Optimization of Heterogeneous Coded Caching

Alexander Michael Daniel and Wei Yu

arXiv:1708.04322v1 [cs.IT] 14 Aug 2017

Abstract—This paper aims to provide an optimization framework for coded caching that accounts for various heterogeneous aspects of practical systems. An optimization theoretic perspective on the seminal work on the fundamental limits of caching by Maddah Ali and Niesen is first developed, whereas it is proved that the coded caching scheme presented in that work is the optimal scheme among a large, non-trivial family of possible caching schemes. The optimization framework is then used to develop a coded caching scheme capable of handling simultaneous non-uniform file length, non-uniform file popularity, and non-uniform user cache size. Although the resulting full optimization problem scales exponentially with the problem size, this paper shows that tractable simplifications of the problem that scale as a polynomial function of the problem size can still perform well compared to the original problem. By considering these heterogeneities both individually and in conjunction with one another, insights into their interactions and influence on optimal cache content are obtained.

Index Terms—Coded caching, linear programming, non-uniform popularity, non-uniform cache size, non-uniform file length

I. INTRODUCTION

A. Background

Caching technologies stand poised to make an important contribution to future 5G cellular networks [1]. One such technique is *coded caching*, which, roughly speaking, is the idea of using carefully designed user cache content to enable content delivery via coded multicast transmissions. (The cached contents themselves are uncoded.) First introduced by Maddah-Ali and Niesen in [2], [3], coded caching has since been the subject of a great number of studies seeking to extend the original scheme into more practical scenarios. The coded caching scheme of [2], [3] is developed for a system in which a central server has complete knowledge of user numbers and identities, users have identical cache sizes and make a single download request, transmission occurs over an error free link, and files are of equal length and popularity. Subsequent work has since extended the coded caching idea to the decentralized system [4], and to systems with non-uniform file popularity [5]–[17], non-uniform file length [18], [19], multiple user requests [20]–[23], non-uniform cache size [24]–[27], and non-uniform channel quality [28]–[40].

The aforementioned works typically either discuss how the scheme of [2], [3] should be modified to accommodate the considered heterogeneity, or develop an entirely new scheme that enables coded multicast transmissions while accommodating the aforementioned heterogeneity. Most of these papers, however, consider only one type of non-uniformity. While this is sensible from the viewpoint of understanding how each heterogeneity affects coded caching systems by itself, practical systems would have to account multiple types of non-uniform parameters. Moreover, we cannot in general expect the effects of these non-uniformities to be additive. It is thus important to consider combinations of heterogeneities, which, to the best of our knowledge, only a few recent works have started to explore: the recent work [19] examines the achievable rate region for a system serving two users with two files, where the user cache sizes and file lengths are not necessarily uniform, while some of the aforementioned work on caching with non-uniform channel quality exploits heterogeneous cache size to rectify disparities in channel quality (see e.g. [33] and references therein).

B. Main Contributions

This paper proposes an optimization theoretic framework to design caching schemes capable of accommodating non-uniform file length, non-uniform file popularity, and non-uniform user cache size at the same time. More specifically, we design a caching scheme that uses a generalized version of the transmission scheme of [4], paired with an optimization problem designed to yield the optimally coded content. This optimization problem, although convex, has the number of variables, constraints, and objective function terms that scale exponentially with the problem size, so subsequently this paper develops high-quality simplifications that scale polynomially with the problem size, yet perform well compared to the original problem. The proposed optimization approach only yields numerical answers corresponding to the optimized caching schemes, but also generate practical insight into the problems considered.

Optimization approaches have been used in the past for content placement for femtocaching systems [41], [42], but its use in the coded caching context has only appeared recently: for instance, [43] uses an information-theory based optimization problem to help characterize the achievable rate region for certain numbers of users and files in the case where all other systems parameters are uniform. More related is the recent work [27] in which an optimization framework similar to the one used here is employed to develop a caching scheme in the case of non-uniform cache size. Crucially, both approaches design cache content in terms of the subsets of users who have cached the content (see Section III). While we directly express our transmission scheme in terms of that cache content, the approach in [27] is to further design two sets of variables through which the transmission scheme is expressed. Moreover, the framework used in [27]

does not distinguish between different files beyond designing a scheme for the worst-case scenario where users request different files. This prevents their framework from addressing file heterogeneity, whereas ours allows for it. Finally, while the optimization problem in [27] suffers from the same exponential scaling problem that the general problem here has, they do not present any tractable methods of solution like we do here.

During the preparation of this paper, we became aware of independent work [17], which uses an optimization framework essentially the same as the one proposed here to study the case of non-uniform file popularity. While independently and simultaneously developed, a number of results in [17] are echoed in this paper. Specifically, both works develop the same exponential-order general optimization problem; then a simplified polynomial-order optimization problem is developed for the non-uniform popularity case (Section IV-C of this paper), although the exact formulations are different. Moreover, both works show that in the special case of uniform popularity, the caching and delivery scheme of [2], [3] is the optimal solution (Section III of this paper); however, the proof in [17] is quite involved, while a much simpler proof is presented here. Ultimately, the focus of [17] is to study the case of non-uniform popularity in great depth, while here, it is only considered as an intermediate step towards the study of the interactions of several heterogeneities at the same time. Thus, despite the similarities, both papers develop many unique insights of practical significance. Indeed, the results of [27] and [17] taken jointly with the results of this paper suggest that the optimization framework common to all three works is likely to be a useful one for coded caching.

C. Notation

The notation $[a : b]$ is used as shorthand for the set of consecutive integers $\{a, a + 1, \ldots, b - 1, b\}$, and $[b]$ is used as an abbreviation of $[1 : b]$. The symbol $\oplus$ is used to denote the bitwise "XOR" operation between two or more files (i.e. strings of bits). Both l and $W^{(l)}$ are used to refer to the l-th file under consideration. For an arbitrary file $W^{(n)}$, $|W^{(n)}|$ refers to the length of the file, and $W_\mathcal{S}^{(l)}$, called a "subfile" of file $W^{(l)}$, refers to the portion of file l stored exclusively on the caches of the users in the set $\mathcal{S}$. For notational convenience, notation of the form $W_{123}^{(l)}$ is used instead of $W_{\{1,2,3\}}^{(i)}$ (for the $\mathcal{S} = \{1, 2, 3\}$ case in this example), returning only to the latter notation if necessary to resolve ambiguity. For a set $\mathcal{S}$, $\mathcal{P}(\mathcal{S})$ refers to the power set of $\mathcal{S}$. For a real number t, $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling functions, respectively.

We define the binomial coefficient $\binom{n}{k}$ in the usual way for $0 \le k \le n$, i.e. $\binom{n}{k} = n!/(k!(n-k)!)$, but for $n < 0$ or $k > n$, we take $\binom{n}{k} = 0$. Moreover, the notation of the so-called "multinomial coefficient" is used, defined as:

$$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1! k_2! \ldots k_m!},$$

where $k_1 + k_2 + \cdots + k_m = n$. We use the notation $\sum_{i=a}^{b} n_i$ in the usual way when $a \le b$, and take $\sum_{i=a}^{b} n_i = 0$ identically

if $a > b$. More generally, a sum over an empty set of indices is taken to equal zero. Finally, if a sum over an empty set is raised to the power of zero, we take the resulting value to be 1; this is simply used for notational convenience.

D. Organization

The organization of the rest of the paper is as follows. Section II introduces the optimization framework used in this paper and Section III uses this framework to provide a new interpretation and understanding of [2]–[4] by showing how they represent feasible points in the optimization problem. Sections IV-VI use the framework to develop tractable simplifications of the original problem for different types and combinations of non-uniformities. Section VII concludes the paper with a summary and discussion of main results.

II. OPTIMIZATION PERSPECTIVE ON CODED CACHING

We begin by providing an optimization perspective on the coded caching schemes of Maddah-Ali and Niesen for both the centralized [2], [3] and the decentralized [4] cases. The transmission scenario consists of a server with a set of N files, $\mathcal{F} = \{1, 2, \ldots, N\} = [N]$, serving a set of K users, $\mathcal{U} = \{1, 2, \ldots, K\} = [K]$, over a shared, error-free link. For full generality, we allow each file l to have a distinct length of F_l bits and a distinct probability p_l of being requested, and allow each user k to have arbitrary cache size of M_k bits.

Central to the coded caching scheme of [2]–[4] is the partitioning of each file $l \in \mathcal{F}$ into subfiles $W_\mathcal{S}^{(l)}$, indexed by all subsets of $\mathcal{S} \subseteq \mathcal{U}$. (No two subfiles have a non-empty intersection; together the subfiles jointly reconstruct the original file.) In the caching phase, each user k caches

$$\bigcup_{l, \mathcal{S} \in \mathcal{P}(\mathcal{U} \setminus \{k\})} W_{\mathcal{S} \cup k}^{(l)} \tag{1}$$

without coding. In the content delivery phase, given the set of user requests $\mathbf{d} = [d_1, \ldots, d_K]$, where d_k denotes the index of the file requested by user k, the server transmits

$$\bigoplus_{k \in \mathcal{S}} W_{\mathcal{S} \setminus \{k\}}^{(d_k)} \tag{2}$$

over the shared link with zero-padding if necessary, for each $\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset$, so that together with uncoded content stored a priori in each user's local cache, all users are guaranteed to be able to reconstruct the entirety of their requested files.

Any coded caching scheme (with coded transmission and uncoded cache) can be described using this "subset partitioning" representation [4]. Different caching strategies differ in their partitioning of the subfiles. But instead of thinking of the above as a way of *labelling* the cache contents, we can also use this to *design* the cache content, and regard the size of each subfile $W_\mathcal{S}^{(l)}$ as design variables. Consequently, instead of designing cache contents by assigning discrete bits to sets, the problem can be simplified by designing only the sizes of the subfiles: if we have $|W_\emptyset^{(l)}| = b_\emptyset^{(l)}, |W_1^{(l)}| = b_1^{(l)}, \ldots, |W_\mathcal{U}^{(l)}| = b_\mathcal{U}^{(l)}$, then the first $b_\emptyset^{(l)} F_l$ bits of $W^{(l)}$ are

assigned to $W_\emptyset^{(l)}$, the next $b_1^{(l)} F_l$ bits to $W_1^{(l)}$, and so on, assigning the final $b_\mathcal{U}^{(l)} F_l$ bits to $W_\mathcal{U}^{(l)}$. (Formally, this requires that $b_\mathcal{S}^{(l)} \in \{0, 1/F_l, \ldots (F_l - 1)/F_l, 1\}$, but for large F_l, this can be relaxed to $b_\mathcal{S}^{(l)} \in [0, 1]$ without any significant loss.)

For example, consider the centralized scenario wherein the numbers and identities of users are known in advance, so the server has the ability to design the cache content of each user. The coded caching scheme of [2], [3] (assuming uniform file length, cache size and uniform popularity) sets $|W_\mathcal{S}|$ to be non-zero for only the $\mathcal{S}$'s with $|\mathcal{S}| = t = KM/N$. In the decentralized setting of [4], all subfiles have non-zero sizes due to the use of random cache content. More generally, $|W_\mathcal{S}^{(l)}|$ can be explicitly designed.

The design of coded caching in this way imposes some natural constraints on the subfile sizes. First, the subfiles together must contain the entire file, i.e.

$$\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U})} |W_\mathcal{S}^{(l)}| = F_l, \quad \forall l \in \mathcal{F}. \tag{3}$$

Second, denoting the amount of cache dedicated to file l by user k as $\mu_{k,l}$, the amount of a file cached by a user is expressed as

$$\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U} \setminus \{k\})} |W_{\mathcal{S} \cup k}^{(l)}| \leq \mu_{k,l}, \quad \forall k \in \mathcal{U}, \forall l \in \mathcal{F}, \tag{4}$$

where

$$\sum_{l=1}^{N} \mu_{k,l} = M_k, \forall k \in [K]. \tag{5}$$

Finally, the subfiles cannot have a negative size:

$$|W_\mathcal{S}^{(l)}| \geq 0, \quad \forall \mathcal{S} \in \mathcal{P}(\mathcal{U}), \quad \forall l \in \mathcal{F}. \tag{6}$$

In the transmission defined in (2), zero-padding is neeeded whenever the subfiles do not have the same length, so the length of a single transmission is determined by the largest subfile in the transmission. For a vector of user requests $\mathbf{d}$, the number of bits sent to satisfy user requests given in $\mathbf{d}$ is thus

$$R_\mathbf{d} = \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}} \left\{ |W_{\mathcal{S} \setminus \{k\}}^{(d_k)}| \right\} \tag{7}$$

The set of choices of $|W_\mathcal{S}^{(l)}|$ define a broad family of caching schemes. To find the most efficient caching strategy among this family of schemes that minimize the expected delivery rate over all demand requests, we can formulate the following optimization problem:

$$\text{minimize} \quad \mathbb{E}[R_\mathbf{d}] = \sum_{\mathbf{d} \in \mathcal{F}^K} p(\mathbf{d}) \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}} \left\{ |W_{\mathcal{S} \setminus \{k\}}^{(d_k)}| \right\} \tag{8}$$

$$\text{subject to} \quad (3)\text{-}(6). \tag{9}$$

We note before continuing that the transmission scheme described above is the same for all possible user requests, which may be suboptimal if there is repetition in the users' requests, i.e. the same file is requested by more than one user. However, it can readily be shown that the probability

of each user requesting a distinct file goes to one as $N \to \infty$ of a fixed K. Since it is likely that $N \gg K$ in practice, the optimization formulation (8) can be therefore be used without fear of significant loss.

The optimization problem (8)-(9) is convex in the $|W_\mathcal{S}|$ variables. However, there are $N2^K$ variables, $N^K(2^K - 1)$ summands in the objective function, and $N2^K + KN + N + K$ constraints, making it impractical to solve for large-scale problems. The rest of this paper is dedicated to developing simplifications of (8)-(9) that allow for high quality (and even optimal) solution while maintaining a tractable problem size.

III. HOMOGENEOUS CODED CACHING

Consider the special case of problem (8)-(9) with uniform file lengths, $F_l = F, \forall l \in [N]$, uniform file popularities, $p_l = 1/N, \forall l \in [N]$, and uniform cache sizes $M_k = M, \forall k \in [K]$; this is the same system originally considered in [2], [3]. The symmetry of the resulting problem can be exploited to reduce the computational complexity of optimization. Specifically, define v_j such that $v_j = |W_\mathcal{S}|$ for all $\mathcal{S}$ such that $|\mathcal{S}| = j$ and for all files W; this reduces the number of optimization variables from an exponential number in K to a linear number in K. Since the file length, file popularity and cache size are all homogeneous, there is symmetry across both the users and the files, and so we would expect the solution to (8)-(9) for this uniform case to have this form, i.e., any two subfiles have the same size if their respective user sets are the same size. The summands of the objective function now simplify as

$$\max_{k \in \mathcal{S}} \left\{ |W_{\mathcal{S} \setminus \{k\}}^{(k)}| \right\} = |W_{\mathcal{S} \setminus \{k'\}}^{(k')}| = v_{|\mathcal{S}|-1}$$

where k' is any user in the set $\mathcal{S} \setminus \{k\}$, because each subfile in a given transmission is the same size. Since the files are equally popular, every request vector is equally likely, and since every transmission scheme requires the same number of bits to satisfy the requests, the average does not need to be taken across all the N^K possible demands as in (8). Finally, since $v_j F$ bits are sent to any subset of $j + 1$ users, and there are $\binom{K}{j+1}$ subsets of $j + 1$ users, the objective function becomes

$$\mathbb{E}[R_\mathbf{d}] = \sum_{j=0}^{K-1} \binom{K}{j+1} v_j F, \tag{10}$$

Note that the number of terms in the objective function now scales linearly in K instead of exponentially in K, because each term in (10) accounts for a combinatorial number of transmissions.

The constraints simplify as well. Since all files are of equal length, only one file reconstruction constraint is required. Then, because there are $\binom{K}{j}$ subsets of size j, the file reconstruction constraint becomes

$$\sum_{j=0}^{K} \binom{K}{j} v_j = F. \tag{11}$$

Moreover, since the cache sizes are uniform, only one cache constraint is needed, and since all file are homogeneous, an

equal amount of memory is allocated to each. The cache constraint thus simplifies to

$$\sum_{j=1}^{K} \binom{K-1}{j-1} v_j \le MF/N, \tag{12}$$

because there are $\binom{K-1}{j-1}$ subsets of size j that contain the index k. As a final step, we follow [2], [3] and normalize the file length $F = 1$ in (10)-(12). This yields the following linear programming problem with $K + 1$ variables, $K + 3$ constraints, and K terms in the objective function:

$$\text{minimize} \quad \sum_{j=0}^{K-1} \binom{K}{j+1} v_j \tag{13}$$

$$\text{subject to} \quad \sum_{j=0}^{K} \binom{K}{j} v_j = 1, \tag{14}$$

$$\sum_{j=1}^{K} \binom{K-1}{j-1} v_j \le M/N, \tag{15}$$

$$v_j \ge 0, \quad \forall j \in \mathcal{U}. \tag{16}$$

Note that the reduction to an optimization problem that scales as a linear function of K and as a constant function of N is possible because of the symmetry created by the uniform file length, file popularity, and cache size. Later in this paper, cases when one or more of these parameters are non-uniform are treated; but the reduction from the exponential order will no longer be without loss of generality.

