entropy

MDPI

Article

Efficient Algorithms for Coded Multicasting in Heterogeneous Caching Networks

Giuseppe Vettigli [1], Mingyue Ji [2,*], Karthikeyan Shanmugam [3], Jaime Llorca [4], Antonia M. Tulino [1,4] and Giuseppe Caire [5]

[1] Department of Electrical Engineering and Information Technology (DIETI), Universitá di Napoli Federico II, 80138 Napoli, Italy; g.vettigli86@gmail.com (G.V.); antoniamaria.tulino@unina.it (A.M.T.)
[2] Department of Electrical and Computer Engineering (ECE), University of Utah, Salt Lake City, UT 84112, USA
[3] IBM Research, New York, NY 10598, USA; karthikeyanshanmugam88@gmail.com
[4] Department of Math and Algorithms, Nokia Bell Labs, Murray Hill, NJ 07738, USA; jaime.llorca@nokia-bell-labs.com (J.L.); a.tulino@nokia-bell-labs.com (A.M.T.)
[5] Faculty of Electrical Engineering and Computer Science (EECS), Technical University of Berlin, 10587 Berlin, Germany; caire@tu-berlin.de
* Correspondence: mingyue.ji@utah.edu

Received: 14 January 2019; Accepted: 13 March 2019; Published: 25 March 2019

Abstract: Coded multicasting has been shown to be a promising approach to significantly improve the performance of content delivery networks with multiple caches downstream of a common multicast link. However, the schemes that have been shown to achieve order-optimal performance require content items to be partitioned into several packets that grows exponentially with the number of caches, leading to codes of exponential complexity that jeopardize their promising performance benefits. In this paper, we address this crucial performance-complexity tradeoff in a heterogeneous caching network setting, where edge caches with possibly different storage capacity collect multiple content requests that may follow distinct demand distributions. We extend the asymptotic (in the number of packets per file) analysis of shared link caching networks to heterogeneous network settings, and present novel coded multicast schemes, based on *local graph coloring*, that exhibit polynomial-time complexity in all the system parameters, while preserving the asymptotically proven multiplicative caching gain even for finite file packetization. We further demonstrate that the packetization order (the number of packets each file is split into) can be traded-off with the number of requests collected by each cache, while preserving the same multiplicative caching gain. Simulation results confirm the superiority of the proposed schemes and illustrate the interesting request aggregation vs. packetization order tradeoff within several practical settings. Our results provide a compelling step towards the practical achievability of the promising multiplicative caching gain in next generation access networks.

Keywords: caching networks; random fractional caching; coded caching; coded multicasting; index coding; finite-length analysis; graph coloring; approximation algorithms

1. Introduction

Recent information-theoretic studies [1–49] have characterized the fundamental limiting performance of several caching networks of practical relevance, in which throughput scales linearly with cache size, showing great promise to accommodate the exponential traffic growth experienced in today's communication networks [50]. In this context, a caching scheme is defined in terms of two phases: the *cache*

placement phase, which operates at a large time-scale and determines the content to be placed at the network caches, and the *delivery phase*, during which user requests are served from the content caches and sources in the network. Some of the network topologies studied include shared link caching networks [1,2,8–14], device-to-device (D2D) caching networks [17–19,33,34], hierarchical caching networks [24], multi-server caching networks [29], and combination caching networks [36–42].

Consider a network with one source (e.g., base station) having access to m files, and n users (e.g., small-cell base stations or end user devices), each with a cache memory of M files. In [17], the authors showed that if the users can communicate between each other via D2D communications, a simple distributed random caching policy and TDMA-based unicast D2D delivery achieves the order-optimal throughput $\Theta\left(\max\{\frac{M}{m}, \frac{1}{m}, \frac{1}{n}\}\right)$ whose linear scaling with M when $Mn \geq m$ exhibits a remarkable multiplicative caching gain, in the sense that the per-user throughput grows proportionally to the cache size M for fixed library size m, and it is independent of the number of users n in the system. Moreover, in this scheme each user caches entire files without the need for partitioning files into packets, and missing files are delivered via unicast transmissions between neighbor nodes, making it efficiently implementable in practice. We recall that order-optimality refers to the fact that the multiplicative gap between information-theoretic converse and achievable performance can be bounded by a constant number when $m, n \to \infty$.

In the case that users cannot communicate between each other, but share a multicast link from the content source, the authors in [8,9] showed that the use of coded multicasting (also referred to as index coding [51]) allows achieving the same order-optimal worst-case throughput as in the D2D caching network. In this case, however, in order to create enough coding opportunities during the delivery phase, requested files are required to be partitioned into a number of packets that grows exponentially with the number of users, leading to coding schemes of exponential complexity [8,9,21].

In [10,12], the authors considered the same shared link caching network, but under random demands characterized by a probability distribution, and proposed a scheme consisting of random aggregate popularity (RAP) placement and chromatic number index coding (CIC) delivery, referred to as RAP-CIC, proved to be order-optimal in terms of average throughput. The authors further provided optimal average rate scaling laws under Zipf [52] demand distributions, whose analytical characterization required resorting to a polynomial-time approximation of CIC, referred to as greedy constrained coloring (GCC). Using RAP-GCC, the authors further established the regions of the system parameters, in which multiplicative caching gains are potentially achievable. While GCC exhibits polynomial complexity in the number of users and packets, the order-optimal performance guarantee still requires, in general, the packetization order (number of packets per file) to grow exponentially with the number of users, as showed in [21].

It is then key to understand if the promised multiplicative caching gain, shown to be asymptotically achievable by the above-referenced schemes, can be preserved in practical settings of finite packetization order. In this context, we shall differentiate between coded multicast schemes that assume a deterministic vs. a random cache placement phase. Deterministic placement policies determine where to store file packets according to a deterministic procedure that takes into account the ID of each packet. In contrast, random placement policies, after determining the number of packets to be cached of each file at each cache, choose the exact packet IDs uniformly at random. While the increased structure of deterministic placement policies can be exploited to design more efficient coded multicast algorithms, random placement policies are desirable in practice, as they provide increased robustness by requiring less cache configuration changes under system dynamics.

The seminal work of [21] showed that all previously proposed schemes (based on both deterministic and random cache placement) required exponential packetization, and that under random placement,

no graph-coloring-based coded multicast algorithm can achieve multiplicative caching gains with sub-exponential packetization. Since the fundamental results of [21], several works have studied the now central problem in caching of finite file packetization. The authors in [53] connect the caching problem to resolvable combinatorial designs and derive a scheme that while improving exponentially over previous schemes [8,9,21], still requires exponential packetization. In [54], the authors introduce the combinatorial concept of Placement Delivery Array (PDA) and derive a caching scheme where the packetization scales super-polynomially with the number of users. The work in [22] establishes a connection with the construction of hypergraphs with extremal properties, and provides the first sub-exponential (but still intractable) scheme. Somewhat surprisingly, some of the authors of [21] introduced a new combinatorial design based on Ruzsa-Szeméredi graphs in [30] and showed that a linear scaling of the number of packets per file with n can be achieved for a throughput of $\Theta(n^{-\delta})$, where δ can be arbitrarily small. However, all the above studies focus on coded multicast algorithms that assume a deterministic cache placement phase. Under random cache placement, several coded multicast algorithms have been proposed in the context of homogenous shared link caching networks [55–60], including our previous work that serves as the basis for this paper.

