_\mathcal{Q}}]\}.$$

Let us now define the quantity $P^{(max)}_\mathcal{Q} \triangleq \max_{\lambda \in \mathcal{Q}} P_{\lambda_\mathcal{Q}}$, and let us note that, for each $\lambda \in \mathcal{Q}$, it holds that $F^{(\lambda)}_{\tau,j} \triangleq \emptyset$ for any index j such that $P_{\lambda_\mathcal{Q}} < j \leq P^{(max)}_\mathcal{Q}$. Because of the design of the cache placement phase, we notice that for any $(t+1)$-tuple $\mathcal{Q}$ and any $j \in [P^{(max)}_\mathcal{Q}]$, the set of subfiles

$$F^{(\lambda)}_{\mathcal{Q}\setminus\{\lambda\},j}, \quad \forall \lambda \in \mathcal{Q} \tag{64}$$

forms a clique of $t+1$ nodes. For any $(t+1)$-tuple $\mathcal{Q} \in [\Lambda]$, we have $P^{(max)}_\mathcal{Q}$ cliques as in (64), all corresponding to $t+1$ nodes. Consequently, we transmit the following $P^{(max)}_\mathcal{Q}$ XORs for each $(t+1)$-tuple $\mathcal{Q} \subseteq [\Lambda]$:

$$X_\mathcal{Q}(j) = \bigoplus_{\lambda \in \mathcal{Q}} F^{(\lambda)}_{\mathcal{Q}\setminus\{\lambda\},j}, \quad \forall j \in [P^{(max)}_\mathcal{Q}], \tag{65}$$

whose structure allows for clique-based decoding as in [2].

Delay Evaluation: For each $(t+1)$-tuple $\mathcal{Q}$, the server transmits

$$P^{(max)}_\mathcal{Q} = \max_{\lambda \in \mathcal{Q}} P_{\lambda_\mathcal{Q}}$$

different XORs. The total number of XORs sent through the channel is

$$\sum_{\mathcal{Q} \in C^{[\Lambda]}_{t+1}} \max_{\lambda \in \mathcal{Q}} P_{\lambda_\mathcal{Q}} = \sum_{\mathcal{Q} \in C^{[\Lambda]}_{t+1}} \max_{\lambda \in \mathcal{Q}} L_\lambda \prod_{j=1}^{t} \bar{L}_{\tau_\lambda(j)} \tag{66}$$

where $\tau_\lambda \triangleq \mathcal{Q} \setminus \{\lambda\}$. The subpacketization applied at cache placement $\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}$ (cf. (32)) and (66) imply that the normalized delivery time of the achievable scheme for any $t, \boldsymbol{L}$ and $\bar{\boldsymbol{L}}$ is

$$T(t, \boldsymbol{L}, \bar{\boldsymbol{L}}) = \frac{\sum_{\mathcal{Q} \in C^{[\Lambda]}_{t+1}} \max_{\lambda \in \mathcal{Q}} L_\lambda \dot{\bar{\mathbf{L}}}_{\tau_\lambda}}{e_t(\bar{\boldsymbol{L}})}, \tag{67}$$

where we have applied the notation $\dot{\bar{\mathbf{L}}}_{\tau_\lambda} \triangleq \prod_{j=1}^{t} \bar{L}_{\tau_\lambda(j)}$, and we recall that $e_t(\bar{\boldsymbol{L}}) \triangleq \sum_{\tau_\lambda \in C^{[\Lambda]}_t} \prod_{j=1}^{t} \bar{L}_{\tau_\lambda(j)}$.

B. Converse bound

To develop the lower bound for $T^*(t, \boldsymbol{L}, \bar{\boldsymbol{L}})$, we immediately observe that the applied placement is determined and given by $\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}$, i.e., the optimal placement for $\bar{\boldsymbol{L}}$. Before proceeding with the proof, we note that, under the cache placement $\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}$, we can simplify the notation by considering as a single subfile $W^{(n)}_\tau$ the set of subfiles stored exactly in the caches in set τ, for any set $\tau \subset [\Lambda]$ of cardinality $|\tau| = t$, such that $W^{(n)}_\tau = \{W^{(n)}_{\tau,m_\tau} : m_\tau \in A_\tau\}$ for any $n \in [N]$.

Under the cache placement $\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}$, Lemma 4 also holds for the considered scenario with imperfect topology knowledge, such that $T^*(\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}, \mathbf{d}, \boldsymbol{L})$ can be lower bounded as

$$T^*(\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}, \mathbf{d}, \boldsymbol{L}) \geq \sum_{\lambda=1}^{\Lambda} \sum_{\ell=1}^{L_{\sigma(\lambda)}} \sum_{\tau_\lambda \subseteq [\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}} |W^{(\mathbf{d}_{\sigma(\lambda)}(\ell))}_{\tau_\lambda}|, \tag{68}$$

$$= \sum_{\lambda=1}^{\Lambda} L_{\sigma(\lambda)} \sum_{q \in C^{[\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}}_t} \frac{\dot{\bar{\mathbf{L}}}_q}{e_t(\bar{\boldsymbol{L}})}, \tag{69}$$

where (69) follows directly from the fact that, under the cache placement $\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}$, we have that

$$|W^{(n)}_\tau| = \begin{cases} \frac{\dot{\bar{\mathbf{L}}}_\tau}{e_t(\bar{\boldsymbol{L}})} & \text{if } |\tau| = t \\ 0 & \text{otherwise} \end{cases} \quad \forall n \in [N].$$

Now, we first note that (69) does not depend on the specific demand $\mathbf{d}$. From this fact, we proceed to maximize over all possible user caches permutations σ to obtain

$$T^*(t, \boldsymbol{L}, \bar{\boldsymbol{L}}) = \max_d T^*(\boldsymbol{\mathcal{Z}}^*_{\bar{\boldsymbol{L}}}, \mathbf{d}, \boldsymbol{L}) \tag{70}$$

$$\geq \max_{\sigma \in S_\Lambda} \sum_{\lambda=1}^{\Lambda} L_{\sigma(\lambda)} \sum_{q \in C^{[\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}}_t} \frac{\dot{\bar{\mathbf{L}}}_q}{e_t(\bar{\boldsymbol{L}})} \tag{71}$$

$$= \max_{\sigma \in S_{\Lambda,\Lambda-t}} \sum_{\lambda=1}^{\Lambda-t} L_{\sigma(\lambda)} \sum_{q \in C^{[\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}}_t} \frac{\dot{\bar{\mathbf{L}}}_q}{e_t(\bar{\boldsymbol{L}})} \tag{72}$$

where $S_{\Lambda,\Lambda-t}$ is the set of $(\Lambda - t)$-permutations of $[\Lambda]$.

C. Proof of Theorem 4

In order to prove Theorem 4, we have to prove that the achievable delivery time in (67) matches the lower bound in (72). To do so, we first notice that (67) and (72) have the same denominator, thus leaving us to prove that the numerator of the achievable delivery time is exactly equal to the numerator of the bound, i.e., to prove that

$$\sum_{\tau\in C_{t+1}^{[\Lambda]}} \max_{\lambda\in\tau} L_\lambda \dot{\bar{\boldsymbol{L}}}_{\tau\setminus\lambda} = \max_{\sigma\in S_{\Lambda,\Lambda-t}} \sum_{\lambda=1}^{\Lambda-t} L_{\sigma(\lambda)} \sum_{\tau\in C_t^{[\Lambda]\setminus\{\sigma(1),\dots,\sigma(\lambda)\}}} \dot{\bar{\boldsymbol{L}}}_\tau. \quad (73)$$

Proving (73) is challenging, and it exemplifies one of the main challenges arising when dealing with asymmetric settings, that we have to operate with asymmetric combinatorial expressions. To prove (73), we start by constructing the set

$$\Phi \triangleq \left\{ \max_{i\in\tau} L_i \dot{\bar{\boldsymbol{L}}}_{\tau\setminus\{i\}} : \tau\subset[\Lambda],\ |\tau| = t+1 \right\}, \quad (74)$$

which is comprised of the addends of the left-hand-side of (73). We naturally have that $|\Phi| = \binom{\Lambda}{t+1}$.

Let us introduce the term *cache leader* in a specific way. In the following, we say that cache j is a *leader* of a set of caches $\tau\subset[\Lambda]$ if $j = \text{argmax}_{i\in\tau} L_i \dot{\bar{\boldsymbol{L}}}_{\tau\setminus\{i\}}$. In other words, cache j is a *leader* of a set $\tau\subset[\Lambda]$ if it is the cache in τ that maximizes the geometric mean of the tuple $\{\bar{L}_i\}_{i\in\tau}$ when one of these $\bar{L}_i$ is substituted by the corresponding L_i from the same cache. The next lemma shows that the caches that act as leaders follow a particular structure.

Lemma 7. *For any $j\in[\Lambda]$, let us denote the number of times that j is a leader in Φ as $\mathcal{N}_j$. Then, $\mathcal{N}_j$ satisfies that*

$$\mathcal{N}_j \in \left\{ 0, \binom{t}{t}, \binom{t+1}{t}, \dots, \binom{\Lambda-1}{t} \right\}. \quad (75)$$

Proof. Without loss of generality, let us assume that j is such that $L_j\bar{L}_i \geq L_i\bar{L}_j\ \forall i\in\phi$ for some $\phi\subset[\Lambda]\setminus\{j\}, |\phi| = m$, and for some $m\in\{t, t+1,\dots,\Lambda-1\}$. Notice that we must have $m\geq t$, since, for $m<t$, j cannot be a leader in Φ. The fact that there exist $\binom{m}{t}$ $t-$combinations of ϕ implies that j is a leader in Φ at least $\binom{m}{t}$ times, since $L_j\bar{L}_\tau \geq L_i\bar{L}_{\{j\}\cup\tau\setminus\{i\}}, \forall i\in\tau, \forall\tau\in C_t^\phi$ by assumption. This consideration and the fact that j might not be a leader in Φ complete the proof. □

Let us now consider the set of all the leaders $\ell_1,\ell_2,\ell_3,\dots,\ell_\Lambda$ in Φ, and let us sort them such that, without loss of generality, we assume that $L_{\ell_j}\bar{L}_{\ell_{j+1}} \geq L_{\ell_{j+1}}\bar{L}_{\ell_j}$ for any $j\in[\Lambda-1]$. It can be easily verified that this order of the leaders implies that

$$L_{\ell_j}\bar{L}_{\ell_i} \geq L_{\ell_i}\bar{L}_{\ell_j} \qquad \forall i>j. \quad (76)$$

Let us consider $j=1$. From the above, we have that $L_{\ell_1}\bar{L}_{\ell_i} \geq L_{\ell_i}\bar{L}_{\ell_1}$ for any $i>1$. We can multiply both sides of the inequality by $\dot{\bar{\boldsymbol{L}}}_\eta$ for any $\eta\subseteq[\Lambda]\setminus\{\ell_1,\ell_i\}$ of cardinality $|\eta| = t-1$, such that we can write that

$$L_{\ell_1}\bar{L}_{\ell_i}\dot{\bar{\boldsymbol{L}}}_\eta \geq L_{\ell_i}\bar{L}_{\ell_1}\dot{\bar{\boldsymbol{L}}}_\eta \qquad \forall\eta\subseteq[\Lambda]\setminus\{\ell_1,\ell_i\}, \quad \forall i>1. \quad (77)$$

There are $\binom{\Lambda-2}{t-1}$ such subsets for each i, thus a total of $(\Lambda-1)\binom{\Lambda-2}{t-1}$ different inequalities. For ℓ_1 to be the leader of a set τ, $|\tau| = t+1$, we need $L_{\ell_1}\bar{L}_{\ell_i}\dot{\bar{\boldsymbol{L}}}_{\tau\setminus\{\ell_1,\ell_i\}} \geq L_{\ell_i}\bar{L}_{\ell_1}\dot{\bar{\boldsymbol{L}}}_{\tau\setminus\{\ell_1,\ell_i\}}$ for any $i:\ell_i\in\tau\setminus\ell_1$. That is, we need t different inequalities among those in (77), each one from a different i. Then, the set of $(\Lambda-1)\binom{\Lambda-2}{t-1}$ different inequalities in (77) imply that ℓ_1 is a leader of $\frac{\Lambda-1}{t}\binom{\Lambda-2}{t-1} = \binom{\Lambda-1}{t}$ different sets $\tau\subseteq[\Lambda]$, $|\tau| = t+1$. Indeed, that amounts to all the possible sets in which ℓ_1 appears.

After having considered ℓ_1 for the sake of comprehension, let us now consider a general ℓ_j. Let us now multiply (76) by $\dot{\bar{\boldsymbol{L}}}_\eta$ for any $\eta\subseteq[\Lambda]\setminus\{\{\ell_k\}_{k\leq j},\ell_i\}$, $|\eta| = t-1$. Note that now we have only considered the subsets that do not include neither ℓ_i nor any ℓ_k for $k\leq j$. Hence, we have $\binom{\Lambda-j-1}{t-1}$ such subsets for each i, thus a total of $(\Lambda-j)\binom{\Lambda-j-1}{t-1}$ different inequalities.

Now, let us denote by Ω_j the set of subsets of cardinality $t+1$ which contain ℓ_j but do not contain any ℓ_k such that $k<j$, i.e., $\Omega_j \triangleq \{\tau : |\tau| = t+1,\ \tau\ni j, \{\tau\setminus\ell_j\}\subseteq[\Lambda]\setminus\{\ell_k\}_{k\leq j}\}$. Note that within the previous set of inequalities, i.e., within

$$L_{\ell_j}\bar{L}_{\ell_i}\dot{\bar{\boldsymbol{L}}}_\eta \geq L_{\ell_i}\bar{L}_{\ell_j}\dot{\bar{\boldsymbol{L}}}_\eta \qquad \forall\eta\subseteq[\Lambda]\setminus\{\{\ell_k\}_{k\leq j},i\},\ \forall i>j, \quad (78)$$

we can find the t inequalities required for ℓ_j to be the leader of any subset τ in Ω_j (since, for an arbitrary $\tau\in\Omega_j$, we need $L_{\ell_j}\bar{L}_{\ell_i}\dot{\bar{\boldsymbol{L}}}_{\tau\setminus\{\ell_j,\ell_i\}} \geq L_{\ell_i}\bar{L}_{\ell_j}\dot{\bar{\boldsymbol{L}}}_{\tau\setminus\{\ell_j,\ell_i\}}$ for any $i:\ell_i\in\tau\setminus\ell_j$). Consequently, the set of $(\Lambda-j)\binom{\Lambda-j-1}{t-1}$ different inequalities in (78) imply that ℓ_j is a leader of $\frac{\Lambda-j}{t}\binom{\Lambda-j-1}{t-1} = \binom{\Lambda-j}{t}$ different sets $\tau\subseteq[\Lambda]$, $|\tau| = t+1$. Indeed, that amounts to all the possible sets in which ℓ_j appears and no ℓ_k, $k<j$, appears.