Note also that the schemes of [2], [3] and [4] are all feasible points of this problem. In particular, assuming that $t = KM/N$ is an integer, the caching scheme of [2], [3] sets $v_t = 1/\binom{K}{t}$ and $v_j = 0$ if $j \ne t$, while the decentralized scheme of [4] sets the variables to be of the form $v_j = (M/N)^j (1 - M/N)^{K-j}$ for all j. For the non-integer t case, a similar scheme is also stated in [2], [3]. The following theorem shows that the caching scheme of [2], [3] is, in fact, the optimal scheme among the broad family of schemes discussed above. The theorem works for the cases of integer or non-integer t. We note that while a similar result for the integer t case exists in [17], the proof used here uses a novel reformulation over the probability simplex. It is considerably simpler and lends additional insight into the problem.

Theorem 1. Assume $M \le N$. The unique, optimal solution to (13)-(16) is:

• For $t = KM/N \in \mathbb{Z}$:

$$v_j^* = \begin{cases} 1/\binom{K}{t} & \text{if } j = t, \\ 0 & \text{if } j \ne t \end{cases} \tag{17}$$

• For $t = KM/N \notin \mathbb{Z}$:

$$v_j^* = \begin{cases} s/\binom{K}{\lfloor t \rfloor} & \text{if } j = \lfloor t \rfloor \\ (1-s)/\binom{K}{\lceil t \rceil} & \text{if } j = \lceil t \rceil \\ 0 & \text{else} \end{cases} \tag{18}$$

where $s = \lceil t \rceil - t$.

Proof: First, we make a change of variables: $a_j = v_j/\binom{K}{j}$; after some mild algebra, the original problem (13)-(16) can be reformulated as:

$$\text{minimize} \quad \sum_{j=0}^{K} \frac{K-j}{j+1} a_j \tag{19}$$

$$\text{subject to} \quad \sum_{j=0}^{K} j a_j \le t \tag{20}$$

$$\sum_{j=0}^{K} a_j = 1 \tag{21}$$

$$a_j \ge 0, \quad \forall j \in [0 : K]. \tag{22}$$

In this form, the optimization problem is more easily understood. Constraints (21) and (22) restrict the feasible space to the probability simplex. The key features of the formulation is that the coefficients of the objective function $\frac{K-j}{j+1}$ are decreasing in j with decreasing second differences, while the cache constraint (20) now has coefficients that increase linearly in j. Intuitively, the optimal solution would involve placing as much "probability mass" in high-j a_j variables in order to lower the objective function, but not so high as to violate the cache constraint. This results in the optimal a_j to concentrate around at most two consecutive j's close to t.

To complete the argument, we first observe that at the optimal solution $\mathbf{a}^*$ of (19)-(22), the cache constraint (20) must be tight, when $M \le N$. This is because if it were not, one can always shift some of the weight of a_j to a higher indexed a_{j+1} without violating the constraints while lowering the objective function. In the practical context, this means that the optimal caching strategy does not waste any cache space.

Second, for a similar reason, we can show that if a feasible solution $\mathbf{a} = [a_0, \dots, a_K]^T$ to (19)-(22) has two non-zero variables $a_{i_1} \ne 0$ and $a_{i_2} \ne 0$ such that $i_2 - i_1 \ge 2$, then $\mathbf{a}$ cannot be an optimal solution of (19)-(22). This is because for any such $\mathbf{a}$, we can always construct a better feasible solution $\bar{\mathbf{a}} = [\bar{a}_0, \dots, \bar{a}_K]^T$ in the following way. For some small Δ, set $\bar{a}_{i_1} = a_{i_1} - \Delta$, $\bar{a}_{i_1+1} = a_{i_1+1} + \Delta$, $\bar{a}_{i_2} = a_{i_2} - \Delta$, $\bar{a}_{i_2-1} = a_{i_2-1} + \Delta$, and $\bar{a}_j = a_j$ for all other j values. (If $i_1 + 1 = i_2 - 1$, then set $\bar{a}_{i_2-1} = a_{i_2-1} + 2\Delta$.) Note that such $\bar{\mathbf{a}}$ remains on the probability simplex; the cache constraint remains satisfied as $\sum_{j=0}^{K} j\bar{a}_j = \sum_{j=0}^{K} j a_j$, while the objective function decreases strictly, because the coefficients of a_j in the objective, $\frac{K-j}{j+1}$ is a decreasing function of j with decreasing second differences.

The above two observations imply that the optimal solution has either only one non-zero variable, or two non-zero variables that have adjacent indices, i.e. some j and $j+1$. In this case, the constraints for the optimal solution to (19)-(22) reduce to

$$j a_j + (j+1) a_{j+1} = t, \tag{23}$$

and

$$a_j + a_{j+1} = 1. \tag{24}$$

This system of equations has a unique solution. Indeed, due to (24) and the positivity constraints, (23) implies that t must be a *convex combination* of j and $j+1$. If $t \notin \mathbb{Z}$, the only integer possibility for j is $j = \lfloor t \rfloor$ and $j + 1 = \lceil t \rceil$; representing $t = s\lfloor t \rfloor + (1 - s)\lceil t \rceil$ for some $s \in (0, 1)$, it becomes clear that the unique solution to (23)-(24) is $a_j = s, a_{j+1} = 1 - s$ for $j = \lfloor t \rfloor$ and $s = \lceil t \rceil - t$. If $t \in \mathbb{Z}$, then we can set $j = t$, then the unique solution to (23)-(24) is $a_j = 1, a_{j+1} = 0$. Using the change of variables $v_i = a_i / \binom{K}{i}$, this gives the optimal solution to (13)-(16) as stated by (17)-(18). ∎

As a final remark for this section, we acknowledge that there exist stronger results about optimal coded caching schemes than Theorem 1 in the literature, for instance, [44] shows that a slightly modified version of the scheme in [2], [3] is the optimal coded caching scheme among all schemes with uncoded cache content using information theoretical upper bounds. The optimization theoretic perspective of this paper is nevertheless worthwhile in that it easily begets extensions to more practical non-uniform scenarios, which is the focus of the rest of this paper.

IV. CODED CACHING WITH HETEROGENEOUS FILES

We now move onto the coded caching problem with heterogeneous parameters. Although the optimization problem (8)-(9) is already capable of accounting for non-uniform file popularity, file length, and cache size, the problem size scales exponentially in system parameters, hence the optimization problem is intractable for practical system sizes. The main contributions of this and the next section are to develop simplifications to (8)-(9) that reduce the computational complexity of the problem while maintaining high-quality performance. Each non-uniformity is considered both individually and in conjunction with the others in order to gain insight into the interactions of their respective effects. We begin by examining the effect of non-uniform file popularity and non-uniform file length, but for now keep the cache sizes uniform across the users.

The procedure for each case is roughly as follows: first a new set of variables are defined that are intended to capture some structural feature of the problem, e.g. the v_j variables of the simplified homogeneous problem (13)-(16). Next, certain conditions, referred to as *memory inequality* constraints, are imposed on the new variables (see e.g. (35)), which forces feasible solutions to dedicate more cache memory to certain kinds of files, e.g. more popular files. While no *a priori* justification for the use of these variable and constraints is given, subsequent numerical results justify their use *a posteriori*.

These memory inequality constraints then allow the simplification of the objective function (8). First, the *max* function can be eliminated from (8), since the memory inequality constraints are sufficient to determine *a priori* which subfiles will be the maximum given some request vector $\mathbf{d}$. This in turn allows the expected rate to computed precisely in terms of *the largest file requested*, *the second largest file requested*, and so on, instead of in terms of the N^K possible request vectors. Since there are K files requested, and N

possibilities for each file, this ultimately reduces the scaling of the objective function from exponential to polynomial, although the objective function does not necessarily scale with NK precisely.

A. Prior Work

The effect of non-uniform file popularity, also referred to as non-uniform demands, on coded caching has been explored in a number of papers [5]–[16]. In [5], [6], Niesen and Maddah-Ali modify their decentralized scheme from [4] by first grouping users according to the popularity of their respective file requests, and then transmitting to the groups sequentially using the scheme from [4]. The authors of [7]–[11] develop an order-optimal scheme using random caching with a graph-based algorithm to design coded multicast transmissions, with [9] focusing specifically on the application of video delivery. In [10], [11], both the popularity of the files and their request correlation are considered when designing the caching and transmission scheme, while in [12]–[15], a heterogeneous network structure is considered, with file popularity organized in discrete levels. Using a novel random caching-based scheme, [16] is able to show order-optimality with a constant that is independent of the popularity distribution. Finally, we repeat earlier comments that [17] uses the same optimization framework that is used to study non-uniform popularity in great depth; we nevertheless show our (similar) work for the sake of exposition.

The literature on the effects of non-uniform file length is, however, comparatively scarce. To the best of our knowledge, Zhang et al [18] provide the only scheme designed to accommodate non-uniform file length for a general number of users. A scheme is provided that uses random caching with a transmission scheme similar to the one used in this paper, and upper and lower bounds on system performance are derived. In particular, [18] explores a random caching scheme where files are cached with a probability proportional to size of the file. Non-uniform file size is also explored in the recent letter [19], in which the achievable rate region for both non-uniform file size and non-uniform cache size is characterized, but only for the case of $K = 2$ users and $N = 2$ files.

Note that it can be argued that if files are indeed different sizes, they can be broken up into smaller packets of a constant size F' bits, and then treated as separate files. While this is a reasonable assumption while investigating other aspects of a coded caching scheme, there are two issues that need be addressed in practice. First, if a file is broken up into multiple pieces, then a user who seeks to to download the entire file must make multiple (correlated) requests to the server - a fact that should be accounted for in subsequent system design. The second practical issue comes from the fact that it is unclear how to set the common file size F': efficiency demands that F' be as large as possible so that any required headers represent a small proportion of the entire download, while at the same time, F' should also be small enough to divide files without significant remainder.

We therefore contend that heterogeneous file length is an important parameter that a practical system must capable

of accommodating in one way or another. The approach to handling non-uniform files sizes discussed above may indeed have some merit; some work has been done to analyze the case of multiple requests from users, see e.g. [20]–[23], and so an approach based on this technique may be viable. This paper, however, uses a different approach: files are not broken up into smaller files of equal lengths, and so the cache content is designed to accommodate their different lengths. The possibility of comparing the performance of the different approaches is left to future work.

To the best of our knowledge, there has been no work exploring the relationship between file length and popularity and the resulting effect on cache content. In the literature discussed above, it is noted (roughly) that more popular files should be allocated more cache memory, but also that larger files should be allocated more cache memory. Given that, in general, file lengths and popularity may not have any correlation, it is not clear how these non-uniformities jointly affect optimal cache content. These interactions are explored in the following.

B. Preliminaries

First, we introduce two lemmas. The first one is a classic result about binomial coefficients, while the second lemma is used to determine the probability that the file n is the k-th largest file requested; the proof of the latter is contained in Appendix A.

Lemma 1 (Chu-Vandermonde Convolution)**.** For $N, N_1, N_2,$ and n positive integers, with $N_1 + N_2 = N$ and $n \leq N$,

$$\binom{N}{n} = \sum_{k=0}^{n} \binom{N_1}{k}\binom{N_2}{n-k} \qquad (25)$$

Lemma 2. Consider K independent multinomial random trials with N possible outcomes per trial, with probabilities $\{p_1, p_2, \ldots, p_N\}$, denoted by $\mathbf{Z} \in [N]^K$, i.e., Z_i, the i-th element of $\mathbf{Z}$, is the outcome of the i-th trial. Let the random vector $\mathbf{Y}$ be a sorted version of $\mathbf{Z}$, but with index shifted by 1, so that Y_0 is the smallest element of $\mathbf{Z}$, Y_1 is the second smallest element, and so on. The probability mass function of Y_m is given by

$$\Pr[Y_0 = i] = \left(\sum_{l=i}^{N} p_l\right)^K - \left(\sum_{l=i+1}^{N} p_l\right)^K, \qquad (26)$$

$$\Pr[Y_1 = i] = \Pr[Y_0 = i] + K\left(\left(\sum_{l=1}^{i-1} p_l\right)\left(\sum_{l=i}^{N} p_l\right)^{K-1}\right.$$
$$\left. - \left(\sum_{l=1}^{i} p_l\right)\left(\sum_{l=i+1}^{N} p_l\right)^{K-1}\right), \qquad (27)$$

and for $m \in [2 : K-1]$,

$$\Pr[Y_m = 1] = \sum_{k=0}^{K-m-1} \binom{K}{m+1+k}$$
$$p_1^{m+1+k}(1-p_1)^{K-m-1-k} \qquad (28)$$

$$\Pr[Y_m = i] =$$

$$\binom{K}{K-m}\left(\left(\sum_{l=i}^{N} p_l\right)^{K-m} - \left(\sum_{l=i+1}^{N} p_l\right)^{K-m}\right)$$

$$\left(\sum_{l=1}^{i-1} p_l\right)^m + \sum_{k=0}^{K-2} \sum_{b=\max\{0,m-1-k\}}^{\min\{m-1,K-2-k\}}$$

$$\left(\binom{K}{2+k, b, K-2-k-b} p_i^{2+k}\right.$$

$$\left.\left(\sum_{l=1}^{i-1} p_l\right)^b \left(\sum_{l=i+1}^{N} p_l\right)^{K-2-k-b}\right) \qquad (29)$$

where the final expression is for $i \in [2 : N]$.

C. Optimization Formulation

The first step in reducing the exponential number of variables in (8)-(9) is the definition of a new set of $(K+1)N$ variables as

$$v_{l,j} = |W_{\mathcal{S}}^{(l)}|, \forall \mathcal{S} \text{ s.t. } |\mathcal{S}| = j, \forall l \in [N]. \qquad (30)$$

similar to the v_j variables used in Section III, except now there is a set of v_j variables for each file to capture the difference in length and popularity between files. Such a reduction enforces $|W_{\mathcal{S}}^{(l)}|$ to depend only on the cardinality of $\mathcal{S}$, which is a reasonable thing to do and in fact can be proved to be without loss of generality for the special case of non-uniform file popularity alone but with uniform file length [17]. The effect of this reduction in the general case is numerically evaluated later in the section.

The simplification of the general constraints in (9) follows similar reasoning used to obtain the constraints (11)-(12) in Section III, except now there are arbitrary file lengths F_l and popularities p_l; this gives

$$\sum_{j=0}^{K} \binom{K}{j} v_{l,j} = F_l, \forall l \in [N] \qquad (31)$$

as the file reconstruction constraints, and

$$\sum_{j=1}^{K} \binom{K-1}{j-1} v_{l,j} \leq \mu_l, \forall l \in [N] \qquad (32)$$

for the cache constraint. The other two constraints,

$$v_{l,j} \geq 0, \forall l \in [N], \forall j \in [0 : K], \qquad (33)$$

and

$$\sum_{l=1}^{N} \mu_l \leq M \qquad (34)$$

have more obvious modifications.

To express the objective function in polynomial number of terms, we now need to impose certain *memory inequality* conditions in order to simplify the max operator in the objective. We propose two different approaches called the *popularity-first* approach and the *length-first* approach respectively for handling the non-uniform file popularity and file length.

1) Popularity-First Approach: In the popularity-first approach, files are labelled in decreasing over of popularity, i.e. such that $p_1 \geq \cdots \geq p_N$. Then, motivated by the idea that more popular files ought to have more cache space dedicated to them, a memory inequality condition is imposed on the cache content as

$$v_{l_1,j} \geq v_{l_2,j}, \forall l_1, l_2 \in [N] \text{ s.t. } l_1 < l_2, j \in [K]. \quad (35)$$

This memory inequality constraint is adopted to help reduce the complexity of the problem; as previously discussed, this constraint (and others like it later in the paper) allow the *max* function in (8) to be eliminated in favour of a linear function of the variables, which in turn allows for the expected rate to be computed in a polynomial number of operations, instead of the exponential number required by (8).

The popularity-first approach is most appropriate for the special case of non-uniform file popularity alone, but with uniform file length. The following proposition shows explicitly the effect that (35) has on the objective function in this case. Note that in this special case of uniform file length, (35), which holds for $j = 1, \ldots, K$, becomes reversed for $j = 0$. To see this, consider two files l_1 and l_2 with $l_1 < l_2$, that satisfy (35); if both have length F, i.e. satisfy (31) with $F_{l_1} = F_{l_2} = F$, then $v_{l_1,0} \leq v_{l_2,0}$ because every other subfile of l_1 is larger than every other subfile of l_2.

As mentioned earlier, this special case of non-uniform file popularity alone with uniform file length has already been considered in independent work [17]. But the problem formulation of [17] does not account for the difference in the lengths of subfiles within the max operator, thus may result in loss of optimality. The expression below is an exact accounting of the expected delivery rate.

Proposition 1. Consider the case of non-uniform file popularity and uniform file length. Let the variables defined in (30) be subject to condition (35) with files labelled in decreasing order of popularity. Then the objective function (8) simplifies exactly as

$$\mathbb{E}\left[\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \backslash \emptyset} \max_{k \in \mathcal{S}} \{|W_{\mathcal{S}\backslash\{k\}}^{(d_k)}|\}\right] =$$
$$\sum_{j=1}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \binom{K-1-i}{j} \Pr[Y_i = l]v_{l,j}$$
$$+ \sum_{i=0}^{K-1} \sum_{l=1}^{N} \Pr[Y_{K-i-1} = l]v_{l,0}, \quad (36)$$

where Y_i is the random variable representing the $(i+1)$-th smallest index in a random request vector **d**.

The proof of Proposition 1 is contained in Appendix B. Note that the probabilities $\Pr[Y_i = l]$ can be obtained directly from Lemma 2: since the files are labelled in decreasing order of popularity, the probability that l is the $(i+1)$-th smallest file requested in **d** is equivalent to the probability that l is the $(i+1)$-th smallest index in **Z**. The optimization problem for

the case of non-uniform file popularity, uniform file length, and uniform cache size can now be written as

$$\text{minimize} \quad \sum_{j=1}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \binom{K-1-i}{j} \Pr[Y_i = l]v_{l,j}$$
$$+ \sum_{i=0}^{K-1} \sum_{l=1}^{N} \Pr[Y_{K-i-1} = l]v_{l,0} \quad (37)$$
$$\text{subject to} \quad (31)\text{-}(35), \quad (38)$$

Note that this is a linear program, with a number of summands in the objective function that scales K^2N, and exactly KN variables and $N(K+3) + K(N(N-1))/2 + 1$ constraints.