In this work, we address the important problem of finite-length coded multicasting under random cache placement, focusing on a more general heterogeneous shared link caching network, in which caches with possibly different sizes collect possibly multiple requests according to possibly different demand distributions (see Figure 1). As shown in Figure 1, this scenario can be motivated by the presence of both end user caches and cache-enabled small-cell base stations or WLAN access points sharing a common multicast link. In this case, each small-cell base station can be modeled as a user cache placing multiple requests. In addition, multiple requests per user also arise in the presence of delay-tolerant content requests (e.g., file downloading). While there have been several information-theoretic studies of shared link caching networks with distinct cache sizes [61–63], and with multiple per-user requests [13,14,34,64,65], none of these works considered the finite-length regime nor addressed the joint effect of random demands, heterogenous cache sizes, and multiple per-user requests.

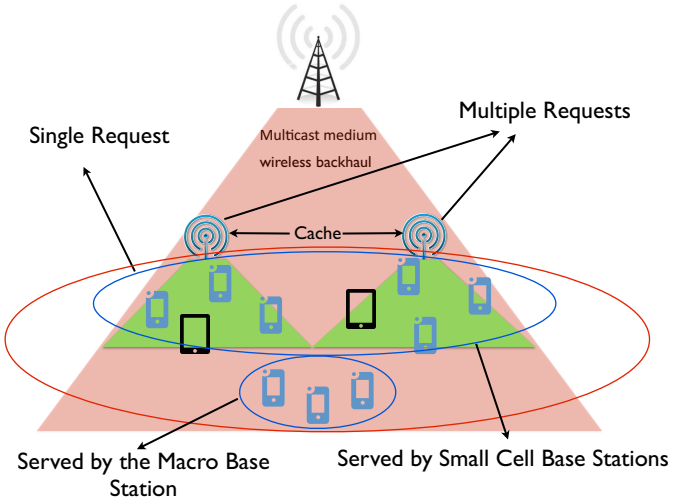

Figure 1. An example of the network model, which consists of a source node (base station in this figure) with access to the content library and connected to the users via a shared (multicast) link. Each user (end users and small-cell base stations) may have different cache size and request a different number of files according to their own demand distribution.

The contributions of this paper are as follows:

1. We provide a generalized model for heterogeneous shared link caching networks, in which users can have different cache sizes and make different number of requests according to different demand distributions.
2. We design two novel coded multicast algorithms based on *local graph coloring*, referred to as Greedy Local Coloring (GLC) and Hierarchical Greedy Local Coloring (HgLC) that exhibit polynomial-time complexity in both the number of caches and the packetization order. In combination with the Random Aggregate Popularity (RAP) placement policy of [10,12], we show that the overall schemes RAP-GLC and RAP-HgLC are order-optimal in the asymptotic file-length regime.
3. Focusing on the finite-length regime, in which content items can be partitioned into a finite number of packets, we show how the general advantage of local graph coloring is especially relevant when the number of per-user requests grow. We validate via simulations the superiority of RAP-GLC, especially with high number of per-user requests. We then show how RAP-HgLC, with a slight increase in the polynomial complexity order, further improves the caching gain of RAP-GLC, remarkably approaching the multiplicative gain that existing schemes can only guarantee in the asymptotic file-length regime.
4. We demonstrate that there is a tradeoff between the required packetization order and the number of requested files per user. In particular, for a given target gain, if the number of requests increases, then the number of packets per file can be reduced, while preserving the target gain. We further quantify the regime of per-user requests for which a caching scheme with unit packetization order (i.e., a scheme that treats only whole files) is order-optimal. Our analysis illustrates the key impact of content request aggregation in time and space on caching performance. That is, if edge caches can wait for collecting multiple requests over time and/or aggregate requests from multiple users, the same performance can be achieved with lower packetization order, and hence lower computational complexity.

The paper is organized as follows. Section 2 introduces the network model and problem formulation. Section 3 describes the construction of coded multicast algorithms using graph coloring, with special focus on the advantages of local graph coloring. Section 4 presents novel polynomial-time local-graph-coloring-based coded multicast schemes. Section 5 analyzes the effect of request aggregation on the performance–complexity tradeoff. Section 6 presents simulation results and related discussions. Finally, concluding remarks are given in Section 7.

2. Network Model and Problem Formulation

We consider a caching network formed by a source node with access to a content library, connected to several caching nodes/users via a single shared (multicast) link. Similar to previous works [8–10,12–14,21,22,30], we define a caching scheme in terms of two phases:

- *Placement phase*, which operates at a large time-scale and determines the content to be placed at the caching nodes,
- *Delivery phase*, during which users requests are served from the content caches and sources in the network.

However, differently from previous works, we generalize the model to a heterogeneous system in which each caching node has a possibly different cache size and requests a possibly different number of files. A practical example of our setting can be represented by a macro base station connected to several cache-enabled small-cell base stations, and a number of user devices served either by the macro base

station or by the small-cell base stations. In this setting, each small cell acts as a super user requesting multiple files resulting from the requests of the users it serves.

Specifically, the heterogeneous caching network consists of a single source node storing a library of files $\mathcal{F} = \{1, \ldots, m\}$, each with entropy F bits, and n user nodes $\mathcal{U} = \{1, \ldots, n\}$, each with a cache of storage capacity $M_u F$ bits (i.e., each user caches up to M_u files). Each user u can requests L_u $(1 \leq L_u \leq m)$ different files according to its individual request probability distribution. We assume that the library files have finite length and consequently a finite packetization order. Our main objective is to design a caching scheme that minimizes the number of transmissions required to satisfy the demands of all users.

In a homogeneous network setting with infinite packetization order, recent works [8–10,12–14] have shown that it is possible to satisfy a scaling number of users with only a constant number of multicast transmissions. The achievable schemes configure user caches with complementary (side) information during the caching phase, such that the resulting coded multicasting opportunities that arise during the delivery phase can be used to minimize the transmission rate (or load) over the shared multicast link. Specifically, reference [12] showed that under Zipf file popularity, a properly optimized random fractional placement policy, referred to as Random Aggregate Popularity (RAP) caching, achieves order-optimality when combined with a graph-coloring-based coded multicast scheme. Unfortunately, even in the homogenous setting, it was shown in [21] that a central limitation of all previous works is that they require infinite packetization order: all existing caching schemes achieve at most a factor of two gain when the packetization order is finite.

In this work, inspired by the fundamental throughput-delay-memory tradeoff derived in [21], our goal is to design computationally efficient schemes that provide good performance in the finite packetization regime. For the caching phase, (1) we restrict our placement policies to the class of random fractional schemes described in [9,10,12–14], proved to be order-optimal in the homogeneous setting. For the delivery phase, (2) we focus on the class of graph-coloring-based index coding schemes, and design two novel polynomial-time algorithms that employ local graph coloring on the (index coding) conflict graph [51].

2.1. Random Fractional Cache Placement

The class of random fractional placement schemes is described as follows:

1. Packetization: Each file is partitioned into B packets of equal-size F/B bits, where the integer B is referred to as the packetization order. Each packet is represented by a symbol in finite field $\mathbb{F}_{2^{F/B}}$, where we assume that F/B is large enough.
2. Random Placement: Each user u caches $p_{f,u} M_u B$ packets independently at random from each file f, where $p_{f,u}$ is the probability that file f is cached at user u, and satisfies $0 \leq p_{f,u} \leq 1/M_u, \forall f \in \mathcal{F}$ such that $\sum_{f=1}^{m} p_{f,u} = 1, \forall u \in \mathcal{U}$.