Interestingly, this implies that each ℓ_j is the leader of all the sets τ in which it appears *and* none of the previous $\{\ell_k\}_{k<j}$ appears. Summing up all the sets for which $\ell_1,\dots,\ell_{\Lambda-t}$ are leaders yields

$$\sum_{n=1}^{\Lambda-t}\binom{\Lambda-n}{t} = \binom{\Lambda}{t+1}, \quad (79)$$

which matches the cardinality of Φ. Hence, there are only $\Lambda-t$ leaders.[6]

Finally, by considering all possible $(\Lambda-t)$-combinations of $[\Lambda]$ as all the possible set of leaders, we can conclude that

$$\sum_{q\in C_{t+1}^{[\Lambda]}} \max_{\lambda\in q} L_\lambda \dot{\bar{\boldsymbol{L}}}_{q\setminus\lambda} = \max_{\sigma\in S_{\Lambda,\Lambda-t}} \sum_{\lambda=1}^{\Lambda-t} L_{\sigma(\lambda)} \sum_{q\in C_t^{[\Lambda]\setminus\{\sigma(1),\dots,\sigma(\lambda)\}}} \dot{\bar{\boldsymbol{L}}}_q, \quad (80)$$

which concludes the proof of Theorem 4. □

D. Performance versus degree of topology knowledge

In order to provide some insights about the previously derived expressions, we illustrate the derived results for the case where the occupancy vector at delivery phase is a disturbed

[6]Note that, for $\ell_{\Lambda-t+1}$, the number of possible subsets of cardinality $t+1$ in $[\Lambda]$ which do not contain any element in $\{\ell_k\}_{k\in[\Lambda-t+1]}$ is zero because the set $[\Lambda]\setminus\{\ell_k\}_{k\in[\Lambda-t]}$ has cardinality t.

version of the occupancy vector known at memory allocation and placement phase. We present a comparison with state-of-the-art schemes to show the extent of the derived results.

We take the natural assumption that the information about the occupancy vector during the phase of memory allocation and placement is a long-term statistic of the realized occupancy vectors at delivery phase. Specifically, we consider in the delivery phase that the actual cache occupancy vector $\boldsymbol{L}$ is a realization of a collection of independent random variables $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_\Lambda)$ with expected value $\bar{\boldsymbol{L}} = (\bar{L}_1, \bar{L}_2, \ldots, \bar{L}_\Lambda)$, i.e., where the cache occupancy vector *assumed* in the placement phase is such that $\bar{L}_\lambda = \mathbb{E}[\mathcal{L}_\lambda]$, $\lambda \in [\Lambda]$, and hence $K = \sum_{\lambda=1}^{\Lambda} \mathbb{E}[\mathcal{L}_\lambda]$.

Before presenting the numerical results, let us first present the following intuitive proposition, which shows that the expected minimum delivery time in the scenario with imperfect knowledge of the topology $\bar{\boldsymbol{L}}$ is lower-bounded by the minimum delivery time of the setting where the topology is perfectly known and it matches $\bar{\boldsymbol{L}}$.

Proposition 4. *For the ($\boldsymbol{\mathcal{L}}$,t) scenario with imperfect topology knowledge, the expected delivery time over $\boldsymbol{\mathcal{L}}$ satisfies*

$$\mathbb{E}_{\boldsymbol{\mathcal{L}}}[T^*(t, \boldsymbol{L}, \bar{\boldsymbol{L}})] \geq T(t, \bar{\boldsymbol{L}}). \tag{81}$$

Proof. We have that

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\mathcal{L}}}[T^*(t, \boldsymbol{L}, \bar{\boldsymbol{L}})] \\
&= \mathbb{E}_{\boldsymbol{\mathcal{L}}}\left[\max_{\sigma \in S_{\Lambda,\Lambda-t}} \sum_{\lambda=1}^{\Lambda-t} L_{\sigma(\lambda)} \sum_{q \in C_t^{[\Lambda]} \setminus \{\sigma(1),\ldots,\sigma(\lambda)\}} \frac{\dot{\boldsymbol{L}}_q}{e_t(\bar{\boldsymbol{L}})}\right] \\
&\geq \max_{\sigma \in S_{\Lambda,\Lambda-t}} \mathbb{E}_{\boldsymbol{\mathcal{L}}}\left[\sum_{\lambda=1}^{\Lambda-t} L_{\sigma(\lambda)} \sum_{q \in C_t^{[\Lambda]} \setminus \{\sigma(1),\ldots,\sigma(\lambda)\}} \frac{\dot{\boldsymbol{L}}_q}{e_t(\bar{\boldsymbol{L}})}\right] \\
&= \max_{\sigma \in S_{\Lambda,\Lambda-t}} \sum_{\lambda=1}^{\Lambda-t} \mathbb{E}_{\boldsymbol{\mathcal{L}}}\left[L_{\sigma(\lambda)}\right] \sum_{q \in C_t^{[\Lambda]} \setminus \{\sigma(1),\ldots,\sigma(\lambda)\}} \frac{\dot{\boldsymbol{L}}_q}{e_t(\bar{\boldsymbol{L}})} \qquad (82) \\
&= \frac{\sum_{\mathcal{Q} \in C_{t+1}^{[\Lambda]}} \max_{\lambda \in \mathcal{Q}} \mathbb{E}_{\boldsymbol{\mathcal{L}}}\left[L_\lambda\right] \dot{\bar{\mathbf{L}}}_{\mathcal{Q}\setminus\{\lambda\}}}{e_t(\bar{\boldsymbol{L}})} \qquad (83) \\
&= \frac{\sum_{\mathcal{Q} \in C_{t+1}^{[\Lambda]}} \max_{\lambda \in \mathcal{Q}} \bar{L}_\lambda \dot{\bar{\mathbf{L}}}_{\mathcal{Q}\setminus\{\lambda\}}}{e_t(\bar{\boldsymbol{L}})} = \frac{e_{t+1}(\bar{\boldsymbol{L}})}{e_t(\bar{\boldsymbol{L}})} = T(t, \bar{\boldsymbol{L}}),
\end{aligned}$$

where (82) follows from the independence of the random variables $\{\mathcal{L}_\lambda\}$ and where (83) has been proven in Section VII-C (cf. (80)). □

Next, we consider a scenario with $N = 70$ files, $\Lambda = 6$ caches and an average number of users of $K = 60$. We consider all the possible occupancy vectors in this setting, i.e., $\{\bar{\boldsymbol{L}} \mid \bar{L}_i \geq \bar{L}_j$ for all $i, j \in [\Lambda]$ such that $i < j$, $\sum_{i=1}^{6} \bar{L}_i = 60\}$. From such set, which contains 12692 occupancy vectors, we randomly select 500. For each one of the selected cases, we evaluate the performance of the considered schemes for three different random scenarios, specifically:

(a) A case where each random variable $\mathcal{L}_\lambda$ follows a Poisson distribution with mean $\bar{L}_\lambda$, i.e. $\mathcal{L}_\lambda \sim \text{Poiss}(\bar{L}_\lambda)$;

(b) A case where each random variable $\mathcal{L}_\lambda$ follows a Gaussian distribution with mean $\bar{L}_\lambda$ and standard deviation equal to $0.3\bar{L}_\lambda$;

(c) A case where, before the delivery phase, 30% of the users that were associated to each cache change their associated cache. Each of these users freely chooses its new associated cache among the $\Lambda = 6$ caches, and the probability of selecting each cache is proportional to the initial amount of users in such cache. Note that in this case the total number of users remains $K = 60$, unlike the two previous cases.

For each of the three cases, we evaluate 20 random realizations of $\boldsymbol{L}$ for each one of the 500 $\bar{\boldsymbol{L}}$.

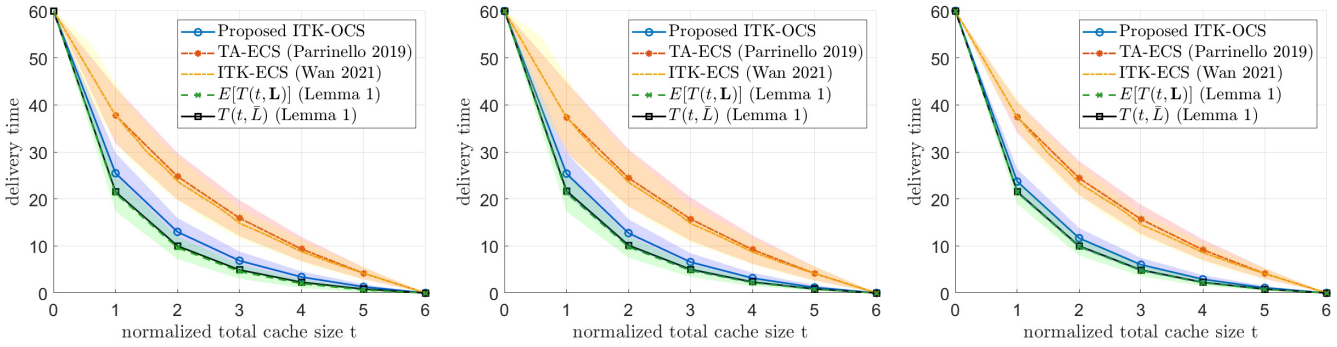
For these experiments, we evaluate the average delivery time $\mathbb{E}_{\boldsymbol{\mathcal{L}}}[T(t, \boldsymbol{L}, \bar{\boldsymbol{L}})]$ of three different schemes:

- The first scheme is the here proposed scheme achieving the delivery time in Theorem 4 for the scenario with imperfect topology knowledge, and that we will denote as *ITK-OCS scheme* (for Imperfect-Topology-Knowledge Optimized-Cache-Size scheme);
- the second scheme is the topology-agnostic scheme in [29], which we refer to as *TA-ECS* (for Topology-Agnostic Equal-Cache-Size scheme), and which does not exploit the knowledge of $\bar{\boldsymbol{L}}$ for the cache placement, and which instead uses the MAN cache placement corresponding to a uniform cache-size allocation;
- finally, the third scheme, which we will refer to as *ITK ECS scheme* (for Imperfect-Topology-Knowledge Equal-Cache-Size scheme), is the one achieving the delivery time in equation (24a) of [59] for the case when there are no cache-less users (i.e., when $K_{\text{mbs}} = 0$ following the notation from [59]). We note that [59] assumes that all caches have the same size, which cannot be optimized. The cache placement of the scheme from [59]) partitions the set of caches in G groups such that all the caches in the same group store the same content, and it applies MAN placement for G caches/users. If the cache occupancy vector $\boldsymbol{L}$ is known in advance during placement, the best partition is chosen by leveraging $\boldsymbol{L}$ in order to minimize the delivery time. In the scenario with imperfect topology knowledge, however, $\boldsymbol{L}$ is not precisely known in advance, and thus the partition in the *ITK ECS scheme* is selected with respect to the available cache occupancy vector knowledge $\bar{\boldsymbol{L}}$. Delivery is performed by means of the multi-round scheme in [29], [9].

Furthermore, we also evaluate the following theoretical genie-aided bounds:

- the memory-rate curve $(t, T(t, \bar{\boldsymbol{L}}))$ from Lemma 1, which would be achieved if $\boldsymbol{\mathcal{L}}$ was deterministic, equal to $\bar{\boldsymbol{L}}$, and perfectly known during placement (i.e., in the topology-aware setting), represented by the diamond purple line;
- the average memory-rate curve $(t, \mathbb{E}_{\boldsymbol{\mathcal{L}}}[T(t, \boldsymbol{L})])$, which would be achieved on average for any $\boldsymbol{\mathcal{L}}$, $\mathcal{L}_\lambda \sim \text{Poiss}(\bar{L}_\lambda)$, if, for each realization, $\boldsymbol{L}$ is known at placement. It is represented by the cross green line.

We remark that the proposed scheme is optimal and that these two last genie-aided methods require extra information that is



(a) $\mathcal{L}_\lambda \sim \text{Poiss}(\bar{L}_\lambda)$ (b) $\mathcal{L}_\lambda \sim \text{Normal}(\bar{L}_\lambda,\ 0.3\bar{L}_\lambda)$ (c) Random migration of users

Fig. 2: Average delivery time for the three considered scenarios. Shadowed areas represent the mean $\pm$ std. deviation.

not actually available.

Figure 2 shows the average delivery time and standard deviation for each one of the considered schemes and distributions, as a function of the cache redundancy. It is evident that the proposed scheme with optimized shared caches largely outperforms the other two schemes, thus highlighting the importance of proper memory allocation even with imperfect knowledge of the occupancy vector. The plot also confirms Proposition 4, interestingly showing that the loss of performance due to the randomness in the number of users per cache is not so important thanks to the use of our proposed scheme. Furthermore, it shows that the optimal performance of the topology-aware scenario where the cache occupancy vector $\bar{\boldsymbol{L}}$ is such that $\bar{L}_\lambda = \mathbb{E}[\mathcal{L}_\lambda], \lambda \in [\Lambda]$, (i.e., assuming that the cache occupancy vector available information is the mean value of $\boldsymbol{\mathcal{L}}$), which is denoted by $T(t, \bar{\boldsymbol{L}})$, is a good approximation of the expected performance over $\boldsymbol{\mathcal{L}}$, i.e. $T(t, \bar{\boldsymbol{L}}) \approx \mathbb{E}_{\boldsymbol{\mathcal{L}}}[T(t, \boldsymbol{L})]$.

VIII. Discussion and Conclusions

This work explores the shared-cache coded caching problem under the well known bottleneck of having an asymmetric user-to-cache association. Such asymmetry was previously shown to inflict substantial performance degradation in coded caching systems. This work reveals, that — under any cumulative cache-size constraint — a carefully optimized cache-size allocation, together with a novel cache-placement and delivery scheme, not only entirely alleviate this bottleneck but in fact turn this bottleneck into an advantage compared with the symmetric case. The new optimized allocation and scheme jointly allow for the maximal coding gain $t+1$ as well as for a boosted local caching gain, and thus for a reduced overall delivery time. Together with the novel cache size allocation and cache content placement, a main contribution of this work is the novel information theoretic converse that proves the information theoretic optimality of the achieved performance under simple and practical assumptions. Crucial to the tightness of the new converse is a novel careful combination of several MAIS bounds (cf. [64]) which, deviating from classical approaches that consist of averaging them uniformly, is one of the first converse results that captures the heterogeneity of the system and hits the exact optimal delivery time. The bounds in the said entangled combination are weighted accordingly to a function of the number of users of the caches that impact in the MAIS bound, which is different for each of them.

In a setting where asymmetry generally resulted in very substantial DoF losses, the new approach manages to exploit knowledge of the number of users connected to each cache to substantially increase the performance compared to topology-agnostic scenarios. Our work has offered tools that can help in the construction of other converses in the presence of asymmetry and has shed more light on how placement can be changed to work together with arbitrary cache sizes.