If the files are also of non-uniform length, further work is required, but a similar optimization problem can nonetheless be developed. The details are omitted here both for the sake of brevity and because numerical results suggest that the popularity-first approach does not perform as well as the length-first approach in the general case when both popularity and file lengths are non-uniform.

2) Length-First Approach: In the length-first approach, files are labelled in decreasing order of length, i.e. such that $F_1 \geq F_2 \geq \cdots \geq F_N$. Then, motivated by the idea that longer files ought to have more cache space dedicated to them, the following memory inequality condition is imposed:

$$v_{l_1,j} \geq v_{l_2,j}, \forall l_1, l_2 \in [N] \text{ s.t. } l_1 < l_2, j \in [0 : K-1]. \quad (39)$$

Using similar reasoning as Proposition 1, it is straightforward to show that

Proposition 2. Consider the case of non-uniform file length with either uniform or non-uniform popularity. Let the variables defined in (30) be subject to condition (39) with files labelled in decreasing order of length. Then the objective function (8) simplifies exactly as

$$\mathbb{E}\left[\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \backslash \emptyset} \max_{k \in \mathcal{S}} \{|W_{\mathcal{S}\backslash\{k\}}^{(d_k)}|\}\right]$$
$$= \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \binom{K-1-i}{j} \Pr[Y_i = l]v_{l,j} \quad (40)$$

where Y_i is the random variable representing the $(i+1)$-th smallest index in a random request vector **d**.

The length-first optimization problem for non-uniform file popularity, non-uniform file length, and uniform cache size is obtained as

$$\text{minimize} \quad \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \binom{K-1-i}{j} \Pr[Y_i = l]v_{l,j} \quad (41)$$
$$\text{subject to} \quad (31)\text{-}(34), (39) \quad (42)$$

which is a linear program with K^2N summands in the objective function, KN variables, and $N(K+3) + K(N(N-1))/2 + 1$ constraints.
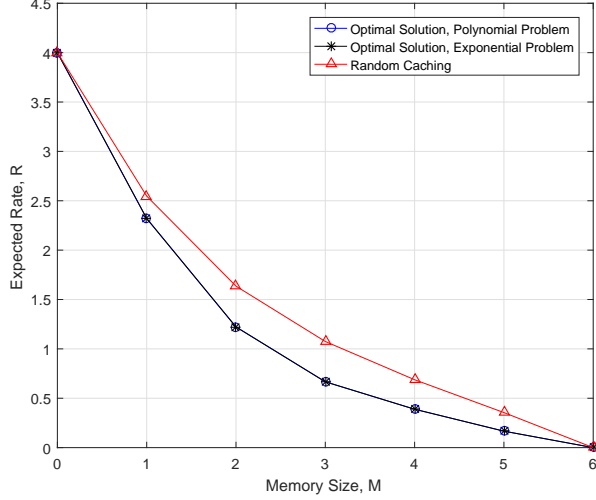
Fig. 1. A comparison of the performance of the solution obtained from (8)-(9) to the solution obtained by (37)-(38), with reference to a baseline random caching scheme, for the case of non-uniform file popularity only
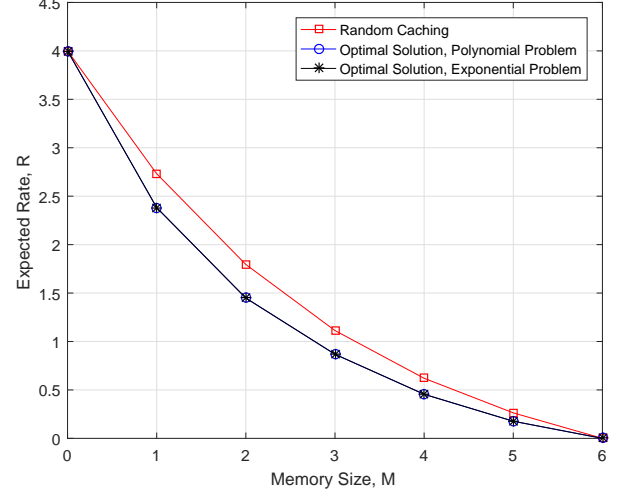

Fig. 2. A comparison of the performance of the solution obtained from (8)-(9) to the solution obtained by (41)-(42), with reference to a baseline random caching scheme, for the case of non-uniform file length only.

D. Numerical Results

To evaluation the effect of the simplified problem formulation, we consider a case with $K = 4$ users with equal cache sizes of M, and $N = 6$ files. When the files are of uniform length, the value $F = 1$ is used, and when they are of non-uniform length, the values $\{F_1, \ldots, F_6\} = \{9/6, 8/6, 7/6, 5/6, 4/6, 3/6\}$ are used. Similarly, when the files are of uniform popularity, the value $p_l = 1/N$ is used for all $l \in [N]$, but when the files have non-uniform popularity, the distribution is given by a Zipf distribution with parameter s, which has been observed empirically to be reasonable model for user demands; a parameter of $s = 0.56$ is used in this paper (see e.g. [42]). When both the file lengths and popularities are non-uniform, the relationship between length and popularity is specified explicitly.

Fig. 1 compares the rate-memory tradeoff curve for the original problem (8)-(9) and the simplified problem (37)-(38) for the non-uniform popularity and uniform length case. A baseline random caching scheme is also included for reference. This random caching scheme is essentially the decentralized scheme of [4] but with file n allocated $\mu_n F$ bits of cache memory instead of MF/N bits; initially, the value is obtained as $\mu_n = \min\{Mp_n, 1\}$ for all $n \in [N]$, and if $\sum_{n=1}^{N} \mu_n < M$ after that, the remaining cache memory is allocated to each file sequentially until the remaining memory runs out.

Conversely, Fig. 2 compares the rate-memory tradeoff curves of (8)-(9) and the simplified problem (41)-(42) for the non-uniform length and uniform popularity case. The random caching baseline scheme used here is essentially equivalent to the one proposed in [18].

Both figures show that the performance of the general problem and the two simplified problems is identical for the respective cases considered here. Indeed, when the numer-

ical solutions of (8)-(9) for these two cases are examined explicitly, it is clear that the memory constraint conditions are indeed satisfied, and so the optimal solutions in these cases are attainable by the respective simplified problems.

Next, Fig. 3 compares the performance of the original problem (8)-(9) to both the length-first and popularity-first simplified problems for the case of non-uniform file length and popularity. The specific pairings of length and popularities used and their associated labels are listed in Table I.

Although somewhat arbitrary, these file length and popularity combinations are intended to simulate a practical scenario where file popularity and length are relatively uncorrelated. While examining this individual case is not sufficient for determining general patterns, it is enough to gain some important insight about the tension between file length and popularity.

Fig. 3 shows that in this case, the length-first scheme yields much better results than the popularity-first scheme. Indeed, the length-first scheme obtains the same performance as the original problem for all M considered except $M = 1$. The reason for this divergence can be seen from Table II which shows the optimal solution to the original problem (8)-(9) in the $M = 1$ case. The value of $|W_{\mathcal{S}}^{(l)}|$ is shown in the l-th column of the row labelled with $\mathcal{S}$, and the files are ordered

TABLE I
THE FILE LABELLING AND LENGTH-POPULARITY PAIRINGS USED IN THE NON-UNIFORM FILE POPULARITY AND LENGTH CASE.

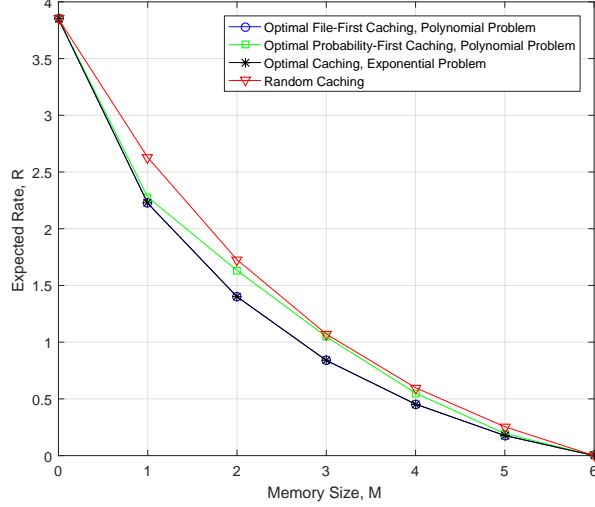| Length-First (LF) and Popularity-First (PF) Labelling			
LF File Index	PF File Index	Length	Popularity
1	5	9/6	0.1176
2	3	8/6	0.1566
3	2	7/6	0.1965
4	6	5/6	0.1062
5	1	4/6	0.2897
6	4	3/6	0.1333

Fig. 3. A comparison of the performance of the solution obtained from (8)-(9) to the solutions obtained by (41)-(42) and a popularity-first optimization problem, with reference to a baseline random caching scheme, for the case of non-uniform file length and popularity.

TABLE II
OPTIMAL SUBFILE SIZES AND MEMORY ALLOCATION FOR THE GENERAL PROBLEM (8)-(9) WITH $K = 4$, $N = 6$, $M = 1$, AND FILE LENGTHS AND POPULARITIES GIVEN IN TABLE I (USING LENGTH-FIRST INDEXING), WITH VALUES ROUNDED TO THREE DECIMAL PLACES.

Subset	File Index					
	1	2	3	4	5	6
$\emptyset$	0.833	0.583	0.417	0.167	0	0
$\{1\}$	0.167	0.188	0.188	0.167	0.167	0.125
$\{2\}$	0.167	0.188	0.188	0.167	0.167	0.125
$\{3\}$	0.167	0.188	0.188	0.167	0.167	0.125
$\{4\}$	0.167	0.188	0.188	0.167	0.167	0.125
$\{1,2\}$	0	0	0	0	0	0
$\cdots$			$\cdots$			
$\{1,2,3,4\}$	0	0	0	0	0	0
Total memory:	0.167	0.188	0.188	0.167	0.167	0.125

using the length-first labelling of Table I. It is clear that file 1, the largest file but fifth-most popular, has been allocated less cache memory than files 2 and 3, which are the third and second most popular files respectively. Thus the memory-inequality constraint (39) is violated and so the length-first simplified problem cannot attain the optimal solution of (8)-(9). Despite this, the optimal value to the simplified problem is only about 10^{-4} larger than the optimal value of (8)-(9), and so the difference is not significant. It would thus appear that, while probability cannot be completely ignored in theory, a length-first approach to caching can yield very good results in practice. This insight is later used in Section VI-A, but first the problem of non-uniform cache size must be studied on its own first; this is done next.

V. CODED CACHING WITH HETEROGENEOUS CACHE SIZES

We next consider simplifying the optimization formulation for the case with non-uniform cache sizes. For now, file popularity and file length are kept uniform; the case with all parameters being non-uniform is treated in the subsequent section. For the case of non-uniform cache size, a decentralized coded caching scheme is developed in [24] and subsequently improved upon in the $K > N$ case by [25], [26]. As previously discussed, [27] uses an optimization framework similar to the one used in this paper to generate a scheme for the centralized case. For the sake of completion, we note again the work [19] in which the rate region for both non-uniform cache and file size is characterized for $K = 2$ users and $N = 2$ files, but the optimal scheme is not yet known for the general case.

A. Optimization Formulation

We first consider a simple case where there are only two cache sizes, "large" and "small", represented by M_L and M_S respectively. The variable K_L is used to represent the number of users with large caches, and K_S is used to represent the number of users with small caches, such that $K_L + K_S = K$. Note that file lengths and popularities are fixed as uniform, i.e. $p_1 = \cdots = p_N = 1/N$ and $F_1 = \cdots = F_N = F = 1$ respectively.

Since files here are equally popular and of the same size, we have symmetry across files, but now the symmetry across users is broken by the non-uniform cache size. However, certain user symmetry still exists, i.e., symmetry among members of the same cache size group. We therefore define three sets of variables for $j \in [0 : K]$, denoted $v_{j,S}, v_{j,L}$, and $v_{j,M}$, as follows. For a subset of users $\mathcal{S}$ with a size $|\mathcal{S}| = j$,

$$|W_S^{(l)}| = \begin{cases} v_{j,S} & \text{if } \mathcal{S} \text{ contains only small-cache users,} \\ v_{j,L} & \text{if } \mathcal{S} \text{ contains only large-cache users,} \\ v_{j,M} & \text{otherwise,} \end{cases}$$
(43)

for all files $l \in [N]$.

This definition suggests some natural constraints. First, since there are only K_S small-cache users, there cannot be a group of j small-cache users if $j > K_S$, so we set

$$v_{j,S} = 0, \forall j > K_S. \tag{44}$$

Similarly, for large-cache users,

$$v_{j,L} = 0, \forall j > K_L. \tag{45}$$

A similar constraint is required for the $v_{1,M}$. Since there cannot be a subset of size one with both large- and small-cache users, we require

$$v_{1,M} = 0; \tag{46}$$

Moreover, since the $j = 0$ variables correspond to subsets of size 0, it it not particularly meaningful to discuss whether or note this corresponds to a small, large, or mixed subset. To avoid any further complications, we simply set

$$v_{0,S} = v_{0,L} = v_{0,M} = v_0, \tag{47}$$

To understand the file reconstruction condition, note that, for $j \geq 2$, there are $\binom{K_S}{j}$ groups of small-cache users, $\binom{K_L}{j}$ groups of large-cache users, and

$$\sum_{i=1}^{j-1} \binom{K_S}{i}\binom{K_L}{j-i}$$

mixed groups, because each mixed group must have at least one small-cache users and at least one-large cache user. By adding and subtracting the $i = 0$ and $i = j$ terms and using Lemma 1, we can rewrite this as

$$\sum_{i=0}^{j} \binom{K_S}{i}\binom{K_L}{j-i} - \binom{K_L}{j} - \binom{K_S}{j}$$
$$= \binom{K}{j} - \binom{K_L}{j} - \binom{K_S}{j},$$

and so the total portion of the file cached by subsets of size $j \geq 2$ is

$$\binom{K_L}{j}(v_{j,L} - v_{j,M}) + \binom{K_S}{j}(v_{j,S} - v_{j,M}) + \binom{K}{j}v_{j,M}. \tag{48}$$

For $j = 1$, there are $K_L = \binom{K_L}{1}$ large users and $K_S = \binom{K_S}{1}$ small users. Moreover, since $v_{1,M} = 0$, it is easy to see that (48) also holds for $j = 1$. Finally, consider the $j = 0$ case. Here we only need to add v_0 to capture the portion of the file not cached by any user. However, note that if we set $j = 0$ in (48) and apply constraint (47), the first two terms become 0, while the last term reduces to $v_{0,M} = v_0$. Thus (48) applies for all j, and so we can conveniently express the file reconstruction constraint as

$$\sum_{j=0}^{K} \binom{K_L}{j}(v_{j,L} - v_{j,M}) + \binom{K_S}{j}(v_{j,S} - v_{j,M})$$
$$+ \binom{K}{j}v_{j,M} = 1. \tag{49}$$

Similar reasoning is used to obtain the cache memory constraints. A small cache user caches every subfile labelled with a subset in which he is contained as a member, and so necessarily caches only subfiles of size $v_{j,S}$ and $v_{j,M}$. Specifically, a small-cache user is a member of $\binom{K_S-1}{j-1}$ small groups of size j, and

$$\sum_{i=0}^{K-2} \binom{K_S - 1}{i}\binom{K_L}{j-1-i} \tag{50}$$

mixed groups. Using Lemma 1 once again, the cache memory constraint for the small user is obtained as

$$\sum_{j=1}^{K} \binom{K_S - 1}{j-1}(v_{j,S} - v_{j,M}) + \binom{K-1}{j-1}v_{j,M} = M_S/N. \tag{51}$$

Note that the $j = 1$ expression reduces to $\binom{K_L-1}{0}(v_{1,S}-0) + 0 = v_{1,S}$, as desired. Note also that each file is allocated an equal M_S/N of the cache because the files are equally sized

and equally popular. Using similar reasoning for large-cache users, we obtain

$$\sum_{j=1}^{K} \binom{K_L - 1}{j-1}(v_{j,L} - v_{j,M}) + \binom{K-1}{j-1}v_{j,M} = M_L/N. \tag{52}$$

As always, it is required that all subfiles be of nonnegative size:

$$v_{j,S} \geq 0, v_{j,L} \geq 0, v_{j,M} \geq 0, \forall j \in \{0, \ldots, K\}. \tag{53}$$

Finally, the memory inequality constraints for this problem are introduced. In general, we would expect the subfiles that large users have cached to be longer than the ones cached by small users. This is codified in the problem explicitly with

$$v_{j,L} \geq v_{j,M}, \ j \in [2:K_L] \tag{54}$$
$$v_{j,M} \geq v_{j,S}, \ j \in [2:K_S] \tag{55}$$
$$v_{j,L} \geq v_{j,S}, \ j \in [1:K_L]. \tag{56}$$

Again note that there is some redundancy in these constraints, but they are nonetheless included for clarity of exposition. Note also that the first two inequalities hold from $j = 2$ to $j = K_L$ and $j = K_S$ respectively; the $j = 1$ is already constrained by (46), while the $j = 0$ is constrained by (47), and the $j > K_L$ and $j > K_S$ cases are governed by (45) and (44) respectively.

As Proposition 3 shows, the memory inequality constraints allow us to greatly simplify the original objective function (8).