We introduce a *caching distribution* matrix $\mathbf{P} = [p_{f,u}] \in \mathbb{R}_{+}^{m \times n}$, where $f \in \mathcal{F}$ and $u \in \mathcal{U}$. Please note that the number of packets of file f cached at user u, $p_{f,u} M_u B$, can be directly determined from the caching distribution matrix $\mathbf{P}$. As described in [10,12–14], the caching distribution must be properly optimized to balance the gains from local cache hits (where requested packets are served by the local cache) and coded multicast opportunities (where requested packets are served by coded transmissions that simultaneously satisfy distinct user requests). When this is the case, we refer to the cache placement scheme as Random Aggregate Popularity (RAP) caching (see e.g., [10,12–14]). Given the number of packets to be cached of a given file, the actual indices of the packets to be cached are chosen uniformly at random, and independently across users. We use $\mathbf{C}_{u,f}$ to denote the set of packets of file f cached at user u and $\mathbf{C} = \{\mathbf{C}_{u,f}\}$ with $u \in \mathcal{U}$ and $f \in \mathcal{F}$ to denote the *packet-level* cache placement realization.

The goal of the placement phase is to configure the user caches to create coding opportunities during the delivery phase that allow serving distinct user requests via common multicast transmissions.

Compared to deterministic placement [8], random placement schemes allow configuring user caches with lower complexity and increased robustness, i.e., changes in system parameters (e.g., number of users, number files, file popularity) require less changes in users' cache configurations [12].

Recall that the placement phase operates at a much larger time-scale than the delivery phase, and hence is unaware of the requests in the subsequent delivery rounds. Therefore, the placement phase can be designed according to the demand distribution, but must be independent of the requests realizations.

2.2. Random Multiple Requests

Each user $u \in \mathcal{U}$ requests L_u $(1 \leq L_u \leq m)$ files independently from other users, following a probability distribution $q_{f,u}$ with $q_{f,u} \in [0,1]$ and $\sum_{f=1}^{m} q_{f,u} = 1$ (i.e., for each request of user u, file f is chosen with probability $q_{f,u}$). We introduce a *demand distribution* matrix $\mathbf{Q} = [q_{f,u}] \in \mathbb{R}_{+}^{m \times n}$, where $f \in \mathcal{F}$ and $u \in \mathcal{U}$. In the following, we use $\mathbf{W} = \{\mathbf{W}_{u,f}\}$, with $u \in \mathcal{U}$ and $f \in \mathcal{F}$, to denote the *packet-level* demand realization where $\mathbf{W}_{u,f}$ denotes the packets of file f requested by user u.

The multiple-request parameters $\{L_u\}$ have a key operational meaning, in that it captures the possibility of edge caches to collect requests across time and space. That is, L_u may represent the amount of requests collected over time (given the delay tolerance of some content requests) as well as the amount of requests collected across space from users served by the given edge cache (e.g., when edge caches are located at helper nodes or small-cell base stations serving multiple individual users).

2.3. Performance Metric

For given realizations of the random fractional cache placement and the random multiple requests, the goal is to design a delivery scheme that minimizes the rate over the shared multicast link required to satisfy all user requests. Since one placement phase is followed by an arbitrarily large number of delivery rounds (each characterized by a new independent request realization), the rate (or load) of the system refers only to the delivery phase (i.e., asymptotically the cache placement costs no rate). Furthermore, it makes sense to consider the average rate, where averaging with respect to the users request distribution takes on the meaning of a time-averaged rate, invoking an ergodicity argument.

At each request round, let $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n\}$ be the demand realization, where $\mathbf{f}_u = \{f_{1,u}, f_{2,u}, \cdots, f_{L_u,u}\}, u \in \mathcal{U}$. The source node computes a multicast codeword as a function of the library and the demand realization $\mathbf{F}$. We assume that the source node communicates to the user nodes through an error-free deterministic shared multicast link.

Given the demand realization $\mathbf{F}$, let the total number of bits transmitted by the source node be $J(\mathbf{F})$. We are interested in the average performance of the coded multicast scheme, and hence define the average rate (or load) as the number of transmitted bits normalized by the file size:

$$R = \frac{\mathbb{E}\left[J(\mathbf{F})\right]}{F}, \tag{1}$$

where the expectation is over the random demand distribution.

3. Graph-Coloring-Based Coded Multicast Delivery

It is important to note that for given cache placement and demand realizations, the delivery phase of a caching scheme reduces to an index coding problem *with a twist*. The only difference with the conventional index coding problem introduced in [51] is that the cache information may contain *part of* (as opposed to entire) requested files, and that users may request *multiple* (as opposed to single) files. Nevertheless, as in index coding, the problem can still be represented by a *conflict graph* [10,12–14], where vertices represent requested packets, and an edge between two vertices indicates a conflict, in the

sense that the packet represented by one vertex is not present in the cache of the user requesting the packet represented by the other vertex. By construction, packets with no conflict in the graph can be simultaneously transmitted via an XOR operation. Performing graph coloring on the conflict graph and transmitting the packets via proper XOR operations, according to the graph coloring, results in an achievable linear index coding scheme, which we refer to as a coded multicast scheme.

In the following, we first illustrate how to construct the conflict graph, we then review classical linear index coding schemes, and then describe our proposed graph-coloring-based coded multicast schemes.

3.1. Conflict Graph Construction

Given cache placement realization $\mathbf{C}$ and demand realization $\mathbf{W}$, the directed conflict graph $\mathcal{H}^d_{\mathbf{C},\mathbf{W}} = (\mathcal{V}, \mathcal{E})$ can be constructed as follows:

- Vertices: For each packet request in $\mathbf{W}$, there is a vertex in $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$. Each vertex $v \in \mathcal{V}$ is uniquely identified or labeled by a *packet-user* pair $\{\rho(v), \mu(v)\}$, where $\rho(v)$ denotes the identity of the packet, and $\mu(v)$ the user requesting it. Hence, if a packet is requested by multiple users, such a packet is represented in as many vertices as the number of users requesting it. Such vertices have the same packet label $\rho(v)$, but different user label $\mu(v)$.
- Arcs: For any $v_1, v_2 \in \mathcal{V}$, there is an edge $(v_2, v_1) \in \mathcal{E}$ with direction from v_2 to v_1 if and only if $\rho(v_1) \neq \rho(v_2)$ and packet $\rho(v_1)$ is not in the cache of user $\mu(v_2)$.

To better understand the rationale behind the conflict graph and its construction, note that for any two vertices v_1 and v_2 that are labeled as $\{\rho(v_1), \mu(v_1)\}$ and $\{\rho(v_2), \mu(v_2)\}$, respectively, we have the following three possible cases:

- $\rho(v_1) \neq \rho(v_2)$ and $\mu(v_1) = \mu(v_2)$: This indicates that two different packets are requested by the same user. Then, v_1 and v_2 are mutually conflicting, in the sense that if sent within the same time-frequency resource they interfere with each other. Hence, in the conflict graph, they are connected with two directed edges, $(v_1, v_2) \in \mathcal{E}$ and $(v_2, v_1) \in \mathcal{E}$;
- $\rho(v_1) = \rho(v_2)$ and $\mu(v_1) \neq \mu(v_2)$: This indicates that the same packet is requested by two different users. Then, v_1 and v_2 are not conflicting, and hence not connected in the conflict graph; i.e., $(v_1, v_2) \notin \mathcal{E}$ and $(v_2, v_1) \notin \mathcal{E}$;
- $\rho(v_1) \neq \rho(v_2)$ and $\mu(v_1) \neq \mu(v_2)$: This indicates that two different packets are requested by two different users. In this case, if packet $\rho(v_1)$ is in the cache of user $\mu(v_2)$, then, even if $\rho(v_1)$ and $\rho(v_2)$ are sent within the same time-frequency resource, user $\mu(v_2)$ will not suffer from interference, since, using its cache information, it can cancel out the undesired packet $\rho(v_1)$ from the received signal. On the other hand, if packet $\rho(v_1)$ is not in the cache of user $\mu(v_2)$, then v_1 conflicts with v_2, and a directed edge is drawn from v_2 to v_1. Similarly, $(v_1, v_2) \in \mathcal{E}$ if and only if $\rho(v_2) \notin \mathbf{C}_{\mu(v_1)}$.