One of the crucial outcomes of our work is that the aforementioned asymmetry bottleneck can be substantially alleviated even if we are only partially aware of the user-to-cache association. A new scheme here proposed manages to substantially alleviate the bottleneck even in the presence of significant uncertainty of the user-to-cache association. This is one if the first studies exploring the connection between coded caching and the degree of knowledge of the network topology. We have also shown that, for any given the memory allocation and cache placement, this scheme is optimal.

All these results show the decisive importance of memory allocation in coded caching, since not considering it leads to the collapse of the coded caching multiplicative gains. Furthermore, this allocation is in fact very impactful even in the presence of noisy knowledge of the topology.

This work has focused on deriving the optimal coded caching gains that can be achieved in the considered heterogeneous setting. The impact of multi-antenna transmission and the case where users can be associated simultaneously to several caches are extensions that are worth investigating. An interesting direction for future works would be to study the subpacketization-constrained version of the problem, which might limit the actual gains in some practical applications. Also, while this paper assumes that the cost of fetching data from the caches is negligible, an interesting extension of this work could explore the performance of such networks where the access to a cache implies a certain performance cost.

APPENDIX I: PROOF OF PROPERTY 2

Let us define $\dot{\boldsymbol{x}}_\theta = \prod_{j=1}^{k} x_{\theta(j)}$. By Property 1, we have

$$\sum_{x_i \in \{\mathcal{X}\setminus\phi\}} x_i \cdot e_{k-1}(\mathcal{X}\setminus\{x_i\}) = |\mathcal{X}\setminus\phi| \cdot e_k(\mathcal{X}) - \sum_{x_i\in\{\mathcal{X}\setminus\phi\}} e_k(\mathcal{X}\setminus\{x_i\}). \quad (84)$$

We can then write that $e_k(\mathcal{X}\setminus\{x_i\}) = \sum_{q\subseteq\mathcal{X},\ |q|=k,\ x_i\notin q} \dot{\boldsymbol{x}}_q$. Hence, a particular set $q \subseteq \mathcal{X}$, $|q| = k$, will appear in $e_k(\mathcal{X}\setminus\{x_i\})$ if and only if $x_i \notin q$, which implies that a particular q will appear in $\sum_{x_i\in\{\mathcal{X}\setminus\phi\}} e_k(\mathcal{X}\setminus\{x_i\})$ only for the addends x_i belonging to $\{\mathcal{X}\setminus\{\phi,q\}\}$. Consequently, it follows that

$$\sum_{x_i\in\{\mathcal{X}\setminus\phi\}} e_k(\mathcal{X}\setminus\{x_i\}) = \sum_{\substack{q\subseteq\mathcal{X}\\|q|=k}} \dot{\boldsymbol{x}}_q \cdot |\mathcal{X}\setminus\{\phi,q\}|. \quad (85)$$

Applying (85) into (84) yields

$$\sum_{x_i \in \{\mathcal{X}\setminus\phi\}} x_i \cdot e_{k-1}(\mathcal{X}\setminus\{x_i\}) = \sum_{q\in C_k^{\mathcal{X}}} \prod_{j=1}^{k} x_{q(j)} \cdot (|\mathcal{X}\setminus\phi| - |\mathcal{X}\setminus\{\phi,q\}|) \quad (86)$$

$$\overset{(a)}{=} \sum_{q\in C_k^{\mathcal{X}}} \prod_{j=1}^{k} x_{q(j)} \cdot |q\setminus\phi|, \quad (87)$$

where in (a) we have applied that

$$|\mathcal{X}\setminus\phi| - |\mathcal{X}\setminus\{\phi,q\}| = |\mathcal{X}| - |\phi| - (|\mathcal{X}| - |\phi\cup q|) = |q\setminus\phi|,$$

which concludes the proof of Property 2. □

APPENDIX II: PROOFS FOR THE ACHIEVABILITY RESULTS

A. Extension of Lemma 1 to non-integer values of t

When the normalized total memory t has a non-integer value, we apply memory sharing along the same lines as in [2]. We write t as $t = \alpha\lfloor t\rfloor + (1-\alpha)\lceil t\rceil$, for $\alpha \in [0,1]$, and we split each file $W^{(n)}$ of the library in two parts $W^{(n),1}, W^{(n),2}$, where $|W^{(n),1}| = \alpha$ and $|W^{(n),2}| = (1-\alpha)$, such that the library remains partitioned in two sub-libraries

$$\mathcal{W}_1 = \{W^{(n),1} | n \in [N]\}, \quad \mathcal{W}_2 = \{W^{(n),2} | n \in [N]\}.$$

Afterwards, we first employ the cache placement scheme in Section V-A for sub-library $\mathcal{W}_1$ with a total sum-cache constraint $\lfloor t\rfloor$, and then we do the same for sub-library $\mathcal{W}_2$ with a total sum-cache constraint $\lceil t\rceil$. The delivery phase now consists of 2 rounds, each as in Section V-B: the first round employs XORs of order $\lfloor t\rfloor + 1$ to serve files $\{W^{(d_k),1} | k \in [K]\}$, whereas the second round employs XORs of order $\lceil t\rceil + 1$ to serve $\{W^{(d_k),2} | k \in [K]\}$. This scheme clearly results in the following delivery time

$$T(t, \boldsymbol{L}) = \alpha T(\lfloor t\rfloor, \boldsymbol{L}) + (1-\alpha) T(\lceil t\rceil, \boldsymbol{L}). \quad (88)$$

We now present a lemma that is instrumental for the proof of Lemma 1 for non-integer values of t.

Lemma 8. *The sequence* $\left\{\frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}\right\}_{t\in[\Lambda]_0}$ *is a decreasing and convex sequence.*

Proof. The proof is relegated to Appendix II-C. □

The achievability of (88) implies that, for integer t, the straight line between points $(t, T(t,\boldsymbol{L}))$ and $(t+1, T(t+1,\boldsymbol{L}))$ is also achievable. Moreover, we know from (38) that $\{T(t,\boldsymbol{L})\}_{t\in[\Lambda]_0} = \left\{\frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}\right\}_{t\in[\Lambda]_0}$, and thus Lemma 8 implies that $\{T(t,\boldsymbol{L})\}_{t\in[\Lambda]_0}$ is a convex sequence. Since the lower convex envelope of a convex sequence is a piece-wise function composed of the segments connecting two successive elements of the sequence $\{T(t,\boldsymbol{L})\}_{t\in[\Lambda]_0}$ (i.e., (88)), Lemma 1 is proven.

B. Proof of equation (33)

Let us now prove (33), i.e., that $\gamma_\lambda = \frac{L_\lambda\ e_{t-1}(\boldsymbol{L}\setminus\{L_\lambda\})}{e_t(\boldsymbol{L})}$ for any $\lambda \in [\Lambda]$. To evaluate γ_λ, we first note that all subfiles are equally-sized and that the placement scheme is *symmetric* with respect to the library files, i.e., the caching strategy does not depend on the file index $n \in [N]$. This suggests that γ_λ can be evaluated as the ratio between the number of subfiles (of any file $W^{(n)}$) stored in cache λ and the total number of subfiles $S = e_t(\boldsymbol{L})$ into which $W^{(n)}$ is split. The fact that a subfile $W^{(n)}_{\tau,m}$ is placed in $\mathcal{Z}_\lambda$ if and only if $\lambda \in \tau$, together with the fact that $|A_\tau| = \prod_{j=1}^{t} L_{\tau(j)}$, automatically yield the numerator of (33). □

C. Proof of Lemma 8

In the following, we prove Lemma 8, which states that the sequence $\{c_t^{(t)}\}_{t\in\{0,\dots,\Lambda\}}$ is a decreasing and convex sequence. Since the t-th elementary symmetric polynomial in $\boldsymbol{L}$ is given by $e_t(\boldsymbol{L}) \triangleq \sum_{q\in C_{t+1}^{[\Lambda]}} \prod_{j=1}^{t+1} L_{q(j)}$, and $c_t^{(t)} \triangleq \frac{\sum_{q\in C_{t+1}^{[\Lambda]}} \prod_{j=1}^{t+1} L_{q(j)}}{\sum_{q\in C_t^{[\Lambda]}} \prod_{j=1}^{t} L_{q(j)}}$, we can write that

$$c_t^{(t)} = e_1(\boldsymbol{L}) \qquad \text{if } t = 0$$
$$c_t^{(t)} = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} \qquad \text{if } 1 \le t \le \Lambda - 1$$
$$c_t^{(t)} = 0 \qquad \text{if } t = \Lambda.$$

The proof of Lemma 8 builds on the relation of the coefficients $c_t^{(t)}$ with the elementary symmetric polynomials and the following lemma.

Lemma 9. *Let* $\{a_k\}$, $k \in [n]$, *be a strictly log-concave sequence satisfying that* $a_k > 0$ *for any* $k < n$. *Then, the sequence* $\{b_k \triangleq \frac{a_{k+1}}{a_k}\}$, $k \in [n-1]$, *is a decreasing (strictly) convex sequence.*

Proof. Since $\{a_k \mid k \in [n]\}$ is a strictly log-concave sequence, it holds that $a_k^2 > a_{k+1}a_{k-1}$. Moreover, since $a_k > 0$ for any $k < n$ by definition of the sequence $\{a_k\}$, we have that

$$a_k^2 > a_{k+1}a_{k-1} \quad \Longleftrightarrow \quad \frac{a_k}{a_{k-1}} > \frac{a_{k+1}}{a_k}. \quad (89)$$

The right-hand side of (89) is equivalent to $b_{k-1} > b_k$, which proves that $\{b_j\}$ is a decreasing sequence. Next, we prove that $\{b_j\}$ is also a convex sequence.

A discrete sequence $\{b_j\}$, $j \in [n-1]$, is convex if and only if $2b_j \leq b_{j-1} + b_{j+1}$ for any $j \in \{2, \ldots, n-2\}$. Thus, the sequence $\left\{\frac{a_{k+1}}{a_k}\right\}_{k\in[n-1]}$ is convex if and only if

$$2\frac{a_{j+1}}{a_j} \leq \frac{a_j}{a_{j-1}} + \frac{a_{j+2}}{a_{j+1}}, \qquad \forall j \in \{2, \ldots, n-2\}. \tag{90}$$

Let us now multiply (90) by the denominators $(a_{j-1}a_j a_{j+1})$ to obtain

$$\begin{aligned} 2a_{j-1}a_{j+1}^2 &\leq a_j^2 a_{j+1} + a_{j-1}a_j a_{j+2} \\ &< a_j^2 a_{j+1} + a_{j-1}a_{j+1}^2 \end{aligned}$$

which follows from the strict log-concavity of $\{a_k\}$ (i.e., $a_j a_{j+2} < a_{j+1}^2$). Re-ordering terms, we obtain

$$a_{j-1}a_{j+1} < a_j^2,$$

which is always true because $\{a_k\}$ is strictly log-concave. Consequently, the sequence $\{b_j\}$ is a strictly convex sequence. □

Continuing with the proof of Lemma 8, we note that, given Lemma 9, we only need to show that the sequence $\{1, \{e_t\}_{t\in[\Lambda]}, 0\}$ is strictly log-concave: If it is strictly log-concave, and upon defining $\{a_k\} = \{1, \{e_t\}_{t\in[\Lambda]}, 0\}$ (such that $b_k \triangleq \frac{a_{k+1}}{a_k} = c_t^{(t)}$), applying Lemma 9 yields that $c_t^{(t)}$ is a strictly convex sequence, which will conclude the proof of Lemma 8.

In order to prove that the sequence $\{1, \{e_t\}_{t\in[\Lambda]}, 0\}$ is strictly log-concave, let us first introduce the *elementary symmetric means* E_k, which are defined as

$$E_k(\boldsymbol{L}) \triangleq \frac{e_k(\boldsymbol{L})}{\binom{n}{k}}, \tag{91}$$

where $\binom{n}{k}$ is the number of addends in $e_k(\boldsymbol{L})$ for a set of $|\boldsymbol{L}| = n$ elements (in our case, $n = \Lambda$). Hereinafter, we omit the dependence of e_k and E_k on the occupancy vector $\boldsymbol{L}$ because $\boldsymbol{L}$ is the same set for any e_k and E_k considered in the following.

These elementary symmetric means have a property, which was was proved by Newton [66], that says that, for any n-tuple of non-negative numbers, it holds that the sequence $\{E_k\}_{k\in[n]}$ is a log-concave sequence, and thus

$$E_k^2 \geq E_{k-1}E_{k+1}, \quad k \in \{2, \ldots, n-1\}, \tag{92}$$

where the inequality is strict unless all the elements of the n-tuple coincide.

In order to prove the log-concavity of our sequence $\{1, \{e_t\}_{t\in[\Lambda]}, 0\}$, we first obtain from (91)-(92) that

$$E_t^2 \geq E_{t-1}E_{t+1} \quad \iff \quad \frac{e_t^2}{\binom{n}{t}^2} \geq \frac{e_{t-1}e_{t+1}}{\binom{n}{t-1}\binom{n}{t+1}}, \tag{93}$$

for any $1 < t < n$. Then, we can write that

$$\begin{aligned} e_t^2 &\geq \frac{\binom{n}{t}^2}{\binom{n}{t-1}\binom{n}{t+1}} e_{t-1}e_{t+1} \\ &> e_{t-1}e_{t+1}, \end{aligned} \tag{94}$$

which implies that $\{e_t\}_{1\leq t\leq\Lambda}$ is strictly log-concave, and where the last step follows from the strict log-concavity of the binomial coefficient [67]. It remains to prove that $e_1^2 > 1 \cdot e_2 = e_2$ and that $e_\Lambda^2 > e_{\Lambda-1} \cdot 0 = 0$. The fact that $e_\Lambda^2 > 0$ is always true because $e_\Lambda = \prod_{j=1}^{\Lambda} L_j > 0$ because $L_j \geq 1$. On the other hand, to show that $e_1^2 > e_2$, let us recall the Maclaurin inequalities [68], which state that

$$E_1 \geq \sqrt{E_2} \geq \sqrt[3]{E_3} \geq \cdots \geq \sqrt[n]{E_n}, \tag{95}$$

with equality holding only if all the L_j for any $j \in [n]$ coincide. Let us focus on the first inequality. Since $n = \Lambda$ and $L_j \geq 1$ for any j, it follows that

$$\begin{aligned} E_1 \geq \sqrt{E_2} \quad &\iff \quad \frac{e_1^2}{\binom{\Lambda}{1}^2} \geq \frac{e_2}{\binom{\Lambda}{2}} \\ &\iff \quad \frac{e_1^2}{\Lambda^2} \geq \frac{e_2}{\frac{\Lambda(\Lambda-1)}{2}} \\ &\iff \quad e_1^2 \geq \frac{2\Lambda}{\Lambda-1} e_2. \end{aligned}$$

Since $e_1^2 \geq \frac{2\Lambda}{\Lambda-1}e_2 > e_2$, we obtain that the sequence $\{1, \{e_t\}_{1\leq t\leq\Lambda}, 0\}$ is strictly log-concave. Thus, applying Lemma 9 yields that $c_t^{(t)}$ is a strictly convex sequence, which concludes the proof of Lemma 8.