Proposition 3. Assuming uniform file length and popularity but two different user cache sizes, and with variables as defined in (43) and satisfying (44)-(47) and (53)-(56), the objective function (8) simplifies exactly as

$$\mathbb{E}\left[\sum_{S \in \mathcal{P}(\mathcal{U})\backslash\emptyset} \max_{k \in S}\{|W_{S\backslash\{k\}}^{(d_k)}|\}\right]$$
$$= \sum_{j=0}^{K-1} \binom{K_s}{j+1}(v_{j,S} - v_{j,M}) + \binom{K}{j+1}v_{j,M}+$$
$$\left(\binom{K_L}{j+1} + \binom{K_S}{1}\binom{K_L}{j}\right)(v_{j,L} - v_{j,M}) \tag{57}$$

The proof of Proposition 3 is in Appendix C. The simplified optimization problem can then be written as

$$\text{minimize} \sum_{j=0}^{K-1} \binom{K_s}{j+1}(v_{j,S} - v_{j,M}) + \binom{K}{j+1}v_{j,M}+$$
$$\left(\binom{K_L}{j+1} + \binom{K_S}{1}\binom{K_L}{j}\right)(v_{j,L} - v_{j,M}) \tag{58}$$

subject to (44)-(47), (49), (51)-(56). $\tag{59}$

This is a linear program that has a number of variables, constraints, and objective function summands that scale linearly in K.
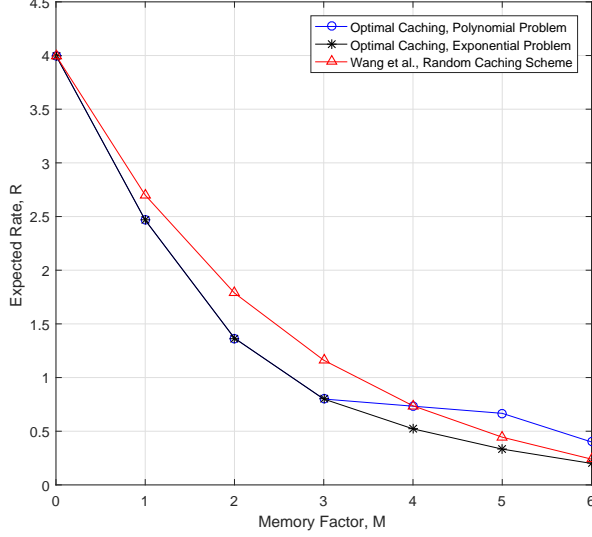
Fig. 4. A comparison of the performance of the solution obtained from (8)-(9) to the solution obtained by (58)-(59) for $K_S = 2$.

TABLE III
OPTIMAL SUBFILE SIZES AND MEMORY ALLOCATION FOR THE GENERAL PROBLEM (8)-(9) WITH $K = 4$, $N = 6$, $K_S = 2$, $M_S = 3.2$ AND $M_L = 4.8$, WITH VALUES ROUNDED TO THREE DECIMAL PLACES.

Subset	File Index					
	1	2	3	4	5	6
$\emptyset$	0	0	0	0	0	0
. . .			. . .			
$\{4\}$	0	0	0	0	0	0
$\{1,2\}$	0.056	0.056	0.056	0.056	0.056	0.056
$\{1,3\}$	0.056	0.056	0.056	0.056	0.056	0.056
$\{1,4\}$	0.056	0.056	0.056	0.056	0.056	0.056
$\{2,3\}$	0.056	0.056	0.056	0.056	0.056	0.056
$\{2,4\}$	0.056	0.056	0.056	0.056	0.056	0.056
$\{3,4\}$	0.056	0.056	0.056	0.056	0.056	0.056
$\{1,2,3\}$	0.033	0.033	0.033	0.033	0.033	0.033
$\{1,2,4\}$	0.033	0.033	0.033	0.033	0.033	0.033
$\{1,3,4\}$	0.300	0.300	0.300	0.300	0.300	0.300
$\{2,3,4\}$	0.300	0.300	0.300	0.300	0.300	0.300
$\{1,2,3,4\}$	0	0	0	0	0	0
Mem. (L):	0.800	0.800	0.800	0.800	0.800	0.800
Mem. (S):	0.533	0.533	0.533	0.533	0.533	0.533

Finally, we remark that only two cache sizes were considered in this paper. While a practical system would likely have more than two cache sizes, it is reasonable to expect that only a small number of cache sizes will be used, e.g. cell phones with 8, 16, 32 or 64 GB of cache memory. The reasoning used here for two cache sizes could then be extended to accommodate these additional cache sizes as needed.

B. Numerical Results

Consider a case where there are $N = 6$ files, uniform in popularity and length, and $K = 4$ users. Define a "memory factor" $M \in [0 : N]$; there are K_S small users with a cache size of $M_S = 0.8M$, and K_L large users with a cache size of $M_L = 1.2M$. Fig. 4 compares the corresponding solution of the general problem (8)-(9) to the solution of the simplified problem (58)-(59) when there are $K_S = 2$ small users. The random caching scheme of [24] is included, but the centralized scheme of [27] (which has exponential complexity) is not. The purpose here is not to determine the best caching scheme for the heterogeneous cache case, but to, first, demonstrate the implicit performance-tractability tradeoff of using the simplified problem (58)-(59) over the general problem, and second, to demonstrate that it is worth the effort of developing and using these problems to design cache content (rather than caching randomly) when the engineering context allows for it. Nevertheless, we expect the performance of the problem in [27] to be very similar, if not identical, to the exponential problem developed here, even though the two optimization frameworks are not identical themselves. Table III also shows the optimal cache content obtained from the general problem in the $K_S = 2$, $M = 4$ case.

Fig. 4 shows that in the $K_S = 2$ case, while the simplified problem tracks the optimal scheme for small cache size

$(M \leq 3)$, it performs worse than even the random caching scheme for large M values. Table III reveals why this is the case. Here, users 1 and 2 are the small users, and users 3 and 4 are the large users. The variable definitions in (43) specify one variable for all mixed subsets of the same size, but consider the subfile sizes for the size-three subsets: the subsets with 2 small users have smaller subfiles than the subsets with 2 large users. Thus using only one $v_{l,j}^M$ variable for these four subsets results in a loss in performance. In principle, one could introduce more variables to accommodate this, albeit at a cost of a more complicated objective function.

VI. CODED CACHING WITH HETEROGENEOUS FILES AND CACHE SIZES

The natural final step in this program is to develop a tractable optimization problem that accommodates non-uniformity in cache size, file size, and popularity at the same time. To the best of our knowledge, there has yet to be a caching scheme proposed that handles heterogeneity in all three of these domains.

A. Optimization Formulation

We begin by defining a new set of variables that, in a sense, combines the functionality of the variables defined in (30) and (43). Let

$$|W_S^{(l)}| = \begin{cases} v_{l,j}^S & \text{if } \mathcal{S} \text{ contains only small-cache users,} \\ v_{l,j}^L & \text{if } \mathcal{S} \text{ contains only large-cache users,} \\ v_{l,j}^M & \text{otherwise,} \end{cases}$$

(60)

for a subset of users $\mathcal{S}$ such that $|\mathcal{S}| = j$ and all files $l \in [N]$. The symmetry across users is broken by the heterogeneity of the user cache size, and the symmetry across files is broken by a the heterogeneity of files in both length and popularity. Nevertheless, we can still exploit the symmetry across subsets

of users of the same size (j) and type (i.e. small, large, mixed) for a particular file (l) to reduce the complexity of the original problem (8)-(9) while still accounting for the aforementioned heterogeneity.

Many of the constraints used in the non-uniform cache size case can be converted to their equivalents for this new case. If $j > K_S$, there cannot be a subset of j small cache users, so

$$v_{l,j}^S = 0, \forall j > K_S, \forall l \in [N], \tag{61}$$

and similarly

$$v_{l,j}^L = 0, \forall j > K_L, \forall l \in [N]. \tag{62}$$

Since there cannot be a subset of size 1 containing both large and small users, the $l = 1$ variable is constrained as

$$v_{l,1}^M = 0, \forall l \in [N], \tag{63}$$

and for convenience, we set

$$v_{l,0}^S = v_{l,0}^L = v_{l,0}^M = v_{l,0}, \forall l \in [N]. \tag{64}$$

The cache size-based memory inequalities should still hold, giving, for a fixed file l,

$$v_{l,j}^L \geq v_{l,j}^M, \ j \in [2 : K_L], \forall l \in [N], \tag{65}$$
$$v_{l,j}^M \geq v_{l,j}^S, \ j \in [2 : K_S], \forall l \in [N], \tag{66}$$
$$v_{l,j}^L \geq v_{l,j}^S, \ j \in [1 : K_L], \forall l \in [N]. \tag{67}$$

We also import conditions from the non-uniform file size and popularity problem. As seen earlier, it is better to prioritize file length rather than popularity, and so we label the files in decreasing order of file length. This gives

$$v_{l_1,j}^L \geq v_{l_2,j}^L, \ j \in [0 : K - 1], \forall l_1, l_2 \text{ s.t. } l_1 < l_2 \tag{68}$$
$$v_{l_1,j}^M \geq v_{l_2,j}^M, \ j \in [0 : K - 1], \forall l_1, l_2 \text{ s.t. } l_1 < l_2 \tag{69}$$
$$v_{l_1,j}^S \geq v_{l_2,j}^S, \ j \in [0 : K - 1], \forall l_1, l_2 \text{ s.t. } l_1 < l_2. \tag{70}$$

The remaining conditions are formed using identical reasoning to the non-uniform cache case, but occur on a file-by-file basis as needed. The file reconstruction constraint remains the same, with the minor change that the file must add up not to the common file length 1, but to F_l, the actual length of the file as expressed below:

$$F_l = \sum_{j=0}^K \binom{K_L}{j}(v_{l,j}^L - v_{l,j}^M) + \binom{K_S}{j}(v_{l,j}^S - v_{l,j}^M)$$
$$+ \binom{K}{j}v_{l,j}^M, \qquad \forall l \in [N] \tag{71}$$

The cache memory constraints are modified similarly, except instead of giving an equal amount M_S/N to each file, an amount μ_l^S is allocated to file l, yielding

$$\mu_l^S = \sum_{j=1}^K \binom{K_S - 1}{j - 1}(v_{l,j}^S - v_{l,j}^M)$$
$$+ \binom{K - 1}{j - 1}v_{l,j}^M, \qquad \forall l \in [N], \tag{72}$$

where it must be the case that

$$\sum_{l=1}^N \mu_l^S = M_S. \tag{73}$$

A similar pair of equations holds for large users:

$$\mu_l^L = \sum_{j=1}^K \binom{K_L - 1}{j - 1}(v_{l,j}^L - v_{l,j}^M)$$
$$+ \binom{K - 1}{j - 1}v_{l,j}^M, \qquad \forall l \in [N], \tag{74}$$

and

$$\sum_{l=1}^N \mu_l^L = M_L. \tag{75}$$

The subfiles are also required to be positive in size, as always:

$$v_{l,j}^L \geq 0, \ v_{l,j}^S \geq 0, \ v_{l,j}^M \geq 0, \ \forall j \in [0 : K], \ l \in [N] \tag{76}$$

Finally, another set of memory inequality constraints are required to break ties between small-index, small-cache subfiles and large-index, large-cache subfiles. Leaving the justification and discussion of this choice to later sections, we develop a caching scheme under the constraints

$$v_{l_1,j}^L \geq v_{l_2,j}^M, \ j \in [2 : K_L], \ \forall l_1, l_2 \in [N] \tag{77}$$
$$v_{l_1,j}^M \geq v_{l_2,j}^S, \ j \in [2 : K_S], \ \forall l_1, l_2, \in [N] \tag{78}$$
$$v_{l_1,j}^L \geq v_{l_2,j}^S, \ j \in [1 : K_L], \ \forall l_1, l_2 \in [N]. \tag{79}$$

In words, this means that subfiles for any file stored on a larger cache type should be larger than the subfiles of *any* file stored on a smaller cache type, independent of which files are involved. The following proposition gives the objective function of the simplified optimization problem.

Proposition 4. Define the following functions of the integer parameters n, m, j and i:

$$\nu_1(n, m, j, i) = \left(\frac{K_S - m}{K - n + 1}\right)$$
$$\binom{K_S - m - 1}{i}\binom{K_L - n + 1 + m}{j - i},$$

$$\nu_2(n, m, j, i) = \left(\frac{K_L - n + 1 + m}{K - n + 1}\right)$$
$$\binom{K_S - m}{i}\binom{K_L - n + m}{j - i},$$

and

$$\nu(n, j) = \sum_{m=0}^{n-1} \frac{\binom{K_S}{m}\binom{K_L}{n-1-m}}{\binom{K}{n-1}}$$
$$\left(\sum_{i=1}^{j-2} \nu_1(n, m, j, i) + \sum_{i=2}^{j-1} \nu_2(n, m, j, i)\right)$$

Then for the variables defined in (60) satisfying (61)-(79), the objective function (8) simplifies exactly as

$$
\mathbb{E}\left[\sum_{\mathcal{S}\in\mathcal{P}(\mathcal{U})\setminus\emptyset}\max_{k\in\mathcal{S}}\{|W_{\mathcal{S}\setminus\{k\}}^{(d_k)}|\}\right]
$$

$$
= \sum_{i=0}^{K-1}\sum_{l=1}^{N}\Pr[Y_i = l]v_{l,0}^{M}
$$

$$
+ \sum_{j=1}^{K_L-1}\sum_{i=0}^{K_L-1}\sum_{l=1}^{N}\binom{K_L-1-i}{j}\Pr[Y_i^L = l]v_{l,j}^{L}
$$

$$
+ \sum_{j=1}^{K_S-1}\sum_{i=0}^{K_S-1}\sum_{l=1}^{N}\binom{K_S-1-i}{j}\Pr[Y_i^S = l]v_{l,j}^{S}
$$

$$
+ \sum_{j=1}^{K_L}\sum_{i=0}^{K_S-1}\sum_{l=1}^{N}\binom{K_L}{j}\Pr[Y_i^S = l]v_{l,j}^{L}
$$

$$
+ \sum_{j=2}^{K_S-1}\sum_{i=0}^{K_S-1}\sum_{l=1}^{N}\binom{K_S-1-i}{j}\binom{K_L}{1}\Pr[Y_i^S = l]v_{l,j}^{M}
$$

$$
+ \sum_{j=\max\{K_S,K_L\}+1}^{K-1}\sum_{i=0}^{K-1}\sum_{l=1}^{N}\binom{K-1-i}{j}\Pr[Y_i = l]v_{l,j}^{M}
$$

$$
+ \sum_{j=3}^{\max\{K_S,K_L\}}\sum_{n=1}^{K}\sum_{l=1}^{N}\Pr[Y_{n-1} = l]\nu(n,j)v_{l,j}^{M} \qquad (80)
$$

Proposition 4 is proved in Appendix D. Although visually complicated, (80) simplifies the original objective function (8) by using only $3(K+1)N$ variables and having a number of terms that scales with K^2N rather than $(2N)^K$. The number of constraints described by (61)-(79) scales with KN^2, and so the following optimization problem is a tractable method of obtaining a caching scheme that accommodates heterogeneity in cache size, file size, and file popularity:

$$
\text{minimize} \quad (80) \qquad (81)
$$
$$
\text{subject to} \quad (61)-(79) \qquad (82)
$$

B. Numerical Results

To demonstrate that this simplified optimization problem performs well when compared to the general problem (8)-(9), we again consider $K = 4$ users ($K_S = 2$ of which are small-cache users) and $N = 6$ files, with all of the non-uniformities used thus far: file labels and popularity/size pairs as given in Table I, and cache sizes of $M_S = 0.8M$ and $M_L = 1.2M$ for $M \in [0 : N]$. Fig. 5 compares the simplified and general problems with a naive random caching baseline.

We see that the simplified problem yields a scheme that closes mirrors, although does not match exactly, the performance of the scheme obtained from the general problem for small and intermediate M values. Table IV shows the optimal solution to the general problem for the $M = 2$ case; we see several violations of the memory inequality constraints that explain why the simplified problem could not achieve as good a performance as the general problem: the "true" optimal solution lies outside of its feasible space. Nevertheless, the
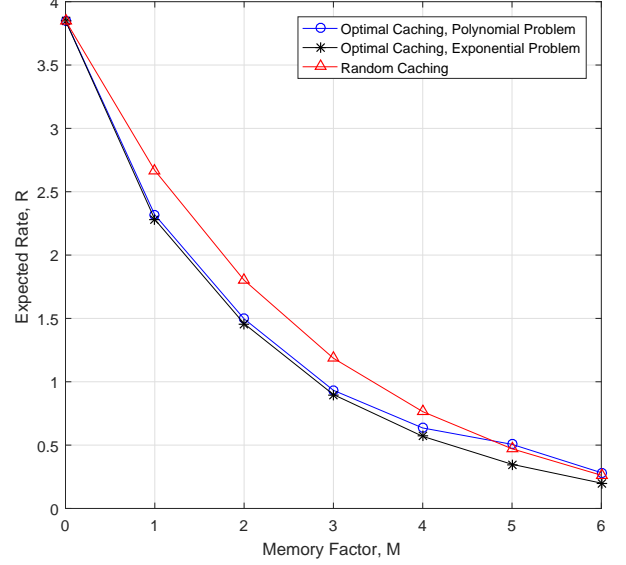

Fig. 5. A comparison of the performance of the solution obtained from (8)-(9) to the solution obtained by (81)-(82), with reference to a baseline random caching scheme, for the case of non-uniform file size, file popularity, and cache size.

simplified problem still achieves good performance compared to the general problem in this regime. For large M, we see that the expected rate of the simplified scheme does not drop as quickly as the general problem solution, and is even eclipsed by the random caching scheme. This occurs for the same reason we saw in Section V in the $K_S = 2$ case (Fig. 4). To compare the relative performance of the three schemes in general, Fig. 6 shows the percent increase in expected rate if the random caching scheme is used over the general and simplified schemes respectively. The significant increase in rate when using random caching make it clear that designing the cache content can be worthwhile when the engineering context allows for it.