Based on the above construction, it follows that the number of interference dimensions faced by a given node is at most the number of its outgoing neighbors.

To illustrate the construction of the directed conflict graph $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$, we present the following example.

Example 1. *We consider a network with* $n = 3$ *users denoted as* $\mathcal{U} = \{1, 2, 3\}$ *and* $m = 3$ *files denoted as* $\mathcal{F} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. *We assume* $M_u = 1, \forall u \in \mathcal{U}$ *and partition each file into three packets. For example,* $\mathrm{A} = \{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3\}$. *Let* $p_{\mathrm{A},u} = p_{\mathrm{B},u} = p_{\mathrm{C},u} = \frac{1}{3}$ *for* $u \in \mathcal{U}$, *which means that one packet from each of* $\mathrm{A}, \mathrm{B}, \mathrm{C}$ *is stored in each user's cache. For the sake of notational convenience, we assume a symmetric caching realization, where the caching configuration* $\mathbf{C}$ *is given by* $\mathbf{C}_{u,\mathrm{A}} = \{\mathrm{A}_u\}, \mathbf{C}_{u,\mathrm{B}} = \{\mathrm{B}_u\}, \mathbf{C}_{u,\mathrm{C}} = \{\mathrm{C}_u\}$*). That is, the cache configuration of each user* $u \in \mathcal{U}$ *is* $\mathbf{C}_u = \{\mathrm{A}_u, \mathrm{B}_u, \mathrm{C}_u\}$. *We let each user make two requests, i.e.,* $L_u = 2$ $(\forall u \in \mathcal{U})$. *Specifically, we let user* 1 *request* A, B, *user* 2 *request* B, C, *and user* 3 *request* C, A, *i.e.,* $\mathbf{f}_1 = \{\mathrm{A}, \mathrm{B}\}, \mathbf{f}_2 =$

$\{B, C\}, \mathbf{f}_3 = \{C, A\}$*), such that* $\mathbf{W}_1 = \{A_2, A_3, B_2, B_3\}, \mathbf{W}_2 = \{B_1, B_3, C_1, C_3\}, \mathbf{W}_3 = \{A_1, A_2, C_1, C_2\}$. *The associated directed conflict graph is shown in Figure 2.*

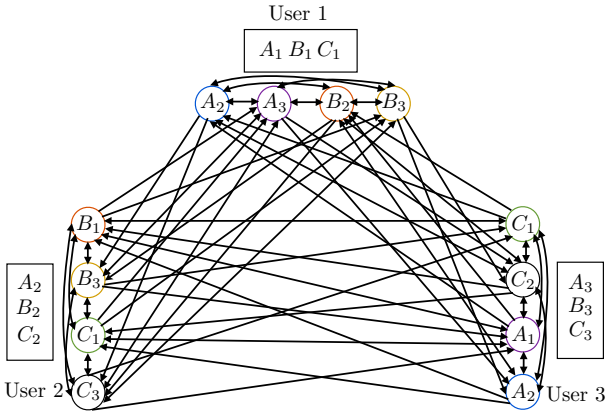

Figure 2. An example of the construction of the directed conflict graph (this figure needs to be viewed in color). The color of each circle in this figure represents the coloring of each vertex.

3.2. Code Construction

Let ω_v denote the content (or realization) of packet $\rho(v)$, $v \in \mathcal{V}$, represented by a symbol in $\mathbb{F}_q$. In general, in a linear index coding scheme of length ℓ, every vertex v is associated with a "coding" vector $\mathbf{g}_v \in \mathbb{F}_q^{\ell \times 1}$ where $v \in [1 : |\mathcal{V}|]$. Let $\mathbf{G} = [\mathbf{g}_1, \ldots \mathbf{g}_{|\mathcal{V}|}]$ and $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_{|\mathcal{V}|}]^{\mathsf{T}}$. Then, the transmitted codeword, $\mathbf{x} \in \mathbb{F}_q^{\ell \times 1}$, is built as follows:

$$\mathbf{x} = \sum_{v \in \mathcal{V}} \omega_v \mathbf{g}_v = \mathbf{G}\boldsymbol{\omega}, \tag{2}$$

Let $\mathcal{N}(v) = \{w : (v, w) \in \mathcal{E}\}$ be the out-neighborhood of v. For any feasible scalar linear index coding scheme of the form (2), the following *interference alignment* condition is necessary: For every vertex v, the coding vector $\mathbf{g}_v$ should be linearly independent of all the coding vectors assigned to the out-neighborhood of v.

In the following, we describe how to construct coding vectors satisfying the interference alignment condition for every vertex. For ease of notation, we use $\mathcal{H}^d$ to denote the directed conflict graph, and $\mathcal{H}$ to represent its underlying undirected skeleton, where the direction of edges is ignored. Recall that an undirected skeleton of a directed graph $\mathcal{H}^d$ is an undirected graph where there is an undirected edge between v_1 and v_2 if, between v_1 and v_2, there is a directed edge in either or both directions in $\mathcal{H}^d$.

3.2.1. Graph Coloring and Chromatic Number

A well-known procedure to construct the coding vectors $\{\mathbf{g}_v,\ v \in \mathcal{N}(v)\}$ is the *coloring* of $\mathcal{H}^d$. In the following, when used without any qualification, a coloring of a directed graph is considered to be a proper (vertex) coloring of its underlying undirected skeleton $\mathcal{H}$, where a proper coloring is a labeling of the graph's vertices with colors, such that no two vertices sharing the same edge have the same color. Please note that by definition, any subset of nodes with the same color in a proper coloring form an *independent set* (i.e., a subset of nodes in a graph, no two of which share the same edge). A coloring using at most k colors is called a (proper) k-coloring. The smallest number of colors needed in a proper coloring of $\mathcal{H}^d$ is called its chromatic number, and is denoted by $\chi(\mathcal{H}^d)$. In the following, we explain why a coloring of $\mathcal{H}^d$ provides a way to design the coding vectors $\{\mathbf{g}_v,\ v \in \mathcal{N}(v)\}$. Let ξ be the total

number of colors in a given coloring of $\mathcal{H}^d$. Let $\mathbf{e}_i$ be the i-th unit vector in the space $\mathbb{F}_q^{\ell\times 1}$, with $\ell = \xi$, i.e., $\mathbf{e}_i = [0, 0, \cdots, 1, \cdots, 0, 0]^\mathsf{T}$, where the 1 is in the i-th position. Now, if vertex v is colored with color i, then, its coding vector is $\mathbf{g}_v = \mathbf{e}_i$. Making this choice for the coding vectors, the associated achievable rate is given by $\frac{\xi}{B}$. Since neighbors are assigned different colors, the interference alignment condition is satisfied for every vertex. Recalling the definition of $\chi(\mathcal{H}^d)$, it is immediate to see that the best achievable rate due to conflict graph coloring is given by $\frac{\chi(\mathcal{H}^d)}{B}$, and, according to the construction of the conflict graph, it is loosely bounded by:

$$\sum_{f\in\mathbf{f}_u}(1 - p_{f,u}M_u) \leq \frac{\chi(\mathcal{H}^d)}{B} \leq \sum_{u=1}^{n}\sum_{f\in\mathbf{f}_u}(1 - p_{f,u}M_u), \tag{3}$$

indicating that the achievable rate is a constant with regards to B. A much tighter bound will be given in Section 4.1.