Appendix III: Proofs of results on the worst case scenario

A. Proof of Corollary 3

From Theorem 4, since the denominator in (27) is equal for any $\boldsymbol{L}$, we have that

$$\operatorname*{argmax}_{\boldsymbol{L}} \; T^*(t, \bar{\boldsymbol{L}}, \boldsymbol{L}) = \operatorname*{argmax}_{\boldsymbol{L}} \sum_{q\in C_{t+1}^{[\Lambda]}} \max_{\lambda\in q} \left\{ L_\lambda \prod_{j=1}^{t} \bar{L}_{\tau_\lambda(j)} \right\}$$

where we recall that τ_λ is defined as $\tau_\lambda \triangleq \{q \setminus \{\lambda\}\}$. Let us define $i_{\max}$ as $i_{\max} \triangleq \operatorname{argmax}_{j\in[\Lambda]} L_j$. It follows that

$$\begin{aligned} \sum_{q\in C_{t+1}^{[\Lambda]}} \max_{\lambda\in q} \left\{ L_\lambda \prod_{j=1}^{t} \bar{L}_{\tau_\lambda(j)} \right\} &= L_{i_{\max}} \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \prod_{j=1}^{t} \bar{L}_{q(j)} \\ &\quad + \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \max_{\lambda\in q} L_\lambda \prod_{j=1}^{t-1} \bar{L}_{\tau_\lambda(j)} \end{aligned} \tag{96}$$

where we have simply split the sum isolating the terms containing $L_{i_{\max}}$.

Let us now consider a different occupancy vector $\boldsymbol{L}'$, which differs from $\boldsymbol{L}$ only on the fact that $L'_{i_{\max}} = L_{i_{\max}} - 1$ and that $L'_k = L_k + 1$, for some $k \in [\Lambda]$, i.e., that one user is

removed from cache $i_{\max}$ and added to cache k. From (96), we can compare the optimal delivery time as

$$\begin{aligned}
&T^*(t,\bar{\boldsymbol{L}},\boldsymbol{L}) - T^*(t,\bar{\boldsymbol{L}},\boldsymbol{L}') \\
&= L_{i_{\max}} \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \prod_{j=1}^{t} \bar{L}_{q(j)} + \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \max_{\lambda\in q} L_\lambda \prod_{j=1}^{t-1} \bar{L}_{\tau_\lambda(j)} \\
&\quad - (L_{i_{\max}}-1) \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \prod_{j=1}^{t} \bar{L}_{q(j)} - \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \max_{\lambda\in q} L'_\lambda \prod_{j=1}^{t-1} \bar{L}_{\tau_\lambda(j)} \\
&\overset{(a)}{=} \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \prod_{j=1}^{t} \bar{L}_{q(j)} + \sum_{q\in \{k\cup C_{t-1}^{[\Lambda]\setminus\{i_{\max},k\}}\}} \max_{\lambda\in q} L_\lambda \prod_{j=1}^{t} \bar{L}_{q(j)} \\
&\qquad - \sum_{q\in \{k\cup C_{t-1}^{[\Lambda]\setminus\{i_{\max},k\}}\}} \max_{\lambda\in q} L'_\lambda \prod_{j=1}^{t} \bar{L}_{q(j)} \\
&\overset{(b)}{\geq} \sum_{q\in C_t^{[\Lambda]\setminus i_{\max}}} \prod_{j=1}^{t} \bar{L}_{q(j)} - \sum_{q\in \{k\cup C_{t-1}^{[\Lambda]\setminus\{i_{\max},k\}}\}} \prod_{j=1}^{t} \bar{L}_{q(j)} \\
&\overset{(c)}{=} \sum_{q\in C_t^{[\Lambda]\setminus\{i_{\max},k\}}} \prod_{j=1}^{t} \bar{L}_{q(j)} > 0
\end{aligned}$$

where (a) follows from the fact that, for the terms relating to the sets without $i_{\max}$, only the cases where cache k is included will be different for $\boldsymbol{L}$ and $\boldsymbol{L}'$; (b) comes from the fact that $L'_\lambda \leq L_\lambda + 1$ for all $\lambda \in \{[\Lambda]\setminus i_{\max}\}$, and (c) because $C_t^{[\Lambda]\setminus\{i_{\max}\}} = \big\{C_t^{[\Lambda]\setminus\{i_{\max},k\}} \bigcup \{k \cup C_{t-1}^{[\Lambda]\setminus\{i_{\max},k\}}\}\big\}$.

Therefore, we have that $T^*(t,\bar{\boldsymbol{L}},\boldsymbol{L}) > T^*(t,\bar{\boldsymbol{L}},\boldsymbol{L}')$. Since we can obtain any occupancy vector from the case $\{1,\ldots,1,K-\lambda+1\}$ by concatenating transformations as the one presented above, it follows that the worst occupancy vector is obtained for $\boldsymbol{L} = \{1,\ldots,1,K-\lambda+1\}$, which concludes the proof of Corollary 3.

B. Proof of Lemma 2

Next, we prove Lemma 2. We recall that we assume that $\Lambda \mid K$. First, due to the max–min inequality, we have that

$$\begin{aligned}
T^*_{wc}(t) &\triangleq \min_{\boldsymbol{\mathcal{Z}}} \max_{\boldsymbol{L}} \max_{\mathbf{d}} T^*(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L}) \\
&\geq \max_{\boldsymbol{L}} \min_{\boldsymbol{\mathcal{Z}}} \max_{\mathbf{d}} T^*(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L}) \\
&= \max_{\boldsymbol{L}} \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}.
\end{aligned}$$

Consequently, to obtain $T^*_{wc}(t)$, we need to solve the following optimization problem.

$$\max_{\boldsymbol{L}=\{L_i\}_{i\in[\Lambda]}} \quad \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} \tag{97}$$

$$s.t. \ \sum_{i=1}^{\Lambda} L_i = K \tag{98}$$

We proceed by applying the method of Lagrangian multipliers. The Lagrangian function is given by

$$\mathcal{L}(\boldsymbol{L},\lambda) = \frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} + \lambda\Big(\sum_{i=1}^{\Lambda} L_i - K\Big). \tag{99}$$

Thus, the partial derivatives of the Lagrangian function are

$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{L},\lambda)}{\partial L_i} &= \frac{\partial}{\partial L_i}\frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})} + \frac{\partial}{\partial L_i}\Big(\lambda\big(\sum_{i=1}^{\Lambda} L_i - K\big)\Big) \qquad (100)\\
&\overset{(a)}{=} \frac{e_t(\boldsymbol{L}\setminus\{L_i\})e_t(\boldsymbol{L}) - e_{t+1}(\boldsymbol{L})e_{t-1}(\boldsymbol{L}\setminus\{L_i\})}{(e_t(\boldsymbol{L}))^2} + \lambda \\
&\overset{(b)}{=} \frac{\Big(e_t\big(\boldsymbol{L}\setminus\{L_i\}\big)\Big)^2 - e_{t-1}\big(\boldsymbol{L}\setminus\{L_i\}\big)e_{t+1}\big(\boldsymbol{L}\setminus\{L_i\}\big)}{(e_t(\boldsymbol{L}))^2} + \lambda, \qquad (101)
\end{aligned}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{L},\lambda)}{\partial \lambda} = \sum_{i=1}^{\Lambda} L_i - K \tag{102}$$

where (a) is obtained from the fact that $\frac{\partial e_j(\boldsymbol{L})}{\partial L_i} = e_{j-1}(\boldsymbol{L}\setminus\{L_i\})$ and (b) from applying Property 1 to $e_t(\boldsymbol{L})$ and $e_{t+1}(\boldsymbol{L})$ and simplifying terms. (101) holds for any $i \in [\Lambda]$.

Next, the following system of equations needs to be satisfied for any maximum/minimum point of $\frac{e_{t+1}(\boldsymbol{L})}{e_t(\boldsymbol{L})}$:

$$\frac{\Big(e_t\big(\boldsymbol{L}\setminus\{L_i\}\big)\Big)^2 - e_{t-1}\big(\boldsymbol{L}\setminus\{L_i\}\big)e_{t+1}\big(\boldsymbol{L}\setminus\{L_i\}\big)}{(e_t(\boldsymbol{L}))^2} + \lambda = 0, \tag{103}$$

$$\sum_{i=1}^{\Lambda} L_i - K = 0 \tag{104}$$

From (103), and for any $i \in [\Lambda]$, it follows that

$$\begin{aligned}
&\Big(e_t\big(\boldsymbol{L}\setminus\{L_i\}\big)\Big)^2 - e_{t-1}\big(\boldsymbol{L}\setminus\{L_i\}\big)e_{t+1}\big(\boldsymbol{L}\setminus\{L_i\}\big) \\
&\qquad = -\lambda(e_t(\boldsymbol{L}))^2, \qquad \forall i \in [\Lambda]. \qquad (105)
\end{aligned}$$

Let us define the function $f_t(\ell) \triangleq e_t^2(\boldsymbol{L}\setminus\{\ell\}) - e_{t+1}(\boldsymbol{L}\setminus\{\ell\})e_{t-1}(\boldsymbol{L}\setminus\{\ell\})$. Thus, from (105), it follows that, for any $i,j \in [\Lambda]$,

$$f_t(L_i) = f_t(L_j). \tag{106}$$

If we compute the first derivative of $f_t(\ell)$ with respect to ℓ, we obtain that

$$\begin{aligned}
\frac{\partial f_t(\ell)}{\partial \ell} &= 2e_t(\boldsymbol{L})e_{t-1}(\boldsymbol{L}\setminus\ell) - e_t(\boldsymbol{L}\setminus\ell)e_{t-1}(\boldsymbol{L}) \\
&\qquad - e_{t+1}(\boldsymbol{L})e_{t-2}(\boldsymbol{L}\setminus\ell) \\
&\overset{(a)}{=} \Big(e_{t-1}^2(\boldsymbol{L}\setminus\ell) - e_t(\boldsymbol{L}\setminus\ell)e_{t-2}(\boldsymbol{L}\setminus\ell)\Big)\ell \\
&\qquad + \Big(e_t(\boldsymbol{L}\setminus\ell)e_{t-1}(\boldsymbol{L}\setminus\ell) - e_{t+1}(\boldsymbol{L}\setminus\ell)e_{t-2}(\boldsymbol{L}\setminus\ell)\Big)
\end{aligned}$$

where (a) follows from applying Property 1 and re-ordering terms. Since, as demonstrated in (94), we have that $\big(e_t(\boldsymbol{L}\setminus\ell)\big)^2 > e_{t-1}\big(\boldsymbol{L}\setminus\ell\big)e_{t+1}\big(\boldsymbol{L}\setminus\ell\big)$ for any $\ell \in [\Lambda]$, and from that we have that $\frac{e_{t-1}}{e_{t-2}} > \frac{e_t}{e_{t-1}} > \frac{e_{t+1}}{e_t}$, we obtain that

$$\frac{\partial f_t(\ell)}{\partial \ell} > 0 \tag{107}$$

This implies that (106) can only be true if $L_i = L_j$. Since (106) applies for all $i,j \in [\Lambda]$, we obtain that the only argument that maximizes (97)-(98) is $L_i = \frac{K}{\Lambda}$ for all $i \in [\Lambda]$.

APPENDIX IV: PROOF OF LEMMA 5

We derive in the following the value of $c_{\tau,n}^{(p)}$ in the expression $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) = \sum_{n=1}^{N}\sum_{\tau\in 2^{[\Lambda]}} c_{\tau,n}^{(p)} \frac{|W_\tau^{(n)}|}{N}$ (cf. (45)), where $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$ has been defined in (44).

From (44) and (42) we can write $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$ as

$$T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L}) = \frac{\overbrace{\sum_{\mathbf{d}\in\mathcal{D}_{wc}}\sum_{\sigma\in\mathcal{S}_\Lambda}\mathcal{Q}_\sigma^{(p)} T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d},\mathbf{L})}^{\triangleq\mathcal{N}_p}}{\underbrace{|\mathcal{D}_{wc}|\sum_{\sigma\in\mathcal{S}_\Lambda}\prod_{j=1}^{p} L_{\sigma(\Lambda-p+j)}}_{\triangleq\mathcal{D}_p}} \tag{108}$$

where $\mathcal{N}_p$ and $\mathcal{D}_p$ denote respectively the numerator and denominator of (108), and where $\mathcal{Q}_\sigma^{(p)} \triangleq \prod_{j=1}^{p} L_{\sigma(\Lambda-p+j)}$ and

$$T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d},\mathbf{L}) \triangleq \sum_{\lambda=1}^{\Lambda}\sum_{\ell=1}^{L_{\sigma(\lambda)}} \sum_{\tau_\lambda\subseteq[\Lambda]\setminus\{\sigma(1),\ldots,\sigma(\lambda)\}} |W_{\tau_\lambda}^{(\mathbf{d}_{\sigma(\lambda)}(\ell))}|. \tag{109}$$

First, we rewrite $\mathcal{D}_p$ (cf. (108)) in a more suitable form as

$$\begin{aligned}\mathcal{D}_p &= P(N,K)\sum_{\sigma\in\mathcal{S}_\Lambda}\prod_{j=1}^{p} L_{\sigma(\Lambda-p+j)}\\ &= P(N,K)\sum_{v\in\mathcal{S}_{\Lambda-p}}\sum_{q\in\mathcal{S}_p}\prod_{j=1}^{p} L_{q(j)}\\ &= P(N,K)(\Lambda-p)!p!\sum_{q\in C_p^{[\Lambda]}}\prod_{j=1}^{p} L_{q(j)},\end{aligned} \tag{110}$$

which follows from basic mathematical manipulations.

For any $n\in[N]$ and any $\tau\subseteq[\Lambda]$, our goal is now to evaluate the coefficient that multiplies each $|W_\tau^{(n)}|$ in $\mathcal{N}_p$ from (108), where we denote this coefficient as $g_{n,\tau}^{(p)}$. We first state the following useful fact.

Fact 1. For any $n\in[N]$ and any $\tau\subseteq[\Lambda]$, if $|W_\tau^{(n)}|$ appears in $T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d},\boldsymbol{L})$ for some $\mathbf{d}\in\mathcal{D}_{wc}$ and some $\sigma\in S_\Lambda$, then it only appears once for all $\lambda\in[\Lambda],\ \ell\in[L_\lambda]$.

Next, we need to split the proof in two cases. First, we consider the case when $|\tau|\geq p$, and afterwards we focus on the case $|\tau|<p$. In the following, we make use of the notation $\dot{\boldsymbol{L}}_q \triangleq \prod_{j=1}^{|q|} L_{q(j)}$ for any $q\subseteq[\Lambda]$.