C. Further Extensions

We first echo the earlier comments about heterogeneous cache sizes: we consider only two different cache sizes here, but it is possible to use the same reasoning to develop a tractable optimization problem for a practical system having more (but not many more) cache sizes.

The primary focus in this section, however, is on the memory inequality constraints (77)-(79) used in developing the simplified problem (81)-(82) of this section. Recall that these constraints require, among other things, that (roughly speaking) for a fixed index j, the $v_{l,j}^L$ variables for *all* files l be larger than the $v_{l,j}^S$ variables for all files. Thus the large-user subfile for the smallest file is larger than the small-user subfile for the largest file. This restriction was required to allow us to write the simplified problem, and numerical results show that the simplified problem still performed well compared to the general problem for the considered

TABLE IV
OPTIMAL SUBFILE SIZES AND MEMORY ALLOCATION FOR THE GENERAL
PROBLEM (8)-(9) WITH $K = 4$, $N = 6$, $K_S = 2$, $M_S = 1.6$, $M_L = 2.4$
AND FILE POPULARITY/SIZE PAIRS GIVEN BY TABLE I, WITH VALUES
ROUNDED TO THREE DECIMAL PLACES.

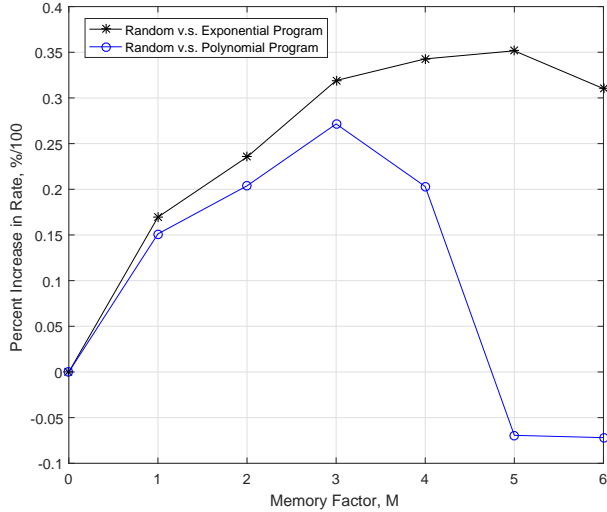| Subset	File Index					
	1	2	3	4	5	6
$\emptyset$	0	0	0	0	0	0
$\{1\}$	0.178	0.178	0.178	0.178	0.165	0.085
$\{2\}$	0.178	0.178	0.178	0.178	0.165	0.085
$\{3\}$	0.178	0.178	0.178	0.178	0.165	0.085
$\{4\}$	0.178	0.178	0.178	0.178	0.165	0.085
$\{1, 2\}$	0	0.077	0.008	0	0	0
$\{1, 3\}$	0.090	0.090	0.090	0.008	0	0
$\{1, 4\}$	0.090	0.090	0.090	0.008	0	0
$\{2, 3\}$	0.090	0.090	0.090	0.008	0	0
$\{2, 4\}$	0.090	0.090	0.090	0.008	0	0
$\{3, 4\}$	0.431	0.188	0.090	0.090	0.008	0
$\{1, 2, 3\}$	0	0	0	0	0	0
...			...			
$\{1, 2, 3, 4\}$	0	0	0	0	0	0
Mem. (L):	0.788	0.554	0.446	0.284	0.173	0.165
Mem. (S):	0.357	0.434	0.365	0.194	0.165	0.085

Fig. 6. The percent increase in expected rate due to random cache content when compared to the simplified problem (81)-(82) and general problem (8)-(9) optimal solutions.

parameters. While a full investigation of the performance of the simplified problem across all parameter values would be labourious, it is still possible to estimate the behaviour for certain parameter regimes. We should expect (77)-(79) to result in a good simplification when the disparity between the large and small cache sizes is big, and when there are small numbers of files, because the large cache users will likely store much larger subfiles than the small cache users, irrespective of the length of the file. Conversely, we should expect the performance of (81)-(82) to be relatively poor when the cache sizes are comparable and there are large numbers of files. In this case, it may make more sense to use something like the "opposite" memory inequality constraint: a small-user subfile variable $v_{l_1,j}^S$ should be larger than any

larger-user subfile variable $v_{l_2,j}^L$ if file l_1 is larger than file l_2. While the simplified optimization problem that would result from this constraint is not explored in this paper, it should be possible to construct such a problem using the same kind of reasoning used here.

Indeed, there may also be other memory inequality constraints that prove to yield useful simplified optimization problems for other parameter sets. The appeal of the tractability of these models is that a server, knowing the relevant parameters for its system, could easily compute the performance of these schemes and choose the best among them; any discussion of the specifics of such schemes, however, is left to future work.

VII. SUMMARY AND CONCLUSIONS

The two primary goals of this paper are to advance a certain optimization theoretic approach to coded caching problems, and to use that framework to derive both specific caching schemes and general insight for system models containing multiple heterogeneities that have yet to be considered in the literature. An exponentially-scaling optimization problem corresponding to a caching scheme capable of handling non-uniform file size, popularity, and cache size is developed. It is shown that the original scheme of Maddah-Ali and Niesen in [2], [3] is the optimal solution of that problem for the special case of uniform file length, popularity, and cache size.

Tractable problems are then developed to handle various combinations of heterogeneous system parameters. The consideration of these special cases also permitted the observation of the effects that these non-uniformities have on the optimal cache content. When considering non-uniform file popularity and size jointly, it is shown that while popularity may in general have some influence on the optimal cache allocation, file size can be a much stronger influence; indeed, very good performance is obtained in the case considered by ignoring file popularity altogether. Finally, with the insights obtained from the previously-explored special cases, we developed a tractable optimization problem corresponding to a caching scheme capable of accommodating all three of the aforementioned heterogeneities, and showed numerically that it performs well compared to the original exponentially-scaling problem.

APPENDIX

A. Proof of Lemma 2

We begin with the proof of the expression of $\Pr[Y_0 = i]$, for which we use induction on i for $i = 1, \dots, N$. We begin first with the $i = 1$ case. Let $\mathbf{Z} \in [N]^K$ denote the sequence of outcomes from the N trials, e.g. if for $K = 3$ and $N = 4$, trial 1 obtains outcome 2, trial 2 obtains outcome 4, and trial 3 obtains outcome 1, we have $\mathbf{Z} = [2, 4, 1]^T$. Let X_n denote the random variable representing the number of times the outcome n occurs in the K trials (i.e. the number of times it appears in $\mathbf{Z}$), and stack the X_n variables in a vector $\mathbf{X} \in [K]^N$; the example above would yield $\mathbf{X} = [1, 1, 0, 1]^T$.

Then the smallest element of $\mathbf{Z}$ is 1 (i.e. $Y_0 = 1$) if and only if $X_1 >= 1$; in other words, since 1 is the smallest possible outcome, if it occurs anywhere in $\mathbf{Z}$ then it is the smallest element. Thus we have

$$
\begin{aligned}
\Pr[Y_0 = 1] &= \Pr[X_1 \geq 1] = 1 - \Pr[X_1 = 0] \\
&= 1 - (1 - p_1)^K = \left(\sum_{l=1}^{N} p_l \right)^K - \left(\sum_{l=2}^{N} p_l \right)^K,
\end{aligned}
$$

which is indeed the formula (26) with $i = 1$, as desired.

For an arbitrary i such that $2 \leq i \leq N - 1$, we note that if the smallest element of $\mathbf{Z}$ is i, (i.e. $Y_0 = i$), then there cannot be any values smaller than i, and there must be at least one i in $\mathbf{Z}$; in other words:

$$
\begin{aligned}
\Pr[Y_0 = i] &= \Pr[X_i \geq 1, X_{i-1} = 0, \ldots X_1 = 0] \\
&= \Pr[X_1 = 0]\Pr[X_2 = 0 | X_1 = 0] \cdots \\
&\quad \Pr[X_{i-1} = 0 | X_{i-2} = \cdots = X_1 = 0] \\
&\quad \Pr[X_i \geq 1 | X_{i-1} = \cdots = X_1 = 0] \\
&= \Pr[X_1 = 0]\Pr[X_2 = 0 | X_1 = 0] \cdots \\
&\quad \Pr[X_{i-1} = 0 | X_{i-2} = \cdots = X_1 = 0] \\
&\quad (1 - \Pr[X_i = 0 | X_{i-1} = \cdots = X_1 = 0]).
\end{aligned}
$$
(83)

Now, comparing the expression (83) for $Y_0 = i$ to the same expression with $Y_0 = i - 1$, it is easy to show that

$$
\begin{aligned}
\Pr[Y_0 = i] = \Pr[Y_0 = i - 1] \\
\left(\frac{\Pr[X_{i-1} = 0 | X_{i-2} = \cdots = X_1 = 0]}{1 - \Pr[X_{i-1} = 0 | X_{i-2} = \cdots = X_1 = 0]} \right) \\
(1 - \Pr[X_i = 0 | X_{i-1} = \cdots = X_1 = 0]).
\end{aligned}
$$
(84)

It is possible to compute these conditional probabilities directly:

$$
\Pr[X_j = 0 | X_{j-1} = \cdots = X_1 = 0] = \left(1 - p_j^{(j)} \right)^K, \quad (85)
$$

where

$$
p_j^{(j)} = \frac{p_j}{\sum_{l=j}^{N} p_l}
$$

is the probability of outcome j occurring in a trial conditioned on the knowledge that outcomes 1 through $j - 1$ have not occurred in that trial. From the definition of $p_j^{(j)}$, we can rewrite (85) as

$$
\Pr[X_j = 0 | X_{j-1} = \cdots = X_1 = 0] = \left(\frac{\sum_{l=j+1}^{N} p_l}{\sum_{l=j}^{N} p_l} \right)^K.
$$

Evaluating this expression for $j = i$ and $j = i - 1$, we can

obtain from (84), after some mild algebraic manipulation,

$$
\begin{aligned}
\Pr[Y_0 = i] &= \Pr[Y_0 = i - 1] \left(\frac{\sum_{l=i}^{N} p_l}{\sum_{l=i-1}^{N} p_l} \right)^K \\
&\quad \frac{\left(\sum_{l=i-1}^{N} p_l \right)^K}{\left(\sum_{l=i-1}^{N} p_l \right)^K - \left(\sum_{l=i}^{N} p_l \right)^K} \\
&\quad \frac{\left(\sum_{l=i}^{N} p_l \right)^K - \left(\sum_{l=i+1}^{N} p_l \right)^K}{\left(\sum_{l=i}^{N} p_l \right)^K} \\
&= \Pr[Y_0 = i - 1] \frac{\left(\sum_{l=i}^{N} p_l \right)^K - \left(\sum_{l=i+1}^{N} p_l \right)^K}{\left(\sum_{l=i-1}^{N} p_l \right)^K - \left(\sum_{l=i}^{N} p_l \right)^K}.
\end{aligned}
$$
(86)

Now by the inductive hypothesis, $\Pr[Y_0 = i - 1]$ is precisely equal to the denominator of (86), and so we obtain

$$
\Pr[Y_0 = i] = \left(\sum_{l=i}^{N} p_l \right)^K - \left(\sum_{l=i+1}^{N} p_l \right)^K, \quad (87)
$$

as desired.

Although the $\Pr[Y_0 = N]$ formula was covered in the preceding paragraph, we discuss it in further detail here because (26) above may not appear sensible in the $i = N$ case. Note that if N is the smallest value in the multinomial vector $\mathbf{Z}$, then it must be the case that every element of $\mathbf{Z}$ is equal to N, otherwise some element not equal to N would be the smallest value. Thus we have

$$
\begin{aligned}
\Pr[Y_0 = N] &= \Pr[X_N = K, X_{N-1} = 0, \ldots, X_1 = 0] \\
&= p_N^K \\
&= \left(\sum_{l=N}^{N} p_l \right)^K - \left(\sum_{l=N+1}^{N} p_l \right)^K,
\end{aligned}
$$

which is the formula (26) with $i = N$, noting that we use the definition that $\sum_{l=a}^{b} n_l = 0$ when $a > b$.

Next, we proceed to the $m = 1$ case. Here we will directly compute $\Pr[Y_1 = i]$ by first deriving $\Pr[Y_1 = i, Y_0 = j]$, and then obtaining the desired quantity from the sum

$$
\Pr[Y_1 = i] = \sum_{j=1}^{N} \Pr[Y_1 = i, Y_0 = j] \quad (88)
$$

Clearly the second smallest element of $\mathbf{Z}$ is no smaller than the smallest element of $\mathbf{Z}$, so there are two cases to consider: $i > j$, and $i = j$. If $i > j$, we write

$$
\begin{aligned}
\Pr[Y_1 &= i, Y_0 = j] \\
&= \Pr[X_i \geq 1, X_{i-1} = 0, \ldots, X_j = 1, \\
&\quad X_{j-1} = \cdots = X_1 = 0] \\
&= \Pr[X_{i-1} = 0, \ldots, X_j = 1, X_{j-1} = \cdots = X_1 = 0] \\
&\quad - \Pr[X_i = 0, X_{i-1} = 0, \ldots, X_j = 1, \\
&\quad X_{j-1} = \cdots = X_1 = 0]
\end{aligned}
$$
(89)

To compute the difference (89), note that we can form a $(K, 4)$ sequential vector of outcomes $\tilde{\mathbf{Z}}(m, n)$, indexed by two integers m and n, from the original (K, N) sequential vector of outcomes $\mathbf{Z}$ in the following way: for a single trial, the first outcome of $\tilde{\mathbf{Z}}(m, n)$ occurs if any of the first $n-1$ outcomes of $\mathbf{Z}$ occur, and so it has the probability $\tilde{p}_1 = \sum_{l=1}^{n-1} p_j$; the second outcome of $\tilde{\mathbf{Z}}(m, n)$ occurs if the n-th outcome of $\mathbf{Z}$ occurs, and so it has a probability of $\tilde{p}_2 = p_n$; the third outcome of $\tilde{\mathbf{Z}}(m, n)$ occurs if any outcomes of $\mathbf{Z}$ from $n+1$ to m occurs, and so it has a probability of $\tilde{p}_3 = \sum_{l=n+1}^{m} p_l$; and the fourth outcome of $\tilde{\mathbf{Z}}(m, n)$ occurs if any of the last $N - m$ outcomes of $\mathbf{Z}$ occur, and so it has a probability $\tilde{p}_4 = \sum_{l=m+1}^{N} p_l$. If we define $\tilde{X}_j$ as the number of times outcome j occurred in $\tilde{\mathbf{Z}}(m, n)$, then we consequently have $\tilde{X}_1 = \sum_{l=1}^{n-1} X_l$, $\tilde{X}_2 = X_n$, $\tilde{X}_3 = \sum_{l=n+1}^{m} X_l$, and $\tilde{X}_4 = \sum_{l=m+1}^{N} X_l$. We can then rewrite[1] (89) using vectors $\tilde{\mathbf{X}}(i, j)$ and $\tilde{\mathbf{X}}(i-1, j)$ as

$$
\begin{aligned}
\Pr[Y_1 = i, Y_0 = j] \\
= \Pr[\tilde{X}_4 = K - 1, \tilde{X}_3 = 0, \tilde{X}_2 = 1, \tilde{X}_1 = 0] \\
- \Pr[\tilde{X}_4 = K - 1, \tilde{X}_3 = 0, \tilde{X}_2 = 1, \tilde{X}_1 = 0]
\end{aligned}
$$

where the first term is computed with respect to $\tilde{\mathbf{X}}(i-1, j)$, and the second with respect to $\tilde{\mathbf{X}}(i, j)$. Using the probabilities defined earlier, this gives

$$
\begin{aligned}
\Pr[Y_1 = i, Y_0 = j] \\
= K\tilde{p}_2(\tilde{p}_4)^{K-1} - K\tilde{p}_2(\tilde{p}_4)^{K-1} \quad (90) \\
= Kp_j \left(\left(\sum_{l=i}^{N} p_l \right)^{K-1} - \left(\sum_{l=i+1}^{N} p_l \right)^{K-1} \right) \quad (91)
\end{aligned}
$$

where, again, the terms in (90) are computed with respect to $\tilde{\mathbf{X}}(i-1, j)$, and $\tilde{\mathbf{X}}(i, j)$ respectively.