3.2.2. Local Graph Coloring and Local Chromatic Number

More efficient sets of coding vectors can be constructed using the approach proposed in [66], which exploits the direction information in $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$, resulting in the following advanced coding scheme:

Definition 1 (**Local Coloring Number**). *Given a proper coloring* $\mathbf{c}$ *of* $\mathcal{H}^d$*, the associated local chromatic number is defined as:*

$$\xi_{\text{lc}}(\mathbf{c}) = \max_{v\in\mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))| \tag{4}$$

where $\mathcal{N}^+(v)$ *is the closed out-neighborhood of vertex* v *(i.e., vertex* v *and all its ongoing neighbors* $\mathcal{N}(v)$*) and* $|\mathbf{c}(\mathcal{N}^+(v))|$ *is the total number of colors in* $\mathcal{N}^+(v)$ *for a given proper color assignment* $\mathbf{c}$*.*

The minimum local coloring number over all proper colorings is referred to as the *local chromatic number* and is formally defined as follows:

Definition 2 (**Local Chromatic Number**). *The directed local chromatic number of a directed graph* $\mathcal{H}^d$ *is defined as:*

$$\chi_{\text{lc}}(\mathcal{H}^d) = \min_{\mathbf{c}\in\mathcal{C}} \xi_{\text{lc}}(\mathbf{c}) \tag{5}$$

where $\mathcal{C}$ *denotes the set of all proper coloring assignments of* $\mathcal{H}^d$*,* $\mathcal{N}^+(v)$ *is the closed out-neighborhood of vertex* v*, and* $|\mathbf{c}(\mathcal{N}^+(v))|$ *is the total number of colors in* $\mathcal{N}^+(v)$ *for a given proper color assignment* $\mathbf{c}$*.*

Encoding Scheme: For a given realization of the cache placement (**C**) and user requests (**W**), let us consider the conflict graph $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$ as in Section 3.1. Given a (proper) ξ-coloring (i.e., a proper coloring of graph $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$ with ξ colors), we compute the associated local coloring number ξ_{lc}. Set $\ell = \xi_{\text{lc}}$ and $p = \xi$. Then, consider the columns of the generator $\mathbf{H}$ of an $\ell \times p$ Maximum Distance Separable (MDS) [67] code over the field $\mathbb{F}_q : q > p$. If the color of a vertex v is i, then the coding vector $\mathbf{g}_v$ assigned to vertex v is given by i-th column $\mathbf{h}_i$ of $\mathbf{H}$. Then, the transmitted multicast codeword, $\mathbf{x} \in \mathbb{F}_q^{\ell\times 1}$, is given by (2).

Decoding Scheme: In any closed out-neighborhood, there are at most ℓ different colors (from the definition of local coloring). Since every ℓ columns of $\mathbf{H}$ are linearly independent (from the defining property of MDS codes), the coding vectors in any closed out-neighborhood have full rank, satisfying

the interference alignment condition. The message ω_v at vertex v is obtained at user v as follows: (1) Using side information at user v, cancel out message parts corresponding to all vertices outside $\mathcal{N}^+(v)$, i.e., $\mathbf{x}' = \mathbf{x} - \sum_{u \notin \mathcal{N}^+(v)} \omega_u \mathbf{g}_u$. This is possible because, by the definition of the conflict graph $\mathcal{H}^d$, the messages $\{\omega_u\}_{u \notin \mathcal{N}^+(v)}$ are available as side information at user v and the encoding mechanism is known to all the users. (2) Find a vector $\mathbf{z}$ in the dual space of $\{\mathbf{g}_u\}_{u \in \mathcal{N}^+(v) \setminus \{v\}}$ such that $\mathbf{z}^T \mathbf{x}' \neq 0$ (this is possible since $\mathbf{g}_v$ is linearly independent of $\{\mathbf{g}_u\}_{u \in \mathcal{N}^+(v) \setminus \{v\}}$ because of the local chromatic number-based construction). Now, $\mathbf{z}^T \mathbf{x}' = (\mathbf{z}^T \mathbf{g}_v)\omega_v$. Therefore, user v recovers its own message. It follows that all users can recover all the requested packets employing such linear scheme.

Achievable Rate: The coding scheme constructed as described above achieves a rate given by ξ_{lc}/B, where B is the number of packets per file.

Example 2. *We consider an example shown in Figure 3. First, we assign colors to each vertex such that the total number of colors $\xi = 5$, and count the local coloring number, which is $\xi_{\text{lc}} = 4$. Then, we construct the generator matrix $\mathbf{A}$ of a $(\xi = 5, \xi_{\text{lc}} = 4)$ MDS code, which is given by*

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{6}$$

After that, we assign the columns of $\mathbf{A}$ to $\mathbf{g}_v$, corresponding from the left to the right to the vertices with the packets $\{\mathrm{A}_2, \mathrm{A}_3, \mathrm{B}_2, \mathrm{B}_1, \mathrm{A}_1\}$, as shown in Figure 3. Finally, the transmitted codewords can be generated which are the rows of the right-hand side of (2)

$$X(1) = \mathrm{A}_1 \oplus \mathrm{A}_2, X(2) = \mathrm{A}_1 \oplus \mathrm{A}_3 \tag{7}$$

$$X(3) = \mathrm{A}_1 \oplus \mathrm{B}_1, X(4) = \mathrm{A}_1 \oplus \mathrm{B}_2 \tag{8}$$

where the length of the code is $\xi_{\text{lc}}/B = 4/3$ file units. It can be easily verified that every user can decoded its desired packets with the cached ones.

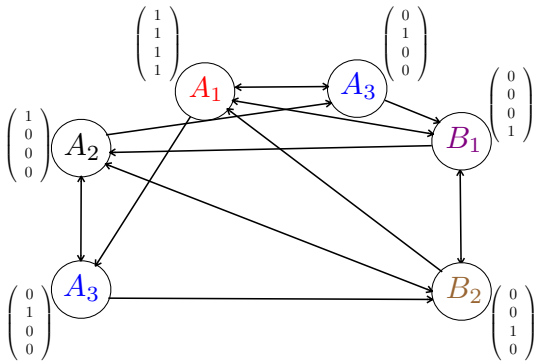

Figure 3. An illustration of coded multicast codewords construction based on local coloring (this figure needs to be viewed in color). The total number of colors is $\xi = 5$, and the local coloring number is $\xi_{\text{lc}} = 4$.

It is immediate to see that the best achievable rate due to local coloring is obtained by computing the local chromatic number of $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$ and using its associated coloring to design the coding vectors, yielding a rate χ_{lc}/B. However, note that to compute χ_{lc}, we must optimize over all proper colorings to find the

local chromatic number. As with the chromatic number, this can be cast as an *Integer Program* and it is hence an NP-hard problem. To overcome this limitation, in Section 4, we propose a greedy approach that (i) exhibits polynomial-time complexity in all the system parameters, (ii) achieves close to optimal performance for finite packetization order, and (iii) is asymptotically (i.e., for infinite packetization order) order-optimal.

3.3. Benefits of Local Coloring

Consider the following relation established for general directed graphs in [66]:

$$\chi_{\text{lc}}(\mathcal{H}^d) \leq \chi(\mathcal{H}^d) \leq \chi_{\text{lc}}(\mathcal{H}^d) O(\log n). \tag{9}$$

Focusing on the conflict graph of interest $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$, the number of vertices can be as large as $B \sum_{u \in \mathcal{U}} L_u$. It then follows from (9) that the gap between the local chromatic number and the chromatic number can be as large as $\log(B \sum_{u \in \mathcal{U}} L_u)$. Please note that this multiplicative factor grows with the number of packets per file B and the number of per-user requests L_u, supporting the extra benefit of local coloring in the multiple-request scenario. In addition, the higher the number of per-user requests, the higher the directionality of the conflict graph, which is the main factor exploited by local coloring to reduce the achievable rate (see Section 3.2.2), further supporting the suitability of local coloring in increasingly practical settings where there is some form of spatial or temporal request aggregation.