A. The $|\tau|\geq p$ case

Let us focus on a demand vector $\mathbf{d}'$ such that subfile $|W_\tau^{(n)}|$ is requested by a certain user $k'\in\mathcal{U}_{\lambda'}$ associated to cache λ'. This simply means that $n=d_{k'}$. Afterward, we will consider all possible $\mathbf{d}'$.

1) Focusing on a given demand vector $\mathbf{d}'$: For a specific demand vector $\mathbf{d}'$, our objective is now to evaluate the coefficient $g_{n,\tau}^{(p)}$ of $|W_\tau^{(d_{k'})}|$ in

$$T_{lb}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\mathbf{L}) \triangleq \sum_{\sigma\in\mathcal{S}_\Lambda}\mathcal{Q}_\sigma^{(p)} T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\mathbf{L}). \tag{111}$$

In this respect, we need to identify those permutations $\sigma\in S_\Lambda$ for which $|W_\tau^{(d_{k'})}|$ appears in $T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\boldsymbol{L})$. From the expression of $T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\boldsymbol{L})$ in (109), the following proposition holds.

Proposition 5. *The permutations σ for which $|W_\tau^{(d_{k'})}|$ appears in $T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\boldsymbol{L})$ are such that λ' appears in the permutation σ before any element of the set τ, where λ' is the cache to which user k' is associated ($k'\in\mathcal{U}_{\lambda'}$). We will refer to such permutations as* valid permutations.

Example: Consider $W_{1,2}^{(3)}$, where $d_k=3$ for $k\in\mathcal{U}_3$ and $\tau=\{1,2\}$. This subfile will not appear in $T_{lb,\sigma}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\boldsymbol{L})$ for permutations $(1,2,3),(2,1,3),(1,3,2)$ and $(2,3,1)$, while it will appear for permutations $(3,1,2)$ and $(3,2,1)$. □

It stems from (111) and Proposition 5 that a valid permutation σ contributes with a weight $\mathcal{Q}_\sigma^{(p)}$ to $g_{n,\tau}^{(p)}$.

With Proposition 5 at hand, we notice that, if λ' appears in any one of the last p positions of a permutation σ, there will certainly be an element of τ (recall that $|\tau|\geq p$) which will precede λ', and thus such permutation σ can not be a *valid permutation*. Then, if $|\tau|\geq p$, any valid permutation does not have λ' in the last p positions. In other words, the set of valid permutations is composed of the permutations whose last p positions are given by a set q belonging to $C_p^{[\Lambda]\setminus\{\lambda'\}}$.

Next, we derive the number of valid permutations for each $q\in C_p^{[\Lambda]\setminus\{\lambda'\}}$.

Proposition 6. *Let $q\in C_p^{[\Lambda]\setminus\{\lambda'\}}$ be a fixed ordered p-tuple. Then, the number of* valid permutations *(as defined in Proposition 5) whose last p positions match q is*

$$\frac{\binom{\Lambda-p}{|\tau\setminus q|+1}}{\binom{\Lambda-p-1}{|\tau\setminus q|}}(\Lambda-p-1)!. \tag{112}$$

Proof. As seen before, Proposition 5 implies that λ' cannot be the last p positions of the permutation σ. Let us then consider the case when λ' is in position r for some $r\in[\Lambda-p]$. Let us also denote the set of elements in the last p positions of σ by q. In this case, we have that in the first $r-1$ positions we cannot place any of the elements in q, we cannot place λ', and we cannot place any element of τ that is not in q. It then follows that, for any q, there are

$$P(\Lambda-p-1-|\tau\setminus q|)(\Lambda-p-r)!$$

ways in which we can fill the first $\Lambda-p$ positions of σ with λ' in position r. Considering all possible r values, we have

$$\sum_{r=1}^{\Lambda-p} P(\Lambda-p-1-|\tau\setminus q|,r-1)(\Lambda-p-r)! \tag{113}$$

different forms in which we can fill the first $\Lambda-p$ positions of σ with λ' appearing in the first $\Lambda-p$ positions. We can

manipulate (113) to obtain that

$$\begin{aligned}
&\sum_{r=1}^{\Lambda-p} P(\Lambda-p-1-|\tau\setminus q|,r-1)(\Lambda-p-r)!\\
&=\sum_{r=1}^{\Lambda-p}\frac{(\Lambda-p-1-|\tau\setminus q|)!(\Lambda-p-r)!|\tau\setminus q|!(\Lambda-p-1)!}{(\Lambda-p-|\tau\setminus q|-r)!|\tau\setminus q|!(\Lambda-p-1)!}\\
&=\sum_{r=1}^{\Lambda-p}\frac{\binom{\Lambda-p-r}{|\tau\setminus q|}}{\binom{\Lambda-p-1}{|\tau\setminus q|}}(\Lambda-p-1)!\\
&=\frac{\binom{\Lambda-p}{|\tau\setminus q|+1}}{\binom{\Lambda-p-1}{|\tau\setminus q|}}(\Lambda-p-1)!, \qquad (114)
\end{aligned}$$

which concludes the proof of Proposition 6. □

For each such set $q \in C_p^{[\Lambda]\setminus\{\lambda'\}}$, there are $p!$ possible orderings of its elements. Consequently, after recalling that each permutation σ has associated a weight $\mathcal{Q}_\sigma^{(p)} \triangleq \prod_{j=1}^{p} L_{\sigma(\Lambda-p+j)}$, we can conclude that each $q \in C_p^{[\Lambda]\setminus\{\lambda'\}}$ has a weight in $T_{lb}(\boldsymbol{\mathcal{Z}},\mathbf{d}',\mathbf{L})$ of

$$p!\sum_{q\in C_p^{[\Lambda]\setminus\{\lambda'\}}}\dot{\boldsymbol{L}}_q. \qquad (115)$$

Combining equation (115) and Proposition 6, the total weight of subfile $W_\tau^{(d_{k'})}$ for demand $\mathbf{d}'$ is

$$p!\sum_{q\in C_p^{[\Lambda]\setminus\{\lambda'\}}}\dot{\boldsymbol{L}}_q\frac{\binom{\Lambda-p}{|\tau\setminus q|+1}}{\binom{\Lambda-p-1}{|\tau\setminus q|}}(\Lambda-p-1)!. \qquad (116)$$

We now proceed to evaluate the total number of demands for which subfile $W_\tau^{(d_{k'})}$ is requested.

2) Joining all possible $\mathbf{d}'$*:* It is easy to see that the total number of demands with $d_{k'}=n$ is $P(N-1,K-1)$. If user k' is associated to any of the caches in set τ (i.e., $\lambda'\in\tau$), then subfile $W_\tau^{(d'_k)}$ will not be requested, since it is already stored in cache λ'. Thus, $W_\tau^{(n)}$ will be requested to the server only if $\lambda'\in[\Lambda]\setminus\{\tau\}$. Considering all possible demand vectors, it follows that the total number of times that subfile $W_\tau^{(n)}$ appears in the demand vector is

$$P(N-1,K-1)\sum_{\lambda\in[\Lambda]\setminus\{\tau\}}L_\lambda. \qquad (117)$$

From equations (116) and (117), we have that the coefficient $g_{n,\tau}^{(p)}$ corresponding to $|W_\tau^{(n)}|$ in $\mathcal{N}_p$ can be written as

$$\begin{aligned}
g_{n,\tau}^{(p)} = P(N-1,K-1)\sum_{\lambda\in[\Lambda]\setminus\{\tau\}}\Big(&L_\lambda p!\\
&\times\sum_{q\in C_p^{[\Lambda]\setminus\{\lambda\}}}\big(\dot{\boldsymbol{L}}_q\frac{\binom{\Lambda-p}{|\tau\setminus q|+1}}{\binom{\Lambda-p-1}{|\tau\setminus q|}}(\Lambda-p-1)!\big)\Big). \qquad (118)
\end{aligned}$$

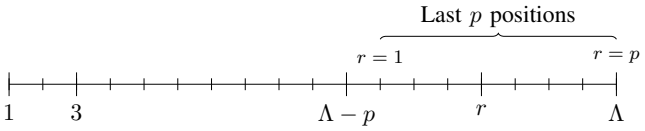


Fig. 3: Illustration of the meaning of index r.

Finally, we obtain the coefficient of any $|W_\tau^{(n)}|$ with $|\tau|\geq p$ in $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$, i.e., $\frac{g_{n,\tau}^{(p)}}{\mathcal{D}_p}$ (cf. (108)), which we rewrite as

$$\begin{aligned}
\frac{g_{n,\tau}^{(p)}}{\mathcal{D}_p} &= \frac{P(N-1,K-1)p!}{P(N,K)(\Lambda-p)!p!\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}\\
&\quad\times\sum_{\lambda\in[\Lambda]\setminus\{\tau\}}L_\lambda\sum_{q\in C_p^{[\Lambda]\setminus\{\lambda\}}}\Big(\dot{\boldsymbol{L}}_q\frac{\binom{\Lambda-p}{|\tau\setminus q|+1}}{\binom{\Lambda-p-1}{|\tau\setminus q|}}(\Lambda-p-1)!\Big)\\
&\overset{(a)}{=}\frac{1}{N\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}\sum_{\lambda\in[\Lambda]\setminus\{\tau\}}L_\lambda\sum_{q\in C_p^{[\Lambda]\setminus\{\lambda\}}}\dot{\boldsymbol{L}}_q\frac{1}{|\tau\setminus q|+1}\\
&\overset{(b)}{=}\frac{\sum_{q\in C_{p+1}^{[\Lambda]}}\dot{\boldsymbol{L}}_q\frac{|q\setminus\tau|}{|\tau\setminus q|+1}}{N\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}\\
&\overset{(c)}{=}\frac{\sum_{q\in C_{p+1}^{[\Lambda]}}\dot{\boldsymbol{L}}_q\frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|}}{N\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}, \qquad (119)
\end{aligned}$$

where (a) follows from basic mathematical manipulations, (b) follows from Property 2 and the fact that for any $\lambda\in[\Lambda]\setminus\{\tau\}$ and $q\in C_p^{[\Lambda]\setminus\{\lambda\}}$ we have $|\tau\setminus q|=|\tau\setminus\{q\cup\{\lambda\}\}|$, and (c) follows from the fact that $|q\setminus\tau|=|q|-|q\cap\tau|$ as well as from $|\tau\setminus q|=|\tau|-|q\cap\tau|$. Defining $c_{n,\tau}^{(p)}\triangleq N\frac{g_{n,\tau}^{(p)}}{\mathcal{D}_p}$ proves the lemma for the case $\tau\geq p$.

B. The $|\tau|<p$ case

We recall that our objective is to obtain the coefficient $c_\tau^{(p)}$ that allows us to write $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$ as $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})=\sum_{n=1}^{N}\sum_{\tau\in2^{[\Lambda]}}c_\tau^{(p)}\frac{|W_\tau^{(n)}|}{N}$, where $T_{lb,p}(\boldsymbol{\mathcal{Z}},\boldsymbol{L})$ has been defined in (44). As for the $|\tau|\geq p$ case, let us focus on a demand vector $\mathbf{d}'$ such that subfile $|W_\tau^{(n)}|$ is requested by a certain user $k'\in\mathcal{U}_\lambda$ associated to cache λ.

The number of permutations for which that λ is not in the last p positions of the permutation is the same as for the case $|\tau|=j\geq p$, and it is given by (119). Let us now denote value in (119) as $\hat{c}_\tau^{(p)}$. However, now, λ can also be found in any position up to the $\Lambda-j$ position of the vector describing the permutation, because $j<p$. In other words, λ can appear in some of the last p positions of the permutation σ. Then, the coefficient $c_\tau^{(p)}$ can be written as

$$c_\tau^{(p)}\triangleq\hat{c}_\tau^{(p)}+\breve{c}_\tau^{(p)} \qquad (120)$$

where $\breve{c}_\tau^{(p)}$ accounts for those permutations in which λ appears in one of the last p positions, which are not considered in $\hat{c}_\tau^{(p)}$. In order to obtain $\breve{c}_\tau^{(p)}$, let us first fix the position of λ, and let r denote in which of the last p positions is λ located. Hence, $r=1$ implies that λ is in the $\Lambda-p+1$ position of the

$$\breve{c}_{\tau}^{(p),num} = \underbrace{P(N-1,K-1)\sum_{\lambda\in[\Lambda]\setminus\{\tau\}} L_\lambda}_{\text{Times that } W_\tau^n \text{ is requested, cf. (117)}} \underbrace{\sum_{s\in C_{p-j-1}^{[\Lambda]\setminus\{\lambda\cup\tau\}}} \overbrace{L_\lambda \dot{\boldsymbol{L}}_\tau \dot{\boldsymbol{L}}_s}^{=\dot{\boldsymbol{L}}_q}}_{\substack{\text{All possible combinations of cache indices in last } p \text{ positions apart from } \lambda,\tau}} \underbrace{\sum_{r=1}^{p-j}}_{\substack{\text{Possible positions of } \lambda}} \underbrace{P(p-r,j)}_{\substack{\text{Ways of placing } \tau\text{'s elements in last } p-r \text{ positions}}} \underbrace{(p-j-1)!}_{\substack{\text{For every } \tau,\lambda, \text{ filling the other } p-j-1 \text{ positions (not occupied by } \tau,\lambda)}} \underbrace{(\Lambda-p)!}_{\substack{\text{Filling the first } \Lambda-p \text{ positions}}}. \tag{122}$$

permutation σ, whereas $r = p$ implies that λ is in the last position (see Fig. 3 for a visual explanation).

If λ can be found in any of the last p positions of the permutation σ, Proposition 5 implies that all the values in τ must also be in those last p positions, and in particular in the positions $\{\Lambda-p+r+1,\dots,\Lambda\}$. Now, the remaining $p-j-1$ positions can be filled with the indices of the other $\Lambda-j-1$ caches. Let us consider a particular set q of indices, $|q| = p$, filling the last p positions. We can write such a set as

$$q \triangleq \{\lambda \cup \tau \cup s\}, \tag{121}$$

where $\lambda \cap \tau = \varnothing$, and $s \in C_{p-j-1}^{[\Lambda]\setminus\{\lambda\cup\tau\}}$. Note that $|[\Lambda]\setminus\{\lambda \cup \tau\}| = \Lambda - j - 1$. Then, the numerator of the coefficient $\breve{c}_\tau^{(p)}$ is given by the term $\breve{c}_\tau^{(p),num}$, whose detailed expression is provided in the top of this page in (122).