Similar reasoning yields the value of the joint probability when $i = j$:

$$
\begin{aligned}
\Pr[Y_1 = i, Y_2 = i] \\
= \Pr[X_i \geq 2, X_{i-1} = \cdots = X_1 = 0] \\
= \Pr[X_{i-1} = \cdots = X_1 = 0] - \Pr[X_i = 0 = \cdots = X_1 = 0] \\
- \Pr[X_i = 1, X_{i-1} = \cdots = X_1 = 0] \\
= \Pr[\tilde{X}_4 = K, \tilde{X}_3 = \tilde{X}_2 = \tilde{X}_1 = 0] \\
- \Pr[\tilde{X}_4 = K, \tilde{X}_3 = \tilde{X}_2 = \tilde{X}_1 = 0] \\
- \Pr[\tilde{X}_4 = K - 1, \tilde{X}_3 = 1, \tilde{X}_2 = \tilde{X}_1 = 0]. \quad (92)
\end{aligned}
$$

Here, the first term in (92) is computed with respect to $\tilde{\mathbf{X}}(i-1, i-2)$, the second with respect to $\tilde{\mathbf{X}}(i, i-1)$, and the third with respect to $\tilde{\mathbf{X}}(i, i-1)$, although this choice of $\tilde{\mathbf{X}}$ variables

---

[1]Note that the $\tilde{X}$ variables lose the (m, n) indices of the original variable $\tilde{\mathbf{X}}(m, n)$. This is done for notational convenience, but will result in an abuse of the notation when multiple $\tilde{\mathbf{X}}(m, n)$ are involved. We will therefore be careful to indicate which $\tilde{X}$ variables belong to which $\tilde{\mathbf{X}}(m, n)$ vectors.

is not unique. This gives

$$
\begin{aligned}
\Pr[Y_1 = i, Y_2 = i] \\
= (\tilde{p}_4)^K - (\tilde{p}_4)^K - K\tilde{p}_2(\tilde{p}_4)^{K-1} \\
= \left(\sum_{l=i}^{N} p_l \right)^K - \left(\sum_{l=i+1}^{N} p_l \right)^K - Kp_i \left(\sum_{l=i+1}^{N} p_l \right)^{K-1}.
\end{aligned}
\tag{93}
$$

We can now evaluate (88) as

$$
\begin{aligned}
&\Pr[Y_1 = i] \\
&= \sum_{j=1}^{N} \Pr[Y_1 = i, Y_0 = j] \\
&= \sum_{j=1}^{i-1} \Pr[Y_1 = i, Y_0 = j] + \Pr[Y_1 = i, Y_0 = i] + 0 \\
&= \sum_{j=1}^{i-1} \left(Kp_j \left(\left(\sum_{l=i}^{N} p_l \right)^{K-1} - \left(\sum_{l=i+1}^{N} p_l \right)^{K-1} \right) \right) \\
&\quad + \left(\sum_{l=i}^{N} p_l \right)^K - \left(\sum_{l=i+1}^{N} p_l \right)^K - Kp_i \left(\sum_{l=i+1}^{N} p_l \right)^{K-1} \\
&= \left(\sum_{l=i}^{N} p_l \right)^K - \left(\sum_{l=i+1}^{N} p_l \right)^K - Kp_i \left(\sum_{l=i+1}^{N} p_l \right)^{K-1} \\
&\quad + K \left(\sum_{l=i}^{N} p_l \right)^{K-1} \left(\sum_{j=1}^{i-1} p_j \right) \\
&\quad - K \left(\sum_{l=i+1}^{N} p_l \right)^{K-1} \left(\sum_{j=1}^{i-1} p_j \right) \\
&= \Pr[Y_0 = i] + K \left(\sum_{l=i}^{N} p_l \right)^{K-1} \left(\sum_{j=1}^{i-1} p_j \right) \\
&\quad - K \left(\sum_{l=i+1}^{N} p_l \right)^{K-1} \left(\sum_{j=1}^{i} p_j \right),
\end{aligned}
\tag{94}
$$

which is the desired formula. Note that, although we refer to $\Pr[Y_0 = i]$ in the formula for $\Pr[Y_1 = i]$, this is only for notational simplicity; we do not wish to suggest some sort of interpretation relating the two quantities in this way.

Finally, we must compute $\Pr[Y_m = i]$ for $m = 2, \ldots, K - 1$. We take an approach similar to the $\Pr[Y_1 = i]$ case, and derive $\Pr[Y_m = i]$ using the joint probabilities $\Pr[Y_m = i, Y_{m-1} = j]$. As before, there are two cases, $i > j$, and $i = j$, as the m-th smallest element of $\mathbf{Z}$ cannot be smaller than the $(m-1)$-th element of $\mathbf{Z}$, and so $\Pr[Y_m = i, Y_{m-1} = j] = 0$ if $i < j$.

In the case where $i > j$, we have

$$\Pr[Y_m = i, Y_{m-1} = j]$$

$$= \sum_{k=1}^{m} \Pr[X_i \geq 1, X_{i-1} = \cdots = X_{j+1} = 0,$$

$$X_j = k, X_{j-1} + \cdots + X_1 = m - k]$$

$$= \sum_{k=1}^{m} \Pr[X_{i-1} = \cdots = X_{j+1} = 0, X_j = k,$$

$$X_{j-1} + \cdots + X_1 = 0] - \Pr[X_i = \cdots = X_{j+1} = 0,$$

$$X_j = k, X_{j-1} + \cdots + X_1 = m - k]$$

$$= (\Pr[X_{i-1} = \cdots = X_{j+1} = 0, X_j + \cdots + X_1 = m]$$

$$- \Pr[X_{i-1} = \cdots = X_j = 0, X_{j-1} + \cdots + X_1 = m])$$

$$- (\Pr[X_i = \cdots = X_{j+1} = 0, X_j + \cdots + X_1 = m]$$

$$- \Pr[X_i = \cdots = X_j = 0, X_{j-1} + \cdots + X_1 = m]).$$

$$(95)$$

Now we recast the four terms of (95) in terms of $\tilde{\mathbf{X}}(i-1, j+1)$, $\tilde{\mathbf{X}}(i-1, j)$, $\tilde{\mathbf{X}}(i, j+1)$, and $\tilde{\mathbf{X}}(i, j)$ respectively:

$$\Pr[Y_m = i, Y_{m-1} = j]$$

$$= \left(\Pr[\tilde{X}_4 = K - m, \tilde{X}_3 = 0, \tilde{X}_2 = 0, \tilde{X}_1 = m] \right.$$

$$\left. - \Pr[\tilde{X}_4 = K - m, \tilde{X}_3 = 0, \tilde{X}_2 = 0, \tilde{X}_1 = m] \right)$$

$$- \left(\Pr[\tilde{X}_4 = K - m, \tilde{X}_3 = 0, \tilde{X}_2 = 0, \tilde{X}_1 = m] \right.$$

$$\left. - \Pr[\tilde{X}_4 = K - m, \tilde{X}_3 = 0, \tilde{X}_2 = 0, \tilde{X}_1 = m] \right).$$

$$= \binom{K}{K-m} (\tilde{p}_4)^{K-j} (\tilde{p}_1)^j + \binom{K}{K-m} (\tilde{p}_4)^{K-j} (\tilde{p}_1)^j$$

$$+ \binom{K}{K-m} (\tilde{p}_4)^{K-j} (\tilde{p}_1)^j + \binom{K}{K-m} (\tilde{p}_4)^{K-j} (\tilde{p}_1)^j$$

$$= \binom{K}{K-m} \left(\left(\sum_{l=i}^{N} p_l \right)^{K-m} \left(\sum_{l=1}^{j} p_l \right)^m \right.$$

$$- \left(\sum_{l=i}^{N} p_l \right)^{K-m} \left(\sum_{l=1}^{j-1} p_l \right)^m$$

$$- \left(\sum_{l=i+1}^{N} p_l \right)^{K-m} \left(\sum_{l=1}^{j} p_l \right)^m$$

$$\left. + \left(\sum_{l=i+1}^{N} p_l \right)^{K-m} \left(\sum_{l=1}^{j-1} p_l \right)^m \right)$$

$$= \binom{K}{K-m} \left(\left(\sum_{l=i}^{N} p_l \right)^{K-m} - \left(\sum_{l=i+1}^{N} p_l \right)^{K-j} \right)$$

$$\left(\left(\sum_{l=1}^{j} p_l \right)^m - \left(\sum_{l=1}^{j-1} p_l \right)^m \right).$$

$$(96)$$

In the case where $i = j$, there are two sub-cases to consider: $i = j \neq 1$ and $i = j = 1$. In the former sub-case, we must have $X_i \geq 2$, and $X_{i-1} + \cdots + X_1 = b \leq m - 1$. Suppose that $X_i = 2 + k$ for some integer $k \in \{0, \ldots, K - 2\}$.

We know that $Y_m = i$ and $Y_{m-1} = i$, but that leaves k Y variables "adjacent" to Y_m and Y_{m-1} that must also have a value of i. Let n_l denote the number of variables $Y_{m'}$ that are equal to i and have $m' > m$, and n_s denote the number of variables $Y_{m'}$ that are equal to i and have $m' < m - 1$. Then $n_l + n_s = k$ and the following must be true: there are at most $K - 1 - m$ variables $Y_{m'}$ with $m' > m$, because there are only $K - 1$ total $Y_{m'}$ variables, and so $n_l \leq K - 1 - m$; moreover there are only $m - 2$ variables $Y_{m'}$ with $m' < m - 1$, and so $n_s \leq m - 2$. We will use these inequalities to place bounds on b as a function of k.

We first consider an upper bound on b. We have already seen that $b \leq m - 1$ in general, but the inequality on n_l induces a second upper bound on b that is sometimes stricter than the first. Note that $b = m - 1$ only if Y_{m-1} is the first $Y_{m'}$ variable with the value i; if $X_i = 2 + k$, then we must have $n_l = k$, and therefore $k \leq K - 1 - m$. Thus if $k > K - 1 - m$, then $b < m - 1$, where the maximum possible b decreases by one every time k increases by one. Indeed, the upper limit on b is imposed by $(m-1) - (k - (K-1-m)) = K - k - 2$; the general upper limit on be is then $b \leq \min\{m-1, K-k-2\}$.

The lower limit on b is obtained through similar reasoning, but we first note that the trivial lower limit on b is 0, which occurs when the number i constitutes (at least) the first m smallest values of $\mathbf{Z}$; in this case $k \geq m - 1$. If $k < m - 1$, then not all $Y_{m'}$ with $m' < m$ can have values of i. In general, $k + 2 + b \geq m + 1$, which implies that $b \geq m - 1 - k$. Then general lower bound on b is therefore $b \geq \max\{0, m - 1 - k\}$.

We are therefore now in a position to write

$$\Pr[Y_m = i, Y_{m-1} = i \neq 1]$$

$$= \sum_{k=0}^{K-2} \sum_{b=\max\{0, m-1-k\}}^{\min\{m-1, K-2-k\}}$$

$$\Pr[X_i = 2 + k, X_{i-1} + \cdots + X_1 = b]$$

$$= \sum_{k=0}^{K-2} \sum_{b=\max\{0, m-1-k\}}^{\min\{m-1, K-2-k\}}$$

$$\Pr[\tilde{X}_4 = K - k - 2 - b, \tilde{X}_3 = 0, \tilde{X}_2 = 2 + k, \tilde{X}_1 = b]$$

$$(97)$$

where the $\tilde{X}$ variables in (97) are with reference to $\tilde{\mathbf{X}}(i, i)$. This can be computed as

$$\Pr[\tilde{X}_4 = K - k - 2 - b, \tilde{X}_3 = 0, \tilde{X}_2 = 2 + k, \tilde{X}_1 = b]$$

$$= \binom{K}{K-k-2-b, b, 2+k} (\tilde{p}_4)^{K-k-2} (\tilde{p}_2)^{2+k} (\tilde{p}_1)^b$$

$$= \binom{K}{K-k-2-b, b, 2+k}$$

$$\left(\sum_{l=i+1}^{N} p_l \right)^{K-k-2-b} (p_i)^{2+k} \left(\sum_{l=1}^{i-1} p_l \right)^b$$

$$(98)$$

When $i = j = 1$, we simply have

$$
\begin{aligned}
&\Pr[Y_m = 1, Y_{m-1} = 1] \\
&= \Pr[X_1 \geq m + 1] \\
&= \sum_{k=0}^{K-1-m} \Pr[X_1 = j + 1 + k] \\
&= \sum_{k=0}^{K-1-m} \binom{K}{m+1+k} (p_1)^{m+1+k} (1 - p_1)^{K-m-1-k}
\end{aligned}
$$

(99)

Finally, we compute $\Pr[Y_m = i]$ as

$$
\begin{aligned}
&\Pr[Y_m = i] \\
&= \sum_{j=1}^{N} \Pr[Y_m = i, Y_{m-1} = j] \\
&= \sum_{j=1}^{i-1} \Pr[Y - m = i, Y - m - 1 = j] \\
&\quad + \Pr[Y_m = i, Y_{m-1} = i] \\
&= \binom{K}{K-m} \left(\left(\sum_{l=i}^{N} p_l \right)^{K-m} - \left(\sum_{l=i}^{N} p_l \right)^{K-m} \right) \\
&\quad \left(\sum_{j=1}^{i-1} \left(\sum_{l=1}^{j} p_l \right)^{m} - \left(\sum_{l=1}^{j-1} p_l \right)^{m} \right) \\
&\quad + \Pr[Y_m = i, Y_{m-1} = i] \\
&= \binom{K}{K-m} \left(\left(\sum_{l=i}^{N} p_l \right)^{K-m} - \left(\sum_{l=i+1}^{N} p_l \right)^{K-m} \right) \\
&\quad \left(\left(\sum_{l=1}^{i-1} p_l \right)^{m} \right) + \Pr[Y_m = i, Y_{m-1} = i]
\end{aligned}
$$

(100)

Combing (100) with (98) and (100) yields the desired result. This completes the proof.

B. Proof of Proposition 1

We wish to show that

$$
\begin{aligned}
&\mathbb{E} \left[\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}} \{ |W_{S \setminus \{k\}}^{(d_k)}| \} \right] \\
&= \sum_{j=1}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \binom{K-1-i}{j} \Pr[Y_i = l] v_{l,j} \\
&\quad + \sum_{i=0}^{K-1} \sum_{l=1}^{N} \Pr[Y_{K-i-1} = l] v_{l,0}
\end{aligned}
$$

(101)

if the memory inequality condition holds for the $v_{l,j}$ variables. We begin with an examination of the left hand side of the equation. Inside the expectation, we sum over all subsets $\mathcal{S}$ of the set of users $\mathcal{U}$. This can be rewritten as a double summation: in the inner summation, we sum over all subsets

of size $j + 1$, and in the outer summation, we sum over all j from 0 to $K - 1$, giving

$$
\begin{aligned}
&\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}} \{ |W_{S \setminus \{k\}}^{(d_k)}| \} \\
&= \sum_{j=0}^{K-1} \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset : |\mathcal{S}| = j+1} \max_{k \in \mathcal{S}} \{ |W_{S \setminus \{k\}}^{(d_k)}| \}
\end{aligned}
$$

(102)

Replacing the $|W_{\mathcal{S}}|$ variables with the appropriate $v_{l,j}$ variables, (102) becomes

$$
\sum_{j=0}^{K-1} \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset : |\mathcal{S}| = j+1} \max_{k \in \mathcal{S}} \{ v_{d_k, j} \}.
$$

(103)

For a fixed $j \geq 1$, we note that we send one transmission to each of the $\binom{K}{j+1}$ subsets of size $j + 1$. For a fixed $\mathbf{d}$, let k_i denote the user requesting the i-th most popular file, i.e. the file i-th smallest index. Then k_1 has requested the most popular file, and so by the memory inequality (35), $v_{d_{k_1}, j}$ is the largest variable for any transmission to a subset of which k_1 is a member. Since k_1 is a member of $\binom{K-1}{j}$ subsets of size $j+1$ that contain k_1 as a member, the inner summation of (103) will have $\binom{K-1}{j}$ terms with the value $v_{d_{k_1}, j}$. Similarly, user k_2 has requested the second most popular file, and so $v_{d_{k_2}, j}$ will be the largest subfile for all subsets that contain k_2 but don't contain k_1. This constitutes $\binom{K-2}{j}$ subsets of size $j + 1$.

This reasoning can be extended until all subsets are characterized in terms of their maximum $v_{l,j}$ variable. User k_i requests the i-th most popular file, and so $v_{d_{k_i}, j}$ will be the largest element sent in any subset containing k_i but not containing $k_1, k_2, \ldots, k_{i-1}$. Since there are $K - i$ users who are not users $k_1, \ldots, k_i$, and user k_i is already in the subset, there are $\binom{K-i}{j}$ subsets that contain k_i but not $k_1, k_2, \ldots, k_{i-1}$. We can therefore eliminate the $\max\{\}$ term from the inner sum of (103) to obtain, for $j = 1, \ldots, K-1$,

$$
\sum_{i=1}^{K} \binom{K-i}{j} v_{d_{k_i}, j}.
$$

(104)

As noted earlier, the memory inequality reverses for $j = 0$, so the least popular files take up the most memory in that case; the reasoning is the same as in the above, but we instead obtain

$$
\sum_{i=1}^{K} \binom{K-i}{0} v_{d_{k_{K+1-i}}, 0} = \sum_{i=1}^{K} v_{d_{k_{K+1-i}}, 0},
$$

(105)

All that remains is to compute the expectation of these terms with respect to the demand vectors. Using the linearity of expectation and the results of (102)-(105), the left hand side of (101) reduces to

$$
\sum_{j=1}^{K-1} \sum_{i=1}^{K} \binom{K-i}{j} \mathbb{E}[v_{d_{k_i}, j}] + \sum_{i=1}^{K} \mathbb{E}[v_{d_{k_{K+1-i}}, 0}]
$$

(106)

To compute the expected value of the $v_{d_{k_i}, j}$ variables ($j = 1, \ldots, K-1$), we note that it has N possible values, $v_{1,j}, v_{2,j}, \ldots, v_{N,j}$, and the probability of each outcome can

be obtained from Lemma 2 in the following way. We have $v_{d_{k_i},j} = v_{l,j}$ if l is the i-th most popular file in the request vector $\mathbf{d}$; since the files are labelled in terms of decreasing order of popularity, the i-th most popular file requested is represented by the i-th smallest index in $\mathbf{d}$. Thus the probability that $d_{k_i} = l$ is equivalent to the probability that l is the i-th smallest index in $\mathbf{d}$, and so by Lemma 2, we have

$$
\begin{aligned}
\mathbb{E}[v_{d_{k_i},j}] &= \sum_{l=1}^{N} \Pr[d_{k_i} = l] v_{l,j} \\
&= \sum_{l=1}^{N} \Pr[Y_{i-1} = l] v_{l,j}. \quad (107)
\end{aligned}
$$

For $j = 0$, the size ordering is reversed, so we are concerned with the *largest* indices of $\mathbf{d}$. However, as has been noted already, the i-th largest index of $\mathbf{d}$ must necessarily be the $K + 1 - i$-th smallest index of $\mathbf{d}$, which gives

$$
\begin{aligned}
\mathbb{E}[v_{d_{k_{K+1-i}},0}] &= \sum_{l=1}^{N} \Pr[d_{k_{K+1-i}} = l] v_{l,0} \\
&= \sum_{l=1}^{N} \Pr[Y_{K-i} = l] v_{l,0}. \quad (108)
\end{aligned}
$$

Combing (107)-(108), we see that (106) is equal to

$$
\sum_{j=1}^{K-1} \sum_{i=1}^{K} \binom{K-i}{j} \sum_{l=1}^{N} \Pr[Y_{i-1} = l] v_{l,j} \\
+ \sum_{i=1}^{K} \sum_{l=1}^{N} \Pr[Y_{K-i} = l] v_{l,0}. \quad (109)
$$

We complete the proof through a cosmetic change of variables $i' = i - 1$ to obtain the desired expression on the right-hand side of (101).