4. Proposed Algorithms and Performance Analysis

As stated earlier, computing the local chromatic number is NP-hard. To circumvent this challenge, in this section, we propose two greedy coded multicast schemes, which together with the cache placement described in Section 2.1, yield the following two caching schemes: Randomized Aggregate Popularity-Greedy Local Coloring (RAP-GLC) and Randomized Aggregate Popularity-Hierarchical greedy Local Coloring (RAP-HgLC). In both cases, the steps for obtaining the coded multicast scheme are as follow:

i. Given a realization of the cache placement ($\mathbf{C}$) and of the user requests ($\mathbf{W}$), build the conflict graph $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$ as in Section 3.1.
ii. Use any of the above algorithms (GLC or HgLC) to compute a proper coloring. Let ξ denote the number of colors used by either of the above algorithms to color $\mathcal{H}^d_{\mathbf{C},\mathbf{W}}$. Let ξ_{lc} be the associated local coloring number.
iii. Consider a (ξ, ξ_{lc}) MDS code and compute the corresponding coded multicast scheme as described in Section 3.2.2.

4.1. Randomized Aggregate Popularity-Greedy Local Coloring (RAP-GLC)

The RAP-GLC algorithm generalizes the RAP-GCC (Random Aggregate Popularity-Greedy Constrained Coloring) algorithm introduced in [12]. RAP-GCC is a caching scheme based on random fractional caching for the placement phase and a coded multicast scheme built on greedy-graph-coloring -based linear index coding [51,68] for the delivery phase. RAP-GLC is more general than RAP-GCC in two aspects: (1) conventional coloring is replaced by local coloring to leverage possible gains in the multiple-request scenario, as described in Sections 3.2.1 and 3.3, and (2) RAP-GLC adaptively (depending on the demand realization) chooses between naive or coded multicasting according to a threshold parameter, instead of sticking to one of them (as in RAP-GCC).

4.1.1. RAP-GLC Algorithm Description

The algorithm associates to each vertex v a label or tag, composed of two fields i.e., $\mathcal{K}_v \equiv (\mathcal{T}_D(v), \mathcal{T}_C(v))$ with $\mathcal{T}_C(v)$ denoting the subset of users caching the packet associated with vertex v, i.e.,

$$\mathcal{T}_C(v) \stackrel{\Delta}{=} \{u \in \mathcal{U} : \rho(v) \in \mathbf{C}_u\}, \tag{10}$$

and $\mathcal{T}_D(v)$ denoting the subset of users requesting the packet associated with vertex v, i.e.,

$$\mathcal{T}_D(v) \stackrel{\Delta}{=} \{u \in \mathcal{U} : \rho(v) \in \mathbf{W}_u\}, \tag{11}$$

which includes the user itself $\mu(v)$ who requests $\rho(v)$ and all the others requesting $\rho(v)$. Please note that the cardinality of $\mathcal{T}_D(v)$ indicates the popularity of packet $\rho(v)$. Furthermore, let

$$\mathcal{T}_v = \{\mu(v)\} \cup \mathcal{T}_C(v).$$

Given a vertex v, if the cardinality of $\mathcal{T}_D(v)$ is higher than a predetermined threshold parameter $t \in \{0 \cdots, n\}$ i.e., $|\mathcal{T}_D(v)| > t$, then all vertices v' such that $\rho(v) = \rho(v')$ are colored with the same color, leading to a naive multicast transmission scheme. If $|\mathcal{T}_D(v)| \leq t$, then RAP-GLC greedily looks for a maximal set of vertices with the same $\mathcal{T}_v$ (Algorithm 1, Line 14) and colors them with the same color if there is no conflict among the vertices (Algorithm 1, Line 15). The threshold parameter t is subject to optimization, as described in Section 4.1.2.

Doing this, RAP-GLC computes a valid coloring of the conflict graph $\mathcal{H}$. Finally, the algorithm computes its associated local coloring number (Algorithm 1, Line 24). The coding scheme employed is based on the MDS code described in Section 3.2.1 associated with the above local coloring.

Algorithm 1 RAP-GLC

1: Let $\mathcal{C} = \emptyset$;
2: Let $\mathbf{c} = \emptyset$;
3: **while** $\mathcal{V} \neq \emptyset$ **do**
4: Pick an arbitrary vertex v in $\mathcal{V}$; Let $\mathcal{I} = \{v\}$;
5: Let $\mathcal{V}' = \mathcal{V} \setminus \{v\}$;
6: **if** $\{\, |\mathcal{T}_D(v)| > t \,\}$ **then**
7: **for all** $v' \in \mathcal{V}'$ with $\rho(v') = \rho(v)$ **do**
8: $\mathcal{I} = \mathcal{I} \cup v'$;
9: **end for**
10: Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
11: Let $\mathbf{c}[\mathcal{I}'] = c$;
12: $\mathcal{V} = \mathcal{V} \setminus \mathcal{I}'$.
13: **else**
14: **for all** $v' \in \mathcal{V}'$ with $\mathcal{T}_{v'} \equiv \mathcal{T}_v$ **do**
15: **if** {There is no edge between v' and $\mathcal{I}$} **then**
16: $\mathcal{I} = \mathcal{I} \cup v'$;
17: **end if**
18: **end for**
19: Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
20: Let $\mathbf{c}[\mathcal{I}] = c$;
21: $\mathcal{V} = \mathcal{V} \setminus \mathcal{I}$.
22: **end if**
23: **end while**
24: **return** the local coloring number $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$ and the corresponding color assignment $\mathbf{c}(\mathcal{N}^+(v))$ for each v;

Time Complexity: In Algorithm 1, both the outer while-loop starting at Line 3, and the inner for-loop starting at Line 6 iterate at most $|\mathcal{V}|$ times, and all other operations inside the loops take constant time. Therefore, the complexity of RAP-GLC is $O(|\mathcal{V}|^2)$ or, equivalently, $O(n^2B^2)$, since $|\mathcal{V}| \leq nB$, which is polynomial in $|\mathcal{V}|$ (or n, B).

4.1.2. RAP-GLC Performance Analysis

In the following, we quantify the performance of RAP-GLC in the asymptotic regime when the number of users and files is kept constant while the packetization order is sent to infinity. Denoting by $\mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, t)]$ the asymptotic average achievable rate of RAP-GLC for a fixed threshold t, the threshold parameter t is optimized to minimize $\mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, t)]$. Hence, denoting by $\bar{R}^{\text{RAP}-\text{GLC}}$ the average rate achieved by RAP-GLC with optimized t, i.e.,

$$\bar{R}^{\text{RAP}-\text{GLC}} = \min_t \mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, t)],$$

we have that

$$\bar{R}^{\text{RAP}-\text{GLC}} \leq \min\{\mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, n)], \mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, 0)].$$

Since $\mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, 0)]$ is just the rate achieved via naive multicasting, then, an upper bound on the average asymptotic performance of RAP-GLC can be obtained by upper bounding $\mathbb{E}[R^{\text{RAP}-\text{GLC}}(\mathbf{P}, \mathbf{Q}, n)]$ which can be obtained by generalizing the asymptotic performance analysis of RAP-GCC derived in [10,12] using conventional graph coloring in the homogeneous shared link caching network to the case of using local coloring in the heterogeneous caching network. Specifically, we extend the order-optimality analysis under single per-user requests ($L = 1$) in the asymptotic regime of $B \to \infty$ [10,12], to that under multiple $L > 1$ per-use requests [13,14]. These theoretical results will serve as rate lower bounds for the finite-length performance of our proposed algorithms.