Adding the denominator to (122), we can simplify as

$$\begin{aligned}\breve{c}_\tau^{(p)} &= \frac{\breve{c}_\tau^{(p),num}}{P(N,K)p!(\Lambda-p)!\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell} \\ &\overset{(a)}{=} \frac{\sum\limits_{\lambda\in[\Lambda]\setminus\{\tau\}} L_\lambda \sum\limits_{s\in C_{p-j-1}^{[\Lambda]\setminus\{\lambda\cup\tau\}}} L_\lambda \dot{\boldsymbol{L}}_\tau \dot{\boldsymbol{L}}_s \sum\limits_{r=1}^{p-j} P(p-r,j)(p-j-1)!}{Np!\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell} \\ &\overset{(b)}{=} \frac{\sum_{\lambda\in[\Lambda]\setminus\{\tau\}} L_\lambda \sum_{s\in C_{p-j-1}^{[\Lambda]\setminus\{\lambda\cup\tau\}}} L_\lambda \dot{\boldsymbol{L}}_\tau \dot{\boldsymbol{L}}_s \frac{p!}{j+1}}{Np!\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell},\end{aligned} \tag{123}$$

where in (a) we have applied that $\frac{P(N-1,K-1)}{P(N,K)} = \frac{1}{N}$ and canceled out $(\Lambda-p)!$, whereas in (b) we have considered that

$$\begin{aligned}\sum_{r=1}^{p-j} P(p-r,j)(p-j-1)! &= (p-j-1)!\sum_{r=1}^{p-j}\frac{(p-r)!}{(p-r-j)!} \\ &= \frac{p!}{j+1}.\end{aligned} \tag{124}$$

Then, by recalling that $j = |\tau|$, it follows that

$$\begin{aligned}\breve{c}_\tau^{(p)} &= \frac{1}{|\tau|+1}\frac{\sum_{\lambda\in[\Lambda]\setminus\{\tau\}} L_\lambda \sum_{s\in C_{p-j-1}^{[\Lambda]\setminus\{\lambda\cup\tau\}}} L_\lambda \dot{\boldsymbol{L}}_\tau \dot{\boldsymbol{L}}_s}{N\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell} \\ &= \frac{1}{N\sum\limits_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}\cdot\frac{\dot{\boldsymbol{L}}_\tau}{|\tau|+1}\sum_{s\in C_{p-j}^{[\Lambda]\setminus\{\tau\}}}\Big(\dot{\boldsymbol{L}}_s\sum_{i=1}^{p-j}L_{s(i)}\Big).\end{aligned} \tag{125}$$

Thus, from (119) and (125), the total coefficient $c_\tau^{(p)} \triangleq \hat{c}_\tau^{(p)} + \breve{c}_\tau^{(p)}$ is then given by

$$\begin{aligned}c_\tau^{(p)} = \frac{1}{N\sum\limits_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell}\Bigg(&\sum_{q\in C_{p+1}^{[\Lambda]}}\dot{\boldsymbol{L}}_q\frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|} \\ &+\frac{\dot{\boldsymbol{L}}_\tau}{|\tau|+1}\sum_{s\in C_{p-j}^{[\Lambda]\setminus\{\tau\}}}\Big(\dot{\boldsymbol{L}}_s\sum_{i=1}^{p-j}L_{s(i)}\Big)\Bigg),\end{aligned} \tag{126}$$

which concludes the proof of Lemma 5. Note that, for the case where $j \geq p$, it holds that $\breve{c}_\tau^{(p)} = 0$, and hence in this case it holds that $c_\tau^{(p)} = \hat{c}_\tau^{(p)}$, which allows us to consider (126) for any value of $|\tau|$. □

Appendix V: Proof of Lemma 6

In this appendix, we obtain the set τ of cardinality $|\tau| = j$ that minimizes $\tilde{c}_\tau^{(p)}$ for each $j \in [\Lambda]_0$. For the sake of readability, let us recover the notation $\dot{\boldsymbol{L}}_q \triangleq \prod_{j=1}^{|q|} L_{q(j)}$, for any set q. Let us start by recalling that $\tilde{c}_\tau^{(p)}$ is defined as

$$\tilde{c}_\tau^{(p)} \triangleq \frac{\sum_{q\in C_{p+1}^{[\Lambda]}}\dot{\boldsymbol{L}}_q\frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|}}{\sum_{\ell\in C_p^{[\Lambda]}}\dot{\boldsymbol{L}}_\ell.}. \tag{127}$$

For the case of $j = 0$, the only possible set τ is the empty set, whereas for the case $j = p$ we can see from (127) that all $\tilde{c}_\tau^{(p)}$ for any τ with $|\tau| = p$ have the same value, and thus any τ for which $|\tau| = p$ is a solution of the optimization problem. We select $\tau_p^\star = \{1,\ 2,\ \dots,\ p\}$ without loss of generality. In the following, we focus on the cases where $j \in \{[\Lambda] \setminus p\}$.

We start by presenting a key lemma where, instead of considering the set τ that optimizes $\operatorname{argmin}_{\tau\subseteq[\Lambda],\,|\tau|=j}\tilde{c}_\tau^{(p)}$, we consider the problem of finding the *element* ν that minimizes $\tilde{c}_\tau^{(p)}$ for a given subset ϕ of cardinality $|\phi| = j-1$, such that $\tau \triangleq \{\nu \cup \phi\}$.

Lemma 10. *Let us consider a fixed set $\phi \subseteq [\Lambda]$ of cardinality $|\phi| = j-1$. Then, for any $p \in [\Lambda]_0$ and $j \in [\Lambda]$, it holds that*

$$\nu^\star(\phi) \triangleq \operatorname*{argmin}_{\nu\in\{[\Lambda]\setminus\phi\}}\tilde{c}_{\{\nu\cup\phi\}}^{(p)} = \operatorname*{argmin}_{\nu\in\{[\Lambda]\setminus\phi\}}\operatorname{sgn}(p-j)L_\nu \tag{128}$$

Proof. The proof is relegated to Appendix V-A. □

In other words, the cache $\nu^\star$ that minimizes $\tilde{c}_{\{\nu\cup\phi\}}^{(p)}$ for a given ϕ is: (i) the cache not belonging to ϕ with the smallest number of users, when $j < p$; (ii) the cache not belonging to ϕ with the biggest number of users, when $j > p$.

Next, we prove Lemma 6 directly from Lemma 10 and the assumption that the caches are sorted such that $L_1 \geq L_2 \geq \cdots \geq L_\Lambda$. For this, we split the proof in the cases $j < p$ and $j > p$.

1) Case $j < p$: From Lemma 10, we have that

$$\nu^\star(\phi) = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \tilde{c}^{(p)}_{\{\nu\cup\phi\}} = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} L_\nu = \max_{\nu \in \{[\Lambda]\setminus\phi\}} \nu \tag{129}$$

where the last step follows from the ordering $L_1 \geq L_2 \geq \cdots \geq L_\Lambda$. It remains to prove that (129) implies that $\operatorname{argmin}_{\tau\subseteq[\Lambda],\,|\tau|=j} \tilde{c}^{(p)}_\tau = \{\Lambda - j + 1,\ \Lambda - j + 2,\ \ldots,\ \Lambda\}$, which will prove Lemma 6 for $j < p$.

Note that, for any set τ not including Λ, i.e., for any $\tau \in C_j^{[\Lambda-1]}$, it follows from Lemma 10 that, for any subset of τ of cardinality $j-1$, which we denote by $\phi \in C^\tau_{j-1}$, we have that $\tilde{c}^{(p)}_{\{\Lambda\cup\phi\}} \leq \tilde{c}^{(p)}_\tau$.

Similarly, for any set τ including Λ but not including $\Lambda-1$, i.e., for any $\tau \in \{\Lambda \cup C_{j-1}^{[\Lambda-2]}\}$, it follows from Lemma 10 that, for any subset of τ of cardinality $j-1$ including Λ, which we denote by $\phi \in \{\Lambda \cup C_{j-2}^{\tau\setminus\Lambda}\}$, we have that $\tilde{c}^{(p)}_{\{\Lambda-1\cup\phi\}} \leq \tilde{c}^{(p)}_\tau$. We can proceed in the same manner for any possible set τ, taking into account the sets that do not include cache λ but include all caches in $\{\lambda+1,\ldots,\Lambda\}$, which leads to the fact that (129) implies that

$$\underset{\substack{\tau\subseteq[\Lambda]\\|\tau|=j}}{\operatorname{argmin}} \tilde{c}^{(p)}_\tau = \{\Lambda - j + 1,\ \Lambda - j + 2,\ \ldots,\ \Lambda\}. \tag{130}$$

2) Case $p < j$: For this case, the only difference is that now $\operatorname{sgn}(p-j) = -1$. This implies that (129) becomes

$$\nu^\star(\phi) = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \tilde{c}^{(p)}_{\{\nu\cup\phi\}} = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmax}} L_\nu = \min_{\nu \in \{[\Lambda]\setminus\phi\}} \nu \tag{131}$$

Hence, we can follow the same steps as for the case $j < p$ but taking into account the sets that do not include cache λ but include all caches in $[\lambda-1]$, which leads to the fact

$$\underset{\substack{\tau\subseteq[\Lambda]\\|\tau|=j}}{\operatorname{argmin}} \tilde{c}^{(p)}_\tau = \{1,\ 2,\ \ldots,\ j\} \tag{132}$$

which concludes the proof for $p < j$ and thus the proof of Lemma 6. □

A. Proof of Lemma 10

Let us start by noting that, since $\frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|} = 1 + \frac{p-|\tau|}{|\tau|+1-|q\cap\tau|}$, we can re-write (127) as follows:

$$\tilde{c}^{(p)}_\tau = \frac{\sum_{q\in C^{[\Lambda]}_{p+1}} \dot{\boldsymbol{L}}_q}{\sum_{\ell\in C^{[\Lambda]}_p} \dot{\boldsymbol{L}}_\ell} + \frac{p-|\tau|}{\sum_{\ell\in C^{[\Lambda]}_p} \dot{\boldsymbol{L}}_\ell} \Big(\underbrace{\sum_{q\in C^{[\Lambda]}_{p+1}} \dot{\boldsymbol{L}}_q \frac{1}{|\tau|+1-|q\cap\tau|}}_{a^{(p)}_\tau} \Big). \tag{133}$$

From this expression, we can see that, for any τ such that $|\tau| = j$, the only term in (133) that differs with respect to any other τ' of the same cardinality is the sum denoted by $a^{(p)}_\tau$. From this fact, and by taking into account that the sign of $(p-|\tau|)$ is different whether $p > |\tau|$ or not, it follows that

$$\underset{\tau\subseteq[\Lambda],\,|\tau|=j}{\operatorname{argmin}} \tilde{c}^{(p)}_\tau = \underset{\tau\subseteq[\Lambda],\,|\tau|=j}{\operatorname{argmin}} \operatorname{sgn}(p-j) a^{(p)}_\tau. \tag{134}$$

In order to prove Lemma 10, we need to prove that

$$\nu^\star(\phi) \triangleq \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \tilde{c}^{(p)}_{\{\nu\cup\phi\}} = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \operatorname{sgn}(p-j) L_\nu \tag{135}$$

From (133)–(134), it follows that

$$\nu^\star(\phi) \triangleq \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \tilde{c}^{(p)}_{\{\nu\cup\phi\}} = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \operatorname{sgn}(p-j) a^{(p)}_{\{\nu\cup\phi\}}. \tag{136}$$

The following lemma is key to obtain (135) from (136).

Lemma 11. *Let us consider a given set $\phi \subseteq [\Lambda]$ of cardinality $|\phi| = j-1$. Then, for any $p \in [\Lambda]_0$ and $j \in [\Lambda]$, it holds that*

$$\begin{aligned} \nu^\star(\phi) &\triangleq \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \operatorname{sgn}(p-j) a^{(p)}_{\{\nu\cup\phi\}} \\ &= \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \operatorname{sgn}(p-j) b^{(p)}_{\{\nu\cup\phi\}} \end{aligned} \tag{137}$$

where, for any $\tau \triangleq \{\nu\cup\phi\} \in C_j^{[\Lambda]}$, $b^{(p)}_\tau$ is defined as

$$b^{(p)}_\tau \triangleq \sum_{\ell=0}^{\min(p+1,j)} \frac{-1}{(j-\ell)(j-\ell+1)} \sum_{\substack{\mu\subseteq\{\tau\setminus\nu\}\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\setminus\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q. \tag{138}$$

Proof. The proof is relegated to Appendix V-B. □

Next, we demonstrate that (137) in Lemma 11 is equivalent to (135). For that, we split the proof for the cases where $p < j$ and $p > j$.

1) Case $p > j$: Note that the sum $\sum_{\mu\subseteq\{\tau\setminus\nu\},\,|\mu|=\ell} \dot{\boldsymbol{L}}_\mu$ in (138) depends only on the $j-1$ elements on τ that are assumed to be fixed in this step (i.e., on ϕ). Furthermore, since it holds that $\operatorname{sgn}(p-j) = 1$, and all the terms in (138) are negative, we need to maximize $\sum_{q\in C^{[\Lambda]\setminus\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q$ in order to minimize (138). Since the optimization variable (ν) is the term that we *remove* from the sum (recall that $q \subset [\Lambda]\setminus\{\tau\}$), this maximization is achieved by selecting the cache in $\{[\Lambda]\setminus\phi\}$ with the smallest L, i.e., it holds that

$$\nu^\star(\phi) = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} b^{(p)}_{\{\nu\cup\phi\}} = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} L_\nu. \tag{139}$$

2) Case $p < j$: Unlike in the previous case, we now have that $\operatorname{sgn}(p-j) = -1$. With the change of sign, all the addends in (138) are positive, and thus we now want to maximize (138). To do so, we seek to minimize $\sum_{q\in C^{[\Lambda]\setminus\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q$. Since the optimization variable (ν) is the term that we *remove* from the sum, this minimization is achieved by selecting the cache in $\{[\Lambda]\setminus\phi\}$ with the biggest L. Thus, applying the same reasoning as for the case $p > j$, we obtain that

$$\nu^\star(\phi) = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmax}} b^{(p)}_{\{\nu\cup\phi\}} = \underset{\nu \in \{[\Lambda]\setminus\phi\}}{\operatorname{argmax}} L_\nu. \tag{140}$$

Finally, from (139) and (140) we obtain (135), which concludes the proof of Lemma 10. □

B. Proof of Lemma 11

We obtain Lemma 11 by re-writing the terms inside $a_\tau^{(p)}$ such that some of the terms do not impact the optimization problem, and hence we can remove them.