C. Proof of Proposition 3

We follow reasoning similar to what we have already seen in the previous proof, where users are divided into subsets that require the same amount of data to be sent to them, and then count how many such subsets there are. For this proof, however, we instead divide the various subsets into subsets containing only small-cache users, subsets containing only large-cache users, and subsets containing both large- and small-cache users.

But first, we note that since the files are all the same size and length, the transmission length will be independent of the request vector $\mathbf{d}$, and so we have

$$
\mathbb{E}\left[\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}} \{|W_{\mathcal{S} \setminus \{k\}}^{(d_k)}|\} \right] \\
= \sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}} \{|W_{\mathcal{S} \setminus \{k\}}^{(d_k)}|\}. \quad (110)
$$

As discussed above, the sum in the above expression is over all $\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset$, which we can separate into small,

large, and mixed sets. For a fixed subset size of $j + 1$, there are $\binom{K_S}{j+1}$ sets of small users, $\binom{K_L}{j+1}$ sets of large users, and

$$
\sum_{i=1}^{j} \binom{K_S}{i} \binom{K_L}{j+1-i} \quad (111)
$$

groups of at least one small user and at least one large user. For a set of $j + 1$ small users, every subfile in a single coded transmission is cached by j small users, and so has the size $v_{j,S}$. Similarly, for any set of $j + 1$ large users, the transmission has the size $v_{j,L}$.

For the mixed subset case, we must consider three cases. First, when there are at least 2 small users and 2 large users in the subset of $j + 1$ users, then since every subfile sent is cached on j of the $j + 1$ users, there must be at least 1 small user and 1 large users among those j users, and so every subfile must be of size $v_{j,M}$. However, if there is only one small user in the subset of $j + 1$ users, then the subfile requested by the small user will have been stored on the caches of j large users, and so will have size $v_{j,L}$. The length of the entire transmission will therefore also be of size $v_{j,L}$. The third case occurs when there is only one large users in the subset of $j + 1$ users. Then the subfile requested by the large user will be store on the caches of j small user and so will be of size $v_{j,S}$, while every other subfile is cached on a mixed set of j users and so will be of size $v_{j,M}$; the entire transmission will therefore be of length $v_{j,M}$. [2]

So, in addition to the $\binom{K_L}{j+1}$ transmissions of size $v_{j,L}$ sent for groups entirely consisting of entirely large users, there are $\binom{K_S}{1}\binom{K_L}{j}$ transmissions of the same size for those mixed subsets with only one small user. The total number of transmissions of size $v_{j,M}$ can then be simplified using Lemma 1 as

$$
\begin{aligned}
&\sum_{i=2}^{j} \binom{K_S}{i} \binom{K_L}{j+1-i} \\
&= \sum_{i=0}^{j+1} \binom{K_S}{i} \binom{K_L}{j+1-i} - \binom{K_S}{j+1} \\
&\quad - \binom{K_S}{1} \binom{K_L}{j} - \binom{K_L}{j+1} \\
&= \binom{K}{j+1} - \binom{K_S}{j+1} \\
&\quad - \binom{K_S}{1} \binom{K_L}{j} - \binom{K_L}{j+1} \quad (112)
\end{aligned}
$$

Altogether, (110) reduces to

$$
\sum_{j=0}^{K-1} \binom{K_s}{j+1} (v_{j,S} - v_{j,M}) + \binom{K}{j+1} v_{j,M} \\
+ \left(\binom{K_L}{j+1} + \binom{K_S}{1} \binom{K_L}{j} \right) (v_{j,L} - v_{j,M}),
$$

which is what we aimed to show. We make a special note that the formula is indeed sensible for $j = 0$: the $j = 0$ term

---

[2]In the case where a subset of size 2 contains one large user and one small user, obviously the entire transmission is of length $v_{1,L}$.

reduces to $\binom{K}{1}v_{0,M} = Kv_0$, as needed for the individual transmissions to the K users.

D. Proof of Proposition 4

We derive the terms of (80) in the order that they appear. In general, we do this using the following steps. First, we identify a certain group of subsets that have similar user composition; then for that group, we determine the number of transmissions that the largest subfile will be in, the number of transmissions that the second largest subfile will be in, and so on. Finally, we compute the expected size of the maximum subfile, the second largest subfile, and so on. This approach will be familiar from previous proofs, but we nevertheless repeat it here due to the complexity of (80).

The groups of subsets that the seven terms of (80) correspond to are, in order: subsets of size one, subsets of size greater than one containing only large users, subsets of size greater than one containing only small users, mixed subsets containing more than one user but only one small user, mixed subsets containing only one large user but more than one small user, subsets containing greater than or equal to $\max\{K_S, K_L\}+2$ users, and subsets containing at least two small and two large users that are less than $\max\{K_S, K_L\}+2$ users. We label these sets of subsets $\mathcal{S}_1, \ldots, \mathcal{S}_7$ respectively. The following lemma shows these sets form a partition (in the loose sense of the word discussed earlier) of $\mathcal{P}(\mathcal{U}) \setminus \emptyset$, and so the sum over all $\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset$ at the beginning of (80) can equivalently be done over all subsets in $\mathcal{S}_1$, then all subsets in $\mathcal{S}_2$, and so on, so that all subsets of users in $\mathcal{P}(\mathcal{U}) \setminus \emptyset$ will have been accounted for precisely once.

Lemma 3. For the sets $\mathcal{S}_1, \ldots, \mathcal{S}_7$ described above,

$$\mathcal{P}(\mathcal{U}) \setminus \emptyset = \bigcup_{i=1}^{7} \mathcal{S}_i, \tag{113}$$

and the $\mathcal{S}_i$ are mutually disjoint.

Proof: That $\bigcup_{i=1}^{7} \mathcal{S}_i \subseteq \mathcal{P}(\mathcal{U}) \setminus \emptyset$ is trivial: for any i, any set in $\mathcal{S}_i$ is a non-empty subset of users, and so must be contained in $\mathcal{P}(\mathcal{U}) \setminus \emptyset$. To show that $\mathcal{P}(\mathcal{U}) \setminus \emptyset \subseteq \bigcup_{i=1}^{7} \mathcal{S}_i$, consider the number of users in an arbitrary subset of users $\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset$: if it is one, then $\mathcal{S} \subseteq \mathcal{S}_1$ and if it is greater than or equal to $\max\{K_S, K_L\} + 2$, then it must be mixed because there are not enough of any one type of user to comprise the entire group, and so is must be that $\mathcal{S} \subseteq \mathcal{S}_6$. Otherwise, suppose $1 < |\mathcal{S}| < \max\{K_S, K_L\} + 1$, consider the number of small users, k_S, in $\mathcal{S}$. If $k_S = 0$, then $\mathcal{S}$ is contains only large users, and so $\mathcal{S} \subseteq \mathcal{S}_2$. If $k_S = 1$, then we have $\mathcal{S} \subseteq \mathcal{S}_4$. If $1 < k_S < |\mathcal{S}|$, then either the number of large users is either one, or more than one; if it is one, then $\mathcal{S} \subseteq \mathcal{S}_5$, while if it is more than one, then $\mathcal{S} \subseteq \mathcal{S}_7$. Finally, if $k_S = |\mathcal{S}|$, there are only small users, and so $\mathcal{S} \subseteq \mathcal{S}_3$, proving that indeed $\mathcal{P}(\mathcal{U}) \setminus \emptyset \subseteq \bigcup_{i=1}^{7} \mathcal{S}_i$. The mutual disjointedness is obvious once it is noted that a subset containing only one large/small user or no large/small users cannot exceed a size of $\max\{K_S, K_L\}$ or $\max\{K_S, K_L\} + 1$ respectively. Each

$\mathcal{S} \subseteq \mathcal{P}(\mathcal{U}) \setminus \emptyset$ thus falls into one and only one set $\mathcal{S}_i$, proving the lemma.

We remark before continuing that, given the specific values of K_S, K_L, some of the above subsets may be empty. As per the notation adopted in this paper, a sum over an empty set is identically zero, and so this will not affect our subsequent calculations. In terms of the expressions below, this will correspond to binomial coefficients $\binom{n}{k}$ with $n < 0$ or $k > n$, both of which, by our notation, gives $\binom{n}{k} = 0$.

So per the above discussion, we can change the summation over all subsets of $\mathcal{P}(\mathcal{U}) \setminus \emptyset$ into seven summations over one of the $\mathcal{S}_i$ each:

$$\mathbb{E}\left[\sum_{\mathcal{S} \in \mathcal{P}(\mathcal{U}) \setminus \emptyset} \max_{k \in \mathcal{S}}\{|W_{\mathcal{S} \setminus \{k\}}^{(d_k)}|\}\right]$$
$$= \sum_{i=1}^{7} \sum_{\mathcal{S} \in \mathcal{S}_i} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\mathcal{S} \setminus \{k\}}^{(d_k)}|\}\right] \tag{114}$$

This allows us to analyze each subset of subsets separately.

We begin with the analysis of $\mathcal{S}_1$, i.e. to broadcasts of individual users. Since each transmission is to only one person, we get

$$\sum_{\mathcal{S} \in \mathcal{S}_1} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\emptyset}^{(d_k)}|\}\right] = \sum_{k=1}^{K} \mathbb{E}\left[|W_{\emptyset}^{(d_k)}|\right] = \sum_{k=1}^{K} \mathbb{E}\left[v_{d_k,0}\right]$$

The above sum is over all users from $k = 1$ to $k = K$, i.e. in lexicographic order. But we can instead sum over all users by adding the user requesting the largest subfile, then the user requesting the second largest subfile, and so on. Using the index i to indicate the user requesting the $(i+1)$-th largest subfile, we can write the expectation $\mathbb{E}[v_{d_k,0}]$ in terms of the random variable Y_i as defined in Lemma 2 to obtain

$$\sum_{k=1}^{K} \mathbb{E}\left[v_{d_k,0}\right] = \sum_{i=0}^{K-1} \mathbb{E}\left[v_{f_d(i+1),0}\right] = \sum_{i=0}^{K-1} \sum_{l=1}^{N} \Pr[Y_i = l]v_{l,0}, \tag{115}$$

which is the first term of (80), with $v_{l,0} = v_{l,0}^M$ as per constraint (64). Here, $f_d(i)$ denotes the index of the i-th largest file in the request vector $\mathbf{d}$ (recall that $f(i)$ was used earlier to denote the i-th largest file in the set of all files).

We next consider $\mathcal{S}_2$, the set of user subsets with more than one user containing only large-cache users. There are $\binom{K_L}{j}$ user subsets of size $j+1$ in $\mathcal{S}_2$, for j values ranging from 1 to $K_L - 1$; we cannot have a subset of only large users that contains more members than there are large users. Since there are only large users in these subsets, the subfiles sent will stored on j large users caches, and so only subfiles of size $v_{j,l}^L$ are sent. As we saw in earlier proofs, the largest subfile requested (i.e. corresponding to the file with the smallest index), will be sent to $\binom{K_L-1}{j}$ subsets, the second largest subfile is the largest subfile for $\binom{K_L-2}{j}$ subsets, and in general, the i-th largest subfile sent will be sent in $\binom{K_L-i}{j}$

subsets, giving

$$\sum_{\mathcal{S} \in \mathcal{S}_2} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\mathcal{S}\setminus\{k\}}^{(d_k)}|\}\right]$$

$$= \sum_{j=1}^{K_L-1} \sum_{i=1}^{K_L} \binom{K_L - i}{j} \mathbb{E}\left[v_{f_d^L(i),j}^L\right]$$

$$= \sum_{j=1}^{K_L-1} \sum_{i=1}^{K_L} \binom{K_L - i}{j} \sum_{l=1}^{N} \Pr[Y_{i-1}^L = l] v_{l,j}^L$$

$$= \sum_{j=1}^{K_L-1} \sum_{i=0}^{K_L-1} \sum_{l=1}^{N} \binom{K_L - i + 1}{j} \Pr[Y_i^L = l] v_{l,j}^L, \quad (116)$$

where the last line is obtained by rearranging the terms and using a minor change of variable for the index of summation i. This is the second term of (80). Here we use $f_d^L(i)$ to refer to the i-th largest file requested within the set of large users, and by $\Pr[Y_i^L = l]$, we mean the probability that file l is the $i + 1$th largest file requested within the set of large users. We can compute $\Pr[Y_i^L = l]$ using Lemma 2 with N files (outcomes) and K_L users (trials). The change from the second to third lines above then follows immediately from the definition of expectation (see Appendix B).

Using identical reasoning for $\mathcal{S}_3$, the set of user subsets of size greater than 1 with only small users, we can obtain

$$\sum_{\mathcal{S} \in \mathcal{S}_3} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\mathcal{S}\setminus\{k\}}^{(d_k)}|\}\right]$$

$$= \sum_{j=1}^{K_S-1} \sum_{i=0}^{K_S-1} \sum_{l=1}^{N} \binom{K_S - i + 1}{j} \Pr[Y_i^S = l] v_{l,j}^S \quad (117)$$

which is the third term of (80). Here, $\Pr[Y_i^S = l]$ is the probability that file l is the $i + 1$-th largest file requested among all small users. This can also be computed using Lemma 2, but with N outcomes and K_S trials.

Next, we consider $\mathcal{S}_4$, the set of mixed subsets containing more than one user but only one small user. We saw in the non-uniform cache memory case that the coded transmissions to these kinds of groups will consist almost entirely of subfiles whose size is described by mixed variables $v_{l,j}^M$, because the subfiles are stored on a mixed subset of users' caches, save for one subfile whose size is described by a large variable $v_{l,j}^L$, because that subfile is stored only on large user caches. Due the memory inequality constraints (77)-(79) that prioritize cache size over file size, the one large variable (corresponding to the file requested by the one small user) will necessarily be the maximum value.

The size of the transmissions sent to these kinds of subsets will therefore depend on what files are requested by the small-cache users. Each small cache user will be in $\binom{K_L}{j}$ many of these subsets for a subset size of $j + 1$, where j takes values from 1 to K_L; if j was any larger, there would have to be more than one small user. We therefore have

$$\sum_{\mathcal{S} \in \mathcal{S}_4} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\mathcal{S}\setminus\{k\}}^{(d_k)}|\}\right] = \sum_{j=1}^{K_L} \sum_{i=1}^{K_S} \binom{K_L}{j} \mathbb{E}\left[v_{f_d^S(i),j}^L\right]$$

$$(118)$$

We use $f_d^S(i)$ to denote the index of the i-th largest file requested by a small-user. Consequently, the second sum in (118) is over all small cache users, in decreasing order of the file size they requested. This allows us to compute the expectation in (118) using the Y_i^S variables in the following way:

$$\sum_{\mathcal{S} \in \mathcal{S}_4} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\mathcal{S}\setminus\{k\}}^{(d_k)}|\}\right]$$

$$= \sum_{j=1}^{K_L} \sum_{i=1}^{K_S} \binom{K_L}{j} \sum_{l=1}^{N} \Pr[Y_{i-1}^S = l] v_{l,j}^L$$

$$= \sum_{j=1}^{K_L} \sum_{i=0}^{K_S-1} \sum_{l=1}^{N} \binom{K_L}{j} \Pr[Y_i^S = l] v_{l,j}^L. \quad (119)$$

The final step is once again attained with a rearranging of terms and a change of variable for the i index of summation. This gives the fourth term in (80).

The fifth term is obtained using similar reasoning. This term corresponds to $\mathcal{S}_5$, the set of mixed user subsets containing exactly one large user and more than one small user. Here, the transmitted subfiles will all be stored on the caches of a mixed subset of users, except for the subfile requested by the large user, which will be stored on the caches of every other user in the subset, i.e. all small users. The large user's requested subfile will have a size described by a small variable $v_{l,j}^S$, and so due to the memory inequality constraints (77)-(79), will never be the largest subfile transmitted; once again, it is the small-cache user requests that determine the largest subfile. The largest subfile requested by a small user will be transmitted to $\binom{K_S-1}{j-1}\binom{K_L}{1}$ subsets of size $j + 1$; the second largest subfile requested among small users will be the largest subfile transmitted when the largest subfile requested is not also being transmitted to that subset, and so will be transmitted $\binom{K_S-2}{j-1}\binom{K_L}{1}$ times. In general, the i-th largest subfile requested among small users will be transmitted only when the previous $i - 1$ largest subfiles are not also being transmitted, and so will be sent $\binom{K_S-i}{j-1}\binom{K_L}{1}$ times.