Let $L = \max_u L_u$ and order $L_u, u \in \mathcal{U}$ as a decreasing sequence $L_{[1]} \geq L_{[2]} \geq L_{[3]}, \ldots, L_{[n]}$, where $L_{[i]}$ is the i-th largest L_u and $[i] = u$ for some $u \in \mathcal{U}$. It can be seen that $L_{[1]} = \max_u L_u$ and $L_{[n]} = \min_u L_u$. Let $n_j = \sum_{[i]} 1\{L_{[i]} - j \geq 0\} > 0$, where $1 \leq j \leq L_{[1]}$ and $1\{\cdot\}$ is the indicator function. Let $\mathcal{U}_{n_j} = \{[i] \in \mathcal{U} : 1\{L_{[i]} - j \geq 0\}\}$. In the next theorem, we provide a performance guarantee of the RAP-GLC algorithm.

Theorem 1. *For any given m, n, M_u, the random caching distribution $\mathbf{P}$ and the random request distribution $\mathbf{Q}$, the average achievable rate of the RAP-GLC algorithm, $\bar{R}^{\text{RAP}-\text{GLC}}$ satisfies*

$$\bar{R}^{\text{RAP}-\text{GLC}} \leq \min\{\psi(\mathbf{P}, \mathbf{Q}), \bar{m} - \bar{M}\}, \tag{12}$$

when $B \to \infty$, where,

$$\bar{m} = \sum_{f=1}^{m} \left(1 - \prod_{u=1}^{n} \left(1 - q_{f,u}\right)^{L_u}\right), \tag{13}$$

$$\bar{M} = \sum_{f=1}^{m} \left(1 - \prod_{u=1}^{n} \left(1 - q_{f,u}\right)^{L_u}\right) \min_u p_{f,u} M_u, \tag{14}$$

$$\psi(\mathbf{P}, \mathbf{Q}) = \sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} \lambda(u, f, \mathcal{U}^\ell),$$

with $\mathcal{U}^\ell$ denoting a set of users with cardinality ℓ,

$$\lambda(u, f_u, \mathcal{U}^\ell) = (1 - p_{f_u,u} M_u) \times \prod_{k \in \mathcal{U}^\ell \setminus \{u\}} (p_{f_u,k} M_k) \prod_{k \in \mathcal{U} \setminus \mathcal{U}^\ell} (1 - p_{f_u,k} M_k) \tag{15}$$

and

$$\rho_{f,u,\mathcal{U}^\ell} \stackrel{\Delta}{=} \mathbb{P}(f = arg \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \lambda(u, f_u, \mathcal{U}^\ell)),$$

denoting the probability that f is the file whose $p_{f,u}$ maximizes the term $\lambda(u, f_u, \mathcal{U}^\ell)$ among $\mathbf{f}(\mathcal{U}^\ell)$ (the set of files requested by $\mathcal{U}^\ell$).

Proof. See Appendix A. □

Using the explicit expression for $\bar{R}^{\mathrm{RAP-GLC}}$ in Theorem 1, we can optimize the caching distribution for a wide class of heterogeneous network models to minimize the number of transmissions. We use $\mathbf{P}^*$ to denote the caching distribution that minimizes $R^{\mathrm{RAP-GLC}}$.

Remark 1. *For the sake of the numerical evaluation of $\psi(\mathbf{q}, \mathbf{p})$, it is worthwhile to note that the probabilities $\rho_{f,u,\mathcal{U}^\ell}$ can be easily computed as follows. Given the subset of users, $\mathcal{U}^\ell$ of cardinality ℓ, let $J_{u_1}, \ldots, J_{u_\ell}$ denote ℓ i.i.d. random variables each of them distributed over $\mathcal{F}$ with pmf $\mathbf{q}_{u_i}$, with $i = 1, \ldots, \ell$. Since $\lambda(u_1, J_{u_1}\mathcal{U}^\ell), \cdots, \lambda(u_\ell, J_{u_\ell}, \mathcal{U}^\ell)$ are i.i.d., the CDF of $Y_\ell \stackrel{\Delta}{=} \max\{\lambda(u_1, J_{u_1}, \mathcal{U}^\ell), \cdots, \lambda(u_\ell, J_{u_\ell}, \mathcal{U}^\ell)\}$ is given by*

$$\begin{aligned} \mathbb{P}(Y_\ell \leq y) &= \prod_{i=1}^{\ell} \mathbb{P}\left(\lambda(u_i, J_{u_i}, \mathcal{U}^\ell) \leq y\right) \\ &= \prod_{i=1}^{\ell} \left(\sum_{j \in \mathcal{F}: \lambda(u_i,j,\mathcal{U}^\ell) \leq y} q_{u_i,j} \right). \end{aligned} \tag{16}$$

Hence, it follows that

$$\begin{aligned} \rho_{f,u,\mathcal{U}^\ell} &= \mathbb{P}(Y_\ell = \lambda(u, f, \mathcal{U}^\ell)) \\ &= \prod_{i=1}^{\ell} \left(\sum_{j \in \mathcal{F}: g_\ell(j) \leq \lambda(u,f,\mathcal{U}^\ell)} q_{u_i,j} \right) - \prod_{i=1}^{\ell} \left(\sum_{j \in \mathcal{F}: g_\ell(j) < \lambda(u,f,\mathcal{U}^\ell)} q_{u_i,j} \right), \end{aligned} \tag{17}$$

which can be easily computed by sorting the values $\{\lambda(u_i, j, \mathcal{U}^\ell) : j \in \mathcal{F}, u_i \in \mathcal{U}^\ell\}$.

Nevertheless, as shown in [21], when B is finite or is not exponential in n, the performance of RAP-GLC can degrade significantly, compromising the promising multiplicative caching gain, although it is already an improved version of RAP-GCC in [12]. This brings us to the other main contribution where we propose a new algorithm that preserves the gain due to coded multicasting even when B is finite.

4.2. Randomized Aggregate Popularity-Hierarchical Greedy Local Coloring (RAP-HgLC) for Finite-Length Packetization

Similarly to RAP-GLC, RAP-HgLC has a predetermined parameter $t \in \{0, \cdots, n\}$ that is optimized to minimize its associated average achievable rate. However, in the RAP-HgLC algorithm, we arrange the vertices in a hierarchy and use this to design a more careful coloring algorithm. The key idea of

RAP-HgLC is to exploit the labeling of each vertex more efficiently. More specifically, as in RAP-GLC, RAP-HgLC associates to each vertex v a label or tag, composed by the two fields $\mathcal{K}_v \equiv (\mathcal{T}_D(v), \mathcal{T}_C(v))$, defined in (10) and (11).

4.2.1. RAP-HgLC Algorithm Description

Before jumping into the algorithm, we introduce the following useful notations and their definitions.