1) Obtaining a new expression for $a_\tau^{(p)}$: It follows that, for every $\tau \in [\Lambda]$, the term $a_\tau^{(p)}$ that we have defined in (133) can be written as

$$a_\tau^{(p)} = \sum_{q \in C_{p+1}^{[\Lambda]}} \dot{\boldsymbol{L}}_q \frac{1}{j+1-|q\cap\tau|} = \sum_{m=0}^{\min(j,p+1)} \frac{1}{j+1-m} \sum_{\substack{q\in C_{p+1}^{[\Lambda]} \\ |q\cap\tau|=m}} \dot{\boldsymbol{L}}_q, \tag{141}$$

where the $\min(j, p+1)$ term in the summation comes from the fact that $|q \cap \tau| \leq \min(|q|, |\tau|)$. For a given τ, let η be a subset of τ of cardinality m, such that $\eta \subseteq \tau$, $|\eta| = m$. The last sum in (141) can be expanded as

$$\sum_{\substack{q\in C_{p+1}^{[\Lambda]} \\ |q\cap\tau|=m}} \dot{\boldsymbol{L}}_q = \sum_{\substack{\eta\subseteq\tau \\ |\eta|=m}} \sum_{\substack{q\in C_{p+1}^{[\Lambda]} \\ q\cap\tau=\eta}} \dot{\boldsymbol{L}}_q. \tag{142}$$

Let us consider a particular $\eta \subseteq \tau$, $|\eta| = m$. Then,

$$\sum_{\substack{q\in C_{p+1}^{[\Lambda]} \\ q\cap\tau=\eta}} \dot{\boldsymbol{L}}_q = \sum_{\substack{q\in C_{p+1}^{[\Lambda]\setminus\{\tau\setminus\eta\}} \\ \eta\subseteq q}} \dot{\boldsymbol{L}}_q = \dot{\boldsymbol{L}}_\eta \sum_{q\in C_{p+1-m}^{[\Lambda]\setminus\{\tau\}}} \dot{\boldsymbol{L}}_q. \tag{143}$$

Let us recall that the term $\sum_{q\in C_{p+1}^{[\Lambda]}} \dot{\boldsymbol{L}}_q$ represents the $(p+1)$-th elementary symmetric polynomial for the set $\boldsymbol{L} \triangleq \{L_\lambda\}_{\lambda=1}^{\Lambda}$, and that the elementary symmetric polynomials satisfy Property 1. Hence, for any set of positive integers Ω and any integer $i \notin \Omega$, we can rewrite (7) in Property 1 using the above notation as

$$L_i \sum_{q\in C_p^\Omega} \dot{\boldsymbol{L}}_q = \sum_{q\in C_{p+1}^{\{\Omega\cup i\}}} \dot{\boldsymbol{L}}_q - \sum_{q\in C_{p+1}^{\Omega}} \dot{\boldsymbol{L}}_q. \tag{144}$$

This is equivalent to saying that the sum over all the terms $\dot{\boldsymbol{L}}_q$ (with $|q| = p+1$) that include L_i is equal to the sum over all the terms $\dot{\boldsymbol{L}}_q$ (with $|q| = p+1$) minus the sum over all the terms $\dot{\boldsymbol{L}}_q$ ($|q| = p+1$) that do not include L_i. This intuitive relation will prove important for the derivation.

Note that $\dot{\boldsymbol{L}}_\eta = \prod_{i\in\eta} L_i$. Then, we can successively apply (144) to (143) for all the L_i, $i \in \eta$, $|\eta| = m$, such that (143) is expanded in 2^m sums as

$$\sum_{\substack{q\in C_{p+1}^{[\Lambda]} \\ q\cap\tau=\eta}} \dot{\boldsymbol{L}}_q = \sum_{\kappa\subseteq\eta} (-1)^{|\kappa|} \sum_{q\in C_{p+1}^{[\Lambda]\setminus\{\tau\setminus\{\eta\setminus s\}\}}} \dot{\boldsymbol{L}}_q \tag{145}$$

$$= \sum_{k=0}^{m} (-1)^k \sum_{\substack{\kappa\subseteq\eta \\ |\kappa|=k}} \sum_{q\in C_{p+1}^{[\Lambda]\setminus\{\tau\setminus\{\eta\setminus\kappa\}\}}} \dot{\boldsymbol{L}}_q. \tag{146}$$

Thus, from (141), (142), and (146), we obtain that

$$a_\tau^{(p)} = \sum_{m=0}^{\min(j,p+1)} \sum_{\substack{\eta\subseteq\tau \\ |\eta|=m}} \sum_{k=0}^{m} \frac{(-1)^k}{j+1-m} \sum_{\substack{\kappa\subseteq\eta \\ |\kappa|=k}} \sum_{q\in C_{p+1}^{[\Lambda]\setminus\{\tau\setminus\{\eta\setminus\kappa\}\}}} \dot{\boldsymbol{L}}_q. \tag{147}$$

Since $\{\tau\setminus\{\eta\setminus\kappa\}\}$ can be the same set for different η, κ, let us count how many times the term $\sum_{q\in C_{p+1}^{[\Lambda]\setminus\{w\}}} \dot{\boldsymbol{L}}_q$ appears in (147) for a certain $w \subseteq [\tau]$, $|w| = i$. Let us fix m (i.e., the cardinality of $|\eta|$) and $j = |\tau|$. It follows that $|w| = |\{\tau\setminus\{\eta\setminus\kappa\}\}| = j - (m-k)$. We have that $w = \tau\setminus\{\eta\setminus\kappa\} = \{\tau\setminus\eta\}\cup\kappa$. This implies that $\kappa \subseteq w$. Furthermore, for any $\kappa \subseteq w$, there exists a distinct and unique η such that $w = \{\tau\setminus\eta\}\cup\kappa$. Since there are $\binom{|w|}{|\kappa|} = \binom{i}{k} = \binom{i}{i+m-j}$ possible $\kappa \subseteq w$ of cardinality k, each of the terms $\sum_{q\in C_{p+1}^{[\Lambda]\setminus\{w\}}} \dot{\boldsymbol{L}}_q$ appears in (147) exactly $\binom{i}{i+m-j}$ times for a particular m, i and j.

Let $\Theta_{\setminus i}$ denote the sum over all $q \in C_{p+1}^{[\Lambda]\setminus\{w\}}$ for any w of cardinality $|w| = i$. This $\Theta_{\setminus i}$ takes the form

$$\Theta_{\setminus i} \triangleq \sum_{\substack{w\subseteq\tau \\ |w|=i}} \sum_{q\in C_{p+1}^{[\Lambda]\setminus\{w\}}} \dot{\boldsymbol{L}}_q. \tag{148}$$

From (148) and the fact that the term $\sum_{q\in C_{p+1}^{[\Lambda]\setminus\{w\}}} \dot{\boldsymbol{L}}_q$ appears in (147) exactly $\binom{i}{i+m-j}$ times for a particular m, i and j, and after applying $k = i+m-j$, it follows that

$$\sum_{k=0}^{m} (-1)^k \sum_{\substack{\eta\subseteq\tau \\ |\eta|=m}} \sum_{\substack{\kappa\subseteq\eta \\ |\kappa|=k}} \sum_{q\in C_{p+1}^{[\Lambda]\setminus\{\tau\setminus\{\eta\setminus\kappa\}\}}} \dot{\boldsymbol{L}}_q = \sum_{i=j-m}^{j} (-1)^{i+m-j} \binom{i}{i+m-j} \Theta_{\setminus i}, \tag{149}$$

where we have substituted $k = i+m-j$.

Then, we can apply (149) into (147) to obtain that

$$a_\tau^{(p)} = \sum_{m=0}^{\min(j,p+1)} \sum_{i=j-m}^{j} \frac{(-1)^{i+m-j}}{j+1-m} \binom{i}{i+m-j} \Theta_{\setminus i}. \tag{150}$$

This expression of $a_\tau^{(p)}$ can be further simplified. Before continuing, let us take a look at the term $\Theta_{\setminus i}$. We show in the following that not all the components of $\Theta_{\setminus i}$ will impact the optimization of $a_\tau^{(p)}$.

2) Reducing $\Theta_{\setminus i}$ to its meaningful components: We recall that, as expressed in (137) in Lemma 11, our goal is to consider a single element ν for a given set ϕ, such that $\tau \triangleq \{\nu \cup \phi\}$, and obtain

$$\underset{\nu\in\{[\Lambda]\setminus\phi\}}{\operatorname{argmin}} \ \operatorname{sgn}(p-j) a_\tau^{(p)}. \tag{151}$$

In order to continue from (150), let us focus on the term $\Theta_{\setminus i} \triangleq \sum_{w\subseteq\tau,\ |w|=i} \sum_{q\in C_{p+1}^{[\Lambda]\setminus\{w\}}} \dot{\boldsymbol{L}}_q$, which has been defined in (148). $\Theta_{\setminus i}$ is composed of $\binom{j}{i}$ sums, one for each $w \subseteq \tau : |w| = i$. Interestingly, if $w \subseteq \tau$ is actually a subset of ϕ ($w \subseteq \phi$), the term $\sum_{q\in C_{p+1}^{[\Lambda]\setminus\{w\}}} \dot{\boldsymbol{L}}_q$ is the same no matter which value in $\{[\Lambda]\setminus\phi\}$ is selected as ν. Thus, such terms are irrelevant for the optimization problem.

Let us then consider the remaining cases that do impact the optimization problem, and let us denote the sum of the $\binom{j-1}{i-1}$ subsets w in (148) which contain ν as $\Theta^{\backslash\nu}_{\backslash i-1}$, i.e.,

$$\Theta^{\backslash\nu}_{\backslash i-1} \triangleq \sum_{\substack{\chi\subseteq\phi\\|\chi|=i-1}} \sum_{q\in C^{[\Lambda]\backslash\{\chi,\nu\}}_{p+1}} \dot{\boldsymbol{L}}_q \tag{152}$$

such that we can define the term that impacts the optimization problem as

$$b^{(p)}_\tau \triangleq \sum_{m=0}^{\min(j,p+1)} \sum_{i=j-m}^{j} \frac{(-1)^{i+m-j}}{j+1-m} \binom{i}{i+m-j} \Theta^{\backslash\nu}_{\backslash i-1}, \tag{153}$$

where $\tau \triangleq \{\phi \cup \nu\}$ and $b^{(p)}_\tau$ is obtained by substituting $\Theta_{\backslash i}$ in (150) by $\Theta^{\backslash\nu}_{\backslash i-1}$. Then, it follows that

$$\underset{\nu\in\{[\Lambda]\backslash\phi\}}{\operatorname{argmin}} \operatorname{sgn}(p-j) a^{(p)}_\tau = \underset{\nu\in\{[\Lambda]\backslash\phi\}}{\operatorname{argmin}} \operatorname{sgn}(p-j) b^{(p)}_\tau. \tag{154}$$

Next, we simplify $\Theta^{\backslash\nu}_{\backslash i-1}$ to later apply this result into $b^{(p)}_\tau$ and obtain Lemma 11.

3) Simplifying the term $\Theta^{\backslash\nu}_{\backslash i-1}$: A combination of $p+1$ elements in $[\Lambda]\backslash\{\nu,\chi\}$, where $\chi\subseteq\phi$ and $|\chi| = i-1$, can be expressed as the concatenation of ℓ elements of $\{\tau\backslash\{\nu,\chi\}\} = \{\phi\backslash\chi\}$ and $p+1-\ell$ elements of $\big\{[\Lambda]\backslash\{\nu,\chi\}\big\}\backslash\big\{\tau\backslash\{\nu,\chi\}\big\} = [\Lambda]\backslash\{\tau\}$, for any $\ell\in[j-i]_0$. Consequently, it follows that

$$\sum_{q\in C^{[\Lambda]\backslash\{\nu,\chi\}}_{p+1}} \dot{\boldsymbol{L}}_q = \sum_{\ell=0}^{j-i} \sum_{\substack{\mu\subseteq\{\phi\backslash\chi\}\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q. \tag{155}$$

Applying (155) into (152) yields

$$\Theta^{\backslash\nu}_{\backslash i-1} = \sum_{\substack{\chi\subseteq\phi\\|\chi|=i-1}} \sum_{\ell=0}^{j-i} \sum_{\substack{\mu\subseteq\{\phi\backslash\chi\}\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q \tag{156}$$

$$= \sum_{\ell=0}^{j-i} \underbrace{\Big(\sum_{\substack{\chi\subseteq\phi\\|\chi|=i-1}} \sum_{\substack{\mu\subseteq\{\phi\backslash\chi\}\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \Big)}_{E_{j-1,\ell}} \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q. \tag{157}$$

Next, we want to count how many times the last sum ($\sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q$) appears in $\Theta^{\backslash\nu}_{\backslash i-1}$. Consider some given i and ℓ. In the term $E_{j-1,\ell}$ in (157), a specific $\dot{\boldsymbol{L}}_\mu$ appears $\binom{j-1-\ell}{i-1}$ times. Then, it holds that

$$\Theta^{\backslash\nu}_{\backslash i-1} = \sum_{\ell=0}^{j-i} \binom{j-1-\ell}{i-1} \sum_{\substack{\mu\subseteq\phi\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q. \tag{158}$$

In the following, we incorporate in (153) the value of $\Theta^{\backslash\nu}_{\backslash i-1}$ obtained in (158) to derive (138) and hence Lemma 11.

4) Obtaining (138): Let us introduce the notation $\varpi \triangleq \min(j, p+1)$ for ease of readability. Then, we continue from (153) as

$$b^{(p)}_\tau \triangleq \sum_{m=0}^{\varpi} \sum_{i=j-m}^{j} \frac{(-1)^{i+m-j}}{j+1-m} \binom{i}{i+m-j} \Theta^{\backslash\nu}_{\backslash i-1} \tag{159}$$

$$\overset{(a)}{=} \sum_{i=j-\varpi}^{j} \sum_{m'=j-\varpi}^{i} \frac{(-1)^{i-m'}}{m'+1} \binom{i}{i-m'} \Theta^{\backslash\nu}_{\backslash i-1} \tag{160}$$

$$\overset{(b)}{=} \sum_{i=j-\varpi}^{j} \frac{(-1)^{i-j+\varpi}}{i+1} \binom{i}{j-\varpi} \Theta^{\backslash\nu}_{\backslash i-1}, \tag{161}$$

where (a) follows from interchanging the summations in (159) and applying the change of variable $m' = j-m$, and where (b) follows from solving the inner summation.