There are subsets of size 3 through $K_S + 1$ in $\mathcal{S}_5$, so indexing subset size with $j + 1$ yields

$$\sum_{\mathcal{S} \in \mathcal{S}_5} \mathbb{E}\left[\max_{k \in \mathcal{S}}\{|W_{\mathcal{S}\setminus\{k\}}^{(d_k)}|\}\right]$$

$$= \sum_{j=2}^{K_S} \sum_{i=1}^{K_S-1} \binom{K_S - i}{j-1} \binom{K_L}{1} \mathbb{E}[v_{f_d^S(i),j}^M]$$

$$= \sum_{j=2}^{K_S} \sum_{i=1}^{K_S-1} \binom{K_S - i}{j-1} \binom{K_L}{1} \left(\sum_{l=1}^{N} \Pr[Y_{i-1}^S = l] v_{l,j}^M\right)$$

$$= \sum_{j=2}^{K_S} \sum_{i=0}^{K_S-2} \sum_{l=1}^{N} \binom{K_S - 1 - i}{j-1} \binom{K_L}{1} \Pr[Y_i^S = l] v_{l,j}^M.$$

$$(120)$$

The last line is once again obtained through rearranging terms and doing a change of variables for the index of summation i. This is the fifth term of (80).

The sixth term of (80) contains the terms for S_6, the set of user subsets with more than $\max\{K_S, K_L\} + 1$ users. These subsets are precisely large enough that they are all mixed and have at least two of each user type in them. Thus only subfiles stored on the caches of mixed subsets of users will be sent, and so they will have a size given by a mixed variable $v_{l,j}^M$. The only factor that determines the largest subfile for a given subset will therefore be the file index. There are no restrictions on subset composition, and so we find ourselves in a familiar situation: the largest subfile requested will be the largest file sent for $\binom{K-1}{j}$ subsets, the second largest subfile requested will be the largest subfile sent for $\binom{K-2}{j}$ subsets, and so on, such that the i-th largest subfile is the largest subfile sent for $\binom{K-i}{j}$ subsets. This gives

$$
\sum_{S \in S_6} \mathbb{E}\left[\max_{k \in S}\{|W_{S\setminus\{k\}}^{(d_k)}|\}\right]
$$
$$
= \sum_{j=\max\{K_S,K_L\}+1}^{K-1} \sum_{i=1}^{K} \binom{K-1-i}{j} \mathbb{E}\left[v_{f_d(i),j}^M\right]
$$
$$
= \sum_{j=\max\{K_S,K_L\}+1}^{K-1} \sum_{i=0}^{K-1} \binom{K-1-i}{j} \Pr[Y_i = l] v_{l,j}^M,
$$
(121)

the sixth term of (80).

The seventh and final term of (80) is by far the most complicated term. It corresponds to S_7, the set of subsets with at least two large-cache and two small-cache users, but less than $\max\{K_S, K_L\} + 2$ users. Here, every subfile sent will be stored on the caches of a mixed subset of users and so will have a size given by a mixed variable $v_{l,j}^M$. The difficulty arises when we try to characterize the number of transmissions for each of the largest subfile requested, second largest subfile requested, and so on - these numbers are dependent on whether the file was requested by a large user or by a small user. An example will illustrate this fact: consider the third largest file requested among all users and transmissions to subsets of 5 users. If a large-cache user has requested the largest subfile, and another large-cache user requests the second largest subfile, then the number of transmissions where the third largest subfile requested is the largest subfile transmitted is $\sum_{n=1}^{2} \binom{K_L-3}{n}\binom{K_S}{4-n}$ if the subfile is requested by a large-cache user, and $\sum_{n=2}^{3} \binom{K_L-2}{n}\binom{K_S-1}{4-n}$ if it is requested by a small-cache user. These two number are clearly not equal in general, and so the cache size of the user making the request matters.

Nevertheless, it is possible to compute the expected rate with a number of terms that scales as a polynomial function of N and K. To this end, let $R_{\mathbf{d},j}^{S_i}(n)$ denote the number of bits sent to subsets in S_i of size $j+1$ as part of the transmissions required to satisfy the request $\mathbf{d}$, when the n-th largest file in $\mathbf{d}$ is the largest subfile transmitted in that subset. By this definition, we have $R_{\mathbf{d}} = \sum_{i=1}^{7} \sum_{j=0}^{K-1} \sum_{n=1}^{K} R_{\mathbf{d},j}^{S_i}(n)$. These values have been implicitly computed for S_1 through S_6 earlier in this appendix; we only introduce this opaque notation now because the complexity of the accounting done

for S_7 demands it.

The $R_{\mathbf{d},j}^{S_7}(n)$ quantity can be further decomposed: the number of bits sent in this case is equal to the product of the number of transmission sent in this case, denoted by $T(n)$, and the number of bits transmitted per transmission, which is given buy the appropriate $v_{l,j}^M$ variable. We can then compute the S_7 term of (80) using conditional expectation in the following way:

$$
\sum_{S \in S_7} \mathbb{E}\left[\max_{k \in S}\{|W_{S\setminus\{k\}}^{(d_k)}|\}\right] = \mathbb{E}\left[\sum_{j=3}^{\max\{K_S,K_L\}} \sum_{n=1}^{K} R_{\mathbf{d},j}^{S_7}(n)\right]
$$
$$
= \sum_{j=3}^{\max\{K_S,K_L\}} \sum_{n=1}^{K} \mathbb{E}\left[R_{\mathbf{d},j}^{S_7}(n)\right]
$$
(122)

For a fixed j value, we compute the expectation conditional on the fact that the n-th largest file is file l, i.e. $Y_{n-1} = l$:

$$
\sum_{n=1}^{K} \mathbb{E}\left[R_{\mathbf{d},j}^{S_7}(n)\right]
$$
$$
= \sum_{n=1}^{K}\sum_{l=1}^{N} \mathbb{E}\left[R_{\mathbf{d},j}^{S_7}(n)|Y_{n-1}=l\right]\Pr[Y_{n-1}=l]
$$
$$
= \sum_{n=1}^{K}\sum_{l=1}^{N} \mathbb{E}\left[v_{f_d(n),j}^M T(n)|Y_{n-1}=l\right]\Pr[Y_{n-1}=l]
$$
$$
= \sum_{n=1}^{K}\sum_{l=1}^{N} v_{l,j}^M \mathbb{E}\left[T(n)|Y_{n-1}=l\right]\Pr[Y_{n-1}=l]. \quad (123)
$$

The last line (123) is obtained because, given that $Y_{n-1} = l$, it follows immediately that $f_d(n) = l$, and so we have $v_{f_d(n),j}^M = v_{l,j}^M$, which is no longer a random quantity.

Comparing (123) to the form of (80) in the statement of the proposition, we see that all that remains is to show that $\mathbb{E}[T(n)|Y_{n-1}=l] = \nu(n,j)$. First, we note that $\mathbb{E}[T(n)|Y_{n-1}=l] = \mathbb{E}[T(n)]$, since the number of transmissions in which the n-th largest subfile requested is the largest subfile sent to the subset depends only on the index n but not the identity of the n-th largest subfile. Next, letting $S(n)$ denote the number of small users in the set of users who requested the $n-1$ largest files we further decompose the expectation using conditional expectation:

$$
\mathbb{E}[T(n)] = \sum_{m=0}^{n-1} \mathbb{E}[T(n)|S(n)=m]\Pr[S(n)=m]. \quad (124)
$$

And further, if $D_S(n) = 1$ represents the event that a small user requested the n-th largest file and $D_S(n) = 0$ representing the event that a large user did it, we have

$$
\mathbb{E}[T(n)] = \sum_{m=0}^{n-1} \mathbb{E}[T(n)|S(n)=m]\Pr[S(n)=m]
$$
$$
= \sum_{m=0}^{n-1}\sum_{r=0}^{1} \mathbb{E}[T(n)|S(n)=m, D_S(n)=r]
$$
$$
\Pr[D_S(n)=r|S(n)=m]\Pr[S(n)=m] \quad (125)
$$

Now with (123)-(125), we have finally expressed the original expectation in (122) in terms of quantities that can be computed directly.

We begin with $\Pr[S(n) = m]$, the probability that there are m small-cache users in the set of users who have the $n - 1$ largest files among all files requested. Since all users have the same preferences, these probabilities are simply determined by the relative numbers of large and small users. Indeed $S(n)$ has a hypergeometric distribution: the probability that m of the $n - 1$ largest files requested are requested by small users (and thus $n - 1 - m$ of these files are requested by large users) is given by

$$\Pr[S(n) = m] = \frac{\binom{K_S}{m}\binom{K_L}{n-1-m}}{\binom{K}{n-1}}. \qquad (126)$$

Next we consider $\Pr[D_S(n) = r|S(n) = m]$, which is obtained with similar reasoning. Once again, since the large and small users have the same preferences, only their relative numbers will determine the probabilities. Since $n - 1$ users have already been accounted for, there are $K - (n-1)$ users left to choose from, and if m of them are small users, there are $K_S - m$ small users left and $K_L - (n - 1 - m)$ large users left. This gives

$$\Pr[D_S(n) = 1|S(n) = m] = \frac{K_S - m}{K - n + 1} \qquad (127)$$

and

$$\Pr[D_S(n) = 0|S(n) = m] = \frac{K_L - n + 1 + m}{K - n + 1} \qquad (128)$$

Finally, we compute $\mathbb{E}\left[T(n)|S(n) = m, D_S(n) = r\right]$; the number of transmissions $T(n)$ is deterministic given the values of $S(n)$ and $D_S(n)$, so no probabilities will be involved in the calculation. First, if $r = 1$, i.e. a small user has the n-th largest file request. In this case, the corresponding subfile is the largest subfile transmitted for any transmission to a subset with at least one other small user and two large users, but not the users responsible for the $n - 1$ larger requested files. This number is obtained as

$$\mathbb{E}\left[T(n)|S(n) = m, D_S(n) = 1\right]$$
$$= \sum_{i=1}^{j-2}\binom{K_S - m - 1}{i}\binom{K_L - n + 1 + m}{j - i} \qquad (129)$$

for a subset size of $j + 1$. The equivalent number for $r = 0$, i.e. a large user has the n-th largest file request, is

$$\mathbb{E}\left[T(n)|S(n) = m, D_S(n) = 0\right]$$
$$= \sum_{i=2}^{j-1}\binom{K_S - m}{i}\binom{K_L - n + m}{j - i}. \qquad (130)$$

Substituting the expressions in (126)-(130) into the appropriate places in (123) - (125) yields the desired term, i.e. the seventh and final term of (80), which concludes the proof.

REFERENCES

[1] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, February 2014.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," in *IEEE Int. Symp. Inf. Theory*, July 2013, pp. 1077–1081.

[3] ——, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[4] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

[5] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," in *IEEE Conf. Computer Commun. Workshops*, April 2014, pp. 221–226.

[6] ——, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1146–1158, Feb 2017.

[7] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3923–3949, 2017.

[8] ——, "On the average performance of caching and coded multicasting with random demands," in *11th Int. Symp. Wireless Commun. Systems*, Aug 2014, pp. 922–926.

[9] A. S. Cacciapuoti, M. Caleffi, M. Ji, J. Llorca, and A. M. Tulino. (2016, May) Speeding up Future Video Distribution via Channel-Aware Caching-Aided Coded Multicast. [Online]. Available: https://arxiv.org/abs/1605.05026.

[10] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip. (2016, Sep.) Correlation-Aware Distributed Caching and Coded Delivery. [Online]. Available: https://arxiv.org/abs/1609.05836.

[11] ——. (2016, Sep.) Cache-Aided Coded Multicast for Correlated Sources. [Online]. Available: https://arxiv.org/abs/1609.05831.

[12] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *IEEE Conf. Computer Commun.*, April 2015, pp. 756–764.

[13] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3108–3141, May 2017.

[14] J. Hachem, N. Karamchandani, and S. Diggavi, "Multi-level coded caching," in *IEEE Int. Symp. Inf. Theory*, June 2014, pp. 56–60.

[15] ——, "Effect of number of users in multi-level coded caching," in *IEEE Int. Symp. Inf. Theory*, June 2015, pp. 1701–1705.

[16] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," in *Inf. Theory and Applications Workshop*, Feb 2015, pp. 98–107.

[17] S. Jin, Y. Cui, H. Liu, and G. Caire. (2017, Jul.) Structural properties of uncoded placement optimization for coded delivery. [Online]. Available: https://arxiv.org/abs/1707.07146.

[18] J. Zhang, X. Lin, C. C. Wang, and X. Wang, "Coded caching for files with distinct file sizes," in *IEEE Int. Symp. Inf. Theory*, June 2015, pp. 1686–1690.

[19] C. Li, "On rate region of caching problems with non-uniform file and cache sizes," *IEEE Commun. Letters*, vol. 21, no. 2, pp. 238–241, Feb 2017.

[20] A. Sengupta, R. Tandon, and T. C. Clancy, "Improved approximation of storage-rate tradeoff for caching via new outer bounds," in *IEEE Int. Symp. Inf. Theory*, June 2015, pp. 1691–1695.

[21] 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.

[22] M. Ji, A. Tulino, J. Llorca, and G. Caire. (2015, Nov.) Caching-Aided Coded Multicasting with Multiple Random Requests. [Online]. Available: https://arxiv.org/abs/1511.07542.

[23] M. Ji, A. M. Tulino, J. Llorca, and G. Caire. (2014, Feb.) Caching and Coded Multicasting: Multiple Groupcast Index Coding. [Online]. Available: https://arxiv.org/abs/1402.4572.

[24] S. Wang, W. Li, X. Tian, and H. Liu. (2015, Apr.) Coded Caching with Heterogenous Cache Sizes. [Online]. Available: https://arxiv.org/abs/1504.01123.

[25] M. M. Amiri, Q. Yang, and D. Gunduz. (2016, Nov.) Decentralized Coded Caching with Distinct Cache Capacities. [Online]. Available: https://arxiv.org/abs/1611.01579.

[26] M. M. Amiri, Q. Yang, and D. Gndz, "Decentralized coded caching with distinct cache capacities," in *50th Asilomar Conf. Signals, Systems and Computers*, Nov 2016, pp. 734–738.

[27] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Centralized coded caching with heterogeneous cache sizes," in *IEEE Wireless Commun. and Netw. Conf.*, March 2017, pp. 1–6.

[28] S. Saeedi Bidokhti, M. Wigger, and R. Timo. (2016, May) Noisy Broadcast Networks with Receiver Caching. [Online]. Available: https://arxiv.org/abs/1605.02317.

[29] R. Timo and M. Wigger. (2015, May) Joint Cache-Channel Coding over Erasure Broadcast Channels. [Online]. Available: https://arxiv.org/abs/1505.01016.

[30] S. S. Bidokhti, M. Wigger, and R. Timo, "Erasure broadcast networks with receiver caching," in *IEEE Int. Symp. Inf. Theory*, July 2016, pp. 1819–1823.

[31] ——, "An upper bound on the capacity-memory tradeoff of degraded broadcast channels," in *9th Int. Symp. Turbo Codes and Iterative Inf. Processing*, Sept 2016, pp. 350–354.

[32] J. Zhang and P. Elia. (2016, Jun.) Wireless Coded Caching: A Topological Perspective. [Online]. Available: https://arxiv.org/abs/1606.08253.

[33] S. Saeedi Bidokhti, M. Wigger, and A. Yener. (2017, Feb.) Benefits of Cache Assignment on Degraded Broadcast Channels. [Online]. Available: https://arxiv.org/abs/1702.08044.

[34] A. Ghorbel, M. Kobayashi, and S. Yang, "Content delivery in erasure broadcast channels with cache and feedback," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6407–6422, Nov 2016.

[35] M. M. Amiri and D. Gunduz. (2017, Feb.) Cache-Aided Data Delivery over Erasure Broadcast Channels. [Online]. Available: https://arxiv.org/abs/1702.05454.

[36] A. Destounis, M. Kobayashi, G. Paschos, and A. Ghorbel. (2017, Jan.) Alpha Fair Coded Caching. [Online]. Available: https://arxiv.org/abs/1701.07730.

[37] A. Ghorbel, K.-H. Ngo, R. Combes, M. Kobayashi, and S. Yang. (2017, Feb.) Opportunistic Content Delivery in Fading Broadcast Channels. [Online]. Available: https://arxiv.org/abs/1702.02179.

[38] L. Zheng, Q. Yan, Q. Chen, and X. Tang. (2016, Nov.) On the Coded Caching Delivery Design over Wireless Networks. [Online]. Available: https://arxiv.org/abs/1611.04853.

[39] W. Huang, S. Wang, L. Ding, F. Yang, and W. Zhang. (2015, Apr.) The Performance Analysis of Coded Cache in Wireless Fading Channel. [Online]. Available: https://arxiv.org/abs/1504.01452.

[40] A. S. Cacciapuoti, M. Caleffi, M. Ji, J. Llorca, and A. M. Tulino, "Speeding up future video distribution via channel-aware caching-aided coded multicast," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2207–2218, Aug 2016.

[41] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE Conf. Computer Commun.*, March 2012, pp. 1107–1115.

[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, Dec 2013.

[43] C. Tian. (2016, Oct.) Symmetry, Outer Bounds, and Code Constructions: A Computer-Aided Investigation on the Fundamental Limits of Caching. [Online]. Available: https://arxiv.org/abs/1611.00024.

[44] Q. Yu, M. A. Maddah-Ali, and A. Salman Avestimehr. (2016, Sep.) The Exact Rate-Memory Tradeoff for Caching with Uncoded Prefetching. [Online]. Available: https://arxiv.org/abs/1609.07817.