- $\mathcal{G}_i$: The i-th layer, $\mathcal{G}_i$ is initialized with the set of vertices $\{v : |\mathcal{T}_v| = i\}$ and at any point in the algorithm contains only vertices with $|\mathcal{K}_v| \geq i$. $\mathcal{G}_i$ is updated continuously in the algorithm. Therefore, higher numbered layers contain vertices with greater popularity.
- $\mathcal{W}_1 \subset \mathcal{G}_i$: a subset of $\mathcal{G}_i$ consists of all the vertices with $|\mathcal{K}_v| = i$ as well as a certain number of vertices with higher popularity (if available at any iteration), defined as

$$\mathcal{W}_1 = \left\{ v \in \mathcal{G}_i : \min_{v \in \mathcal{G}_i} |\mathcal{K}_v| \leq |\mathcal{K}_v| \leq \min_{v \in \mathcal{G}_i} |\mathcal{K}_v| + \left\lfloor a \left(\max_{v \in \mathcal{G}_i} |\mathcal{K}_v| - \min_{v \in \mathcal{G}_i} |\mathcal{K}_v| \right) \right\rfloor \right\}, \tag{18}$$

  where $a \in [0, 1]$ is a design parameter and $\mathcal{W}_1$ is updated with every iteration.
- $\mathcal{Q}_i$ (see Algorithm 2): another subset of $\mathcal{G}_i$ that is updated every iteration.
- $\mathcal{W}_2 \subset \mathcal{Q}_i$: a subset of vertices in $\mathcal{Q}_i$ defined as:

$$\mathcal{W}_2 = \left\{ v' \in \mathcal{Q}_i : \min_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| \leq |\mathcal{K}_{v'}| \leq \min_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| + \left\lfloor b \left(\max_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| - \min_{v' \in \mathcal{Q}_i} |\mathcal{K}_{v'}| \right) \right\rfloor \right\}, \tag{19}$$

  where $b \in [0, 1]$ is another design parameter.

Based on the above definitions, it follows that the total set vertices $\mathcal{V}$ forms an n-layer hierarchy with the i-th layer composed of the set of vertices $\mathcal{G}_i$.

Key Idea: Starting from layer n, at any layer $i \leq n$, the RAP-HgLC algorithm attempts to form an independent set of size at least i; when there are no more such independent sets, all remaining packets are dropped to layer $i - 1$, and transmission actions on those packets are *deferred* to later layers. This is the key difference between RAP-HgLC and RAP-GLC. That is, RAP-HgLC makes an extra effort to place nodes with large labels into large independent sets.

We will now describe how the above key idea is implemented in RAP-HgLC. The RAP-HgLC algorithm forms large independent sets in a "top-down" fashion, starting with the highest layer, and iteratively moving to lower layers until layer 1. The following two steps are performed at each layer:

1. **Step I**: The first step is similar to that in RAP-GLC algorithm. Given a vertex v, the algorithm first checks if the cardinality of $\mathcal{T}_D(v)$ is higher than t, i.e., $|\mathcal{T}_D(v)| > t$ then all the vertices v' such that $\rho(v) = \rho(v')$ are colored with the same color. If $|\mathcal{T}_D(v)| \leq t$ then the algorithm greedily finds independent sets of size i, where every vertex v in the independent set (Algorithm 2, Line 20) has the same $\mathcal{K}_v$ (Algorithm 2, Line 19). After removing these vertices, the rest of the vertices in $\mathcal{G}_i$ are left for the second step.
2. **Step II**: A candidate pool of vertices $\mathcal{W}_1 \subseteq \mathcal{G}_i$ is created. This set contains vertices v such that $|\mathcal{K}_v|$ being close to the smallest available $|\mathcal{K}_v|$'s. We randomly pick a vertex v from $\mathcal{W}_1$ (Algorithm 2, Line 31). The design parameter a determines how close is the picked $|\mathcal{K}_v|$ to the smallest available ones. We gradually form an independent set of size i with v included as follows: Form another set $\mathcal{W}_2$ (Algorithm 2, Line 34), excluding v, whose vertices have $|\mathcal{K}_{v'}|$ that is bigger but closer to that of v determined by b, sample repeatedly with replacement from it to grow the independent set. If an independent set of size at least i cannot be formed, we drop the vertex v to the lower layer $\mathcal{G}_{i-1}$, and take it into account in the next layer iteration. Otherwise, we assign a color to the independent set.

$\mathcal{W}_1$ is repeatedly formed and random sampling from $\mathcal{W}_1$ repeated till every vertex in $\mathcal{G}_i$ is dropped or colored.

Algorithm 2 HgLC

$\mathcal{C} = \emptyset$;
$\mathbf{c} = \emptyset$;
choose $a \in [0, 1]$
choose $b \in [0, 1]$
for all $i = n, n-1, \ldots, 2, 1$ **do**
  for all $v \in \mathcal{G}_i$ and $|\mathcal{K}_v| = i$ **do**
    $\mathcal{I} = \{v\}$;
    Let $\mathcal{V}' = \mathcal{V} \setminus \{v\}$;
    if $\{\ |\mathcal{T}_D(v)| > t\ \}$ **then**
      for all $v' \in \mathcal{V}'$ with $\rho(v') = \rho(v)$ **do**
        $\mathcal{I} = \mathcal{I} \cup v'$;
      end for
      Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
      Let $\mathbf{c}[\mathcal{I}] = c$;
      for all $i = n, n-1, \ldots, 2, 1$ **do**
        $\mathcal{G}_i = \mathcal{G}_i \setminus \mathcal{I}$;
      end for
    else
      for all $v' \in \mathcal{G}_i \setminus \mathcal{I}$ with $\mathcal{K}_{v'} \equiv \mathcal{K}_v$ **do**
        if {There is no edge between v' and $\mathcal{I}$} **then**
          $\mathcal{I} = \mathcal{I} \cup v'$;
        end if
      end for
      if $|\mathcal{I}| = i$ **then**
        Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
        $\mathbf{c}[\mathcal{I}] = c$, $\mathcal{C} = \mathcal{C} \cup c$;
        $\mathcal{G}_i = \mathcal{G}_i \setminus \mathcal{I}$;
      end if
    end if
  end for
  for all $v \in \mathcal{G}_i$ with v randomly picked from $\mathcal{W}_1 \subset \mathcal{G}_i$ **do**
    $\mathcal{I} = \{v\}$;
    $\mathcal{Q}_i = \mathcal{G}_i \setminus \mathcal{I}$;
    for all $v' \in \mathcal{Q}_i$ with v' randomly picked from $\mathcal{W}_2 \subset \mathcal{Q}_i$. **do**
      if $\{\mathcal{K}_{v'} \supset \mathcal{K}_v\ \} \cap$ {No edge between v' and $\mathcal{I}$} **then**
        $\mathcal{I} = \mathcal{I} \cup v'$;
        $\mathcal{Q}_i = \mathcal{Q}_i \setminus \{v'\}$;
      else
        $\mathcal{Q}_i = \mathcal{Q}_i \setminus \{v'\}$;
      end if
    end for
    if $|\mathcal{I}| \geq i$ **then**
      Color all the vertices in $\mathcal{I}$ by $c \notin \mathcal{C}$;
      $\mathbf{c}[\mathcal{I}] = c$, $\mathcal{C} = \mathcal{C} \cup c$;
      $\mathcal{G}_i = \mathcal{G}_i \setminus \mathcal{I}$;
    else
      $\mathcal{G}_i = \mathcal{G}_i \setminus \{v\}$, $\mathcal{G}_{i-1} = \mathcal{G}_{i-1} \cup \{v\}$;
    end if
  end for
end for
$\mathbf{c}$ =LocalSearch($\mathcal{H}_{\mathbf{C},\mathbf{W}}, \mathbf{c}, \mathcal{C}$);
return the local coloring number $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$ and the corresponding color assignment $\mathbf{c}(\mathcal{N}^+(v))$ for each v;

Remark 2. *Please note that RAP-GLC goes through the same Step I as RAP-HgLC, and then simply assigns a different color to each remaining uncolored vertex. On the other hand, Step II in RAP-HgLC tries to find further independent sets among the remaining uncolored vertices. It is this extra step that guarantees the performance of RAP-HgLC to be no worse than that of RAP-GLC.*

The RAP-HgLC algorithm, when operating on the i-th layer, *always* colors at least i vertices with the same color. Please note that if there are remaining vertices when reaching