Let us now substitute in (161) the expression of $\Theta^{\backslash\nu}_{\backslash i-1}$ provided in (158), which leads to

$$b^{(p)}_\tau = \sum_{i=j-\varpi}^{j} \Bigg(\frac{(-1)^{i-j+\varpi}}{i+1} \binom{i}{j-\varpi} \times \sum_{\ell=0}^{j-i} \binom{j-1-\ell}{i-1} \sum_{\substack{\mu\subseteq\phi\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q \Bigg). \tag{162}$$

Since $\sum_{i=j-\varpi}^{j} \sum_{\ell=0}^{j-i} f(i,\ell) = \sum_{\ell=0}^{\varpi} \sum_{i=j-\varpi}^{j-\ell} f(i,\ell)$), we can interchange the summations to obtain that

$$b^{(p)}_\tau = \sum_{\ell=0}^{\varpi} \sum_{i=j-\varpi}^{j-\ell} F_{\ell,p,j} \sum_{\substack{\mu\subseteq\phi\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q, \tag{163}$$

where $F_{\ell,p,j} \triangleq \frac{(-1)^{i-j+\varpi}}{i+1} \binom{i}{j-\varpi} \binom{j-1-\ell}{i-1}$. Let us consider $\sum_{i=j-\varpi}^{j-\ell} F_{\ell,p,j}$. To simplify the notation, let us define $h \triangleq j-\varpi$ and $g = \varpi - \ell$. Thus, it follows that

$$\sum_{i=j-\varpi}^{j-\ell} F_{\ell,p,j} = \sum_{i=h}^{h+g} \frac{(-1)^{i-h}}{i+1} \binom{i}{h} \binom{h+g-1}{i-1} \tag{164}$$

$$= \frac{-1}{(h+g+1)(h+g)} = \frac{-1}{(j-\ell)(j-\ell+1)}. \tag{165}$$

Incorporating (165) into (163) yields

$$b^{(p)}_\tau = \sum_{\ell=0}^{\varpi} \frac{-1}{(j-\ell)(j-\ell+1)} \sum_{\substack{\mu\subseteq\phi\\|\mu|=\ell}} \dot{\boldsymbol{L}}_\mu \sum_{q\in C^{[\Lambda]\backslash\{\tau\}}_{p+1-\ell}} \dot{\boldsymbol{L}}_q, \tag{166}$$

which concludes the proof of (138), and consequently it also concludes the proof of Lemma 11. □

Appendix VI: Proof of Proposition 2

In the following, we prove that the sequence $\{\tilde{c}^{(p)}_{\tau^\star_j}\}$ is a decreasing sequence in $j\in[\Lambda]_0 = \{0\cup[\Lambda]\}$, where we recall that $\tilde{c}^{(p)}_\tau$ is given by

$$\tilde{c}^{(p)}_\tau \triangleq \frac{\sum_{q\in C^{[\Lambda]}_{p+1}} \frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|} \prod_{j=1}^{p+1} L_{q(j)}}{\sum_{\ell\in C^{[\Lambda]}_p} \prod_{j=1}^{p} L_{\ell(j)}},$$

and where $\tau_j^\star \triangleq \text{argmin}_{\tau\subset[\Lambda]_0,\,|\tau|=j}\,\tilde{c}_\tau^{(p)}$. Since the denominator of $\tilde{c}_\tau^{(p)}$ is the same for any τ, we focus on the numerator. First, let us denote the numerator of $\tilde{c}_\tau^{(p)}$ by $A(p,\tau)$, such that

$$A(p,\tau) \triangleq \sum_{q\in C_{p+1}^{[\Lambda]}} \frac{p+1-|q\cap\tau|}{|\tau|+1-|q\cap\tau|}\prod_{j=1}^{p+1} L_{q(j)}. \tag{167}$$

Hence, we need to prove that for any $0\leq j\leq \Lambda-1$ it holds that

$$A(p,\ \tau_j^\star) > A(p,\ \tau_{j+1}^\star). \tag{168}$$

Let us consider an arbitrary j, $0\leq j\leq\Lambda-1$. We select a set τ' with cardinality $j+1$ that includes $\tau_j^\star$, and we write τ' as $\tau'=\{\tau_j^\star\cup r\}$, where $r\in\{[\Lambda]\setminus\tau_j^\star\}$. Then, it follows from (167) that

$$A(p,\tau') = \sum_{q\in C_{p+1}^{[\Lambda]}} \frac{p+1-|q\cap\{\tau_j^\star\cup r\}|}{(j+1)+1-|q\cap\{\tau_j^\star\cup r\}|}\prod_{j=1}^{p+1} L_{q(j)}. \tag{169}$$

Note that $\prod_{j=1}^{p+1} L_{q(j)}$ is independent of $\tau', \tau_j^\star$. Furthermore, it holds that

$$\frac{p+1-|q\cap\{\tau_j^\star\cup r\}|}{(j+1)+1-|q\cap\{\tau_j^\star\cup r\}|} < \frac{p+1-|q\cap\tau_j^\star|}{j+1-|q\cap\tau_j^\star|} \tag{170}$$

for any $p\in\Lambda$, $r\in\{[\Lambda]\setminus\tau_j^\star\}$, $q\in C_{p+1}^{[\Lambda]}$. Merging (169) and (170) yields

$$A(p,\tau') < \sum_{q\in C_{p+1}^{[\Lambda]}} \frac{p+1-|q\cap\tau_j^\star|}{|\tau_j^\star|+1-|q\cap\tau_j^\star|}\prod_{j=1}^{p+1} L_{q(j)} = A(p,\tau_j^\star).$$

By definition, $A(p,\tau')\geq A(p,\tau_{j+1}^\star)$ for any τ' such that $|\tau'|=j+1$. Thus,

$$A(p,\tau_{j+1}^\star) \leq A(p,\tau') < A(p,\tau_j^\star), \tag{171}$$

which concludes the proof of Proposition 2. □

REFERENCES

[1] E. Parrinello and P. Elia, "Coded caching with optimized shared-cache sizes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.

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

[3] 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, no. 1, pp. 647–663, Jan 2019.

[4] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2016, pp. 161–165.

[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, no. 2, pp. 1281–1296, Feb 2018.

[6] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1146–1158, Feb 2017.

[7] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 349–366, Jan 2018.

[8] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, Aug 2015.

[9] S. Jin, Y. Cui, H. Liu, and G. Caire, "A new order-optimal decentralized coded caching scheme with good performance in the finite file size regime," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5297–5310, Aug. 2019.

[10] N. Zhang and M. Tao, "Fitness-aware coded multicasting for decentralized caching with finite file packetization," *IEEE Wireless Commun. Letters*, vol. 7, no. 5, pp. 740–743, Oct. 2018.

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

[12] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan 2018.

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

[14] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1291–1295.

[15] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[16] E. Ozfatura, S. Ulukus, and D. Gündüz, "Coded distributed computing with partial recovery," *IEEE Trans. Inf. Theory*, vol. 68, no. 3, pp. 1945–1959, Mar. 2022.

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

[18] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental Limits of Cache-Aided Interference Management," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3092–3107, 2017.

[19] E. Lampiris, A. Bazco-Nogueras, and P. Elia, "Resolving the feedback bottleneck of multi-antenna coded caching," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2331–2348, 2022.

[20] S. P. Shariatpanahi, G. Caire, and B. H. Khalaj, "Physical-layer schemes for wireless coded caching," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2792–2807, 2019.

[21] A. Tolli, S. P. Shariatpanahi, J. Kaleva, and B. H. Khalaj, "Multi-antenna interference management for coded caching," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2091–2106, 2020.

[22] H. Zhao, A. Bazco-Nogueras, and P. Elia, "Coded caching gains at low SNR over Nakagami fading channels," in *Asilomar Conf. Signals, Syst., Comput. (ACSSC)*, Nov. 2021.

[23] ——, "Vector coded caching multiplicatively increases the throughput of realistic downlink systems," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2683–2698, 2023.

[24] 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, no. 5, pp. 3142–3160, May 2017.

[25] A. Bazco-Nogueras and P. Elia, "Rate-memory trade-off for the cache-aided MISO broadcast channel with hybrid CSIT," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2021.

[26] E. Lampiris and P. Elia, "Full coded caching gains for cache-less users," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7635–7651, 2020.

[27] H. Joudeh, E. Lampiris, P. Elia, and G. Caire, "Fundamental limits of wireless caching under mixed cacheable and uncacheable traffic," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4747–4767, 2021.

[28] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[29] E. Parrinello, A. Ünsal, and P. Elia, "Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2252–2268, 2020.

[30] N. S. Karat, S. Dey, A. Thomas, and B. S. Rajan, "An optimal linear error correcting delivery scheme for coded caching with shared caches," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 1217–1221.

[31] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Benefits of edge caching with coded placement for asymmetric networks and shared caches," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 4, pp. 1240–1252, 2021.

[32] M. Dutta and A. Thomas, "Decentralized coded caching for shared caches," *IEEE Commun. Letters*, vol. 25, no. 5, pp. 1458–1462, 2021.

[33] B. Asadi and L. Ong, "Centralized caching with shared caches in heterogeneous cellular networks," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2019, pp. 1–5.

[34] S. Sasi and B. S. Rajan, "Multi-access coded caching scheme with linear sub-packetization using PDAs," *IEEE Trans. Commun.*, vol. 69, no. 12, pp. 7974–7985, 2021.

[35] M. Cheng, D. Liang, K. Wan, M. Zhang, and G. Caire, "A novel transformation approach of shared-link coded caching schemes for multiaccess networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 849–854.

[36] K. S. Reddy and N. Karamchandani, "Structured index coding problem and multi-access coded caching," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 4, pp. 1266–1281, 2021.

[37] B. Serbetci, E. Parrinello, and P. Elia, "Multi-access coded caching: gains beyond cache-redundancy," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.

[38] F. Brunero and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 1037–1056, 2023.

[39] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching-aided coded multicasting with multiple random requests," in *Proc. IEEE Inf. Theory Workshop (ITW)*, May 2015, pp. 1–5.

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

[41] H. Xu, C. Gong, and X. Wang, "Efficient file delivery for coded prefetching in shared cache networks with multiple requests per user," *IEEE Trans. Commun.*, vol. 67, no. 4, pp. 2849–2865, 2019.

[42] 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, no. 12, pp. 8402–8413, 2013.

[43] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1176–1188, June 2018.

[44] E. Peter, K. K. Krishnan Namboodiri, and B. Sundar Rajan, "Coded caching with shared caches and private caches," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2023, pp. 119–124.

[45] B. Merikhi and M. R. Soleymani, "Cache-aided delivery network in a shared cache framework with correlated sources," in *ACM Int. Symp. QoS and Security for Wireless and Mobile Netw.*, 2022, p. 121–129. [Online]. Available: https://doi.org/10.1145/3551661.3561372

[46] A. Asadzadeh and G. Caire, "Coded caching with small subpacketization via spatial reuse and content base replication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 2982–2986.

[47] M. J. Salehi, E. Parrinello, S. P. Shariatpanahi, P. Elia, and A. Tölli, "Low-complexity high-performance cyclic caching for large MISO systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 5, pp. 3263–3278, 2021.

[48] E. Peter and B. Sundar Rajan, "Multi-antenna coded caching for shared caches with arbitrary user-to-cache association," *IEEE Commun. Letters*, vol. 27, no. 7, pp. 1729–1733, 2023.

[49] H. Zhao, A. Bazco-Nogueras, and P. Elia, "Wireless coded caching can overcome the worst-user bottleneck by exploiting finite file sizes," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5450–5466, 2022.

[50] ——, "Wireless coded caching with shared caches can overcome the near-far bottleneck," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 350–355.

[51] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Coded caching and storage planning in heterogeneous networks," in *IEEE Wireless Commun. and Netw. Conf. (WCNC)*, March 2017.

[52] C.-H. Chang, B. Peleato, and C.-C. Wang, "Coded caching with full heterogeneity: Exact capacity of the two-user/two-file case," *IEEE Trans. Inf. Theory*, vol. 68, no. 11, pp. 7060–7076, 2022.

[53] X. Peng, J. Zhang, S. H. Song, and K. B. Letaief, "Cache size allocation in backhaul limited wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–6.

[54] T. Liu, S. Zhou, and Z. Niu, "Joint optimization of cache allocation and content placement in urban vehicular networks," in *Proc. IEEE Global Conf. Commun. (GLOBECOM)*, 2018, pp. 1–6.

[55] J. Liao, K. Wong, Y. Zhang, Z. Zheng, and K. Yang, "Coding, multicast, and cooperation for cache- enabled heterogeneous small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6838–6853, 2017.

[56] F. Brunero and P. Elia, "Unselfish coded caching can yield unbounded gains over selfish caching," *IEEE Trans. Inf. Theory*, vol. 68, no. 12, pp. 7871–7891, 2022.

[57] A. Tang, S. Roy, and X. Wang, "Coded caching for wireless backhaul networks with unequal link rates," *IEEE Trans. Commun.*, vol. 66, no. 1, pp. 1–13, 2018.

[58] S. Saeedi Bidokhti, M. Wigger, and A. Yener, "Benefits of cache assignment on degraded broadcast channels," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 6999–7019, 2019.

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

[60] 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.

[61] C. P. Niculescu, "A new look at Newton's inequalities," *Journal of Inequalities in Pure & Applied Mathematics (JIPAM)*, vol. 1, Paper No. 17, 14 p., 2000.

[62] M. Salehi, A. Tölli, and S. P. Shariatpanahi, "A multi-antenna coded caching scheme with linear subpacketization," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020.

[63] N. Mital, D. Gündüz, and C. Ling, "Coded caching in a multi-server system with random topology," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4620–4631, 2020.

[64] F. Arbabjolfaei, B. Bandemer, Y. H. Kim, E. Şaşoğlu, and L. Wang, "On the capacity region for index coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul 2013, pp. 962–966.

[65] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 135–139.

[66] I. Newton, *Arithmetica universalis: sive de compositione et resolutione arithmetica liber*. Apud Marcum Michaelem Rey, 1761.

[67] R. Stanley, "Log-concave and unimodal sequences in algebra, combinatorics, and geometry," *Annals of the New York Academy of Sciences*, vol. 576, pp. 500–535, 12 2006.

[68] M. Lin and N. S. Trudinger, "On some inequalities for elementary symmetric functions," *Bulletin of the Australian Mathematical Society*, vol. 50, no. 2, pp. 317–326, 1994.

Emanuele Parrinello (Member, IEEE) received the B.Sc. degree in telecommunication engineering and the M.Sc. degree (Hons.) in communications and computer networks engineering from the Politecnico di Torino in 2015 and 2018, respectively, the M.Sc. degree in mobile communications from the EURECOM, Télécom Paris, in 2018, and the Ph.D. degree from Sorbonne University, France, in 2021. He is currently employed as a Signal Processing Engineer at CEVA, France. His research interests lie in caching networks, network information theory, wireless communication, and signal processing

Antonio Bazco-Nogueras (Member, IEEE) received the B.S. and M.S. degrees in Telecommunications Engineering from University of Zaragoza, Spain, in 2014 and 2016, respectively. He obtained the Ph.D. degree from Sorbonne Université, Paris, France, in collaboration with the Mitsubishi Electric R&D Centre Europe, Rennes, France, in 2019. He was a post-doctoral researcher at EURECOM, Sophia-Antipolis, France, from 2020 to 2021. He is currently a post-doctoral researcher at IMDEA Networks Institute, Madrid, Spain. He is recipient of the Madrid Talent Attraction Grant 2021. His research interests include multi-user information theory, intelligent and self-configuring networks, decentralized systems, content delivery networks, and cooperative wireless networks.

Petros Elia (Member, IEEE) 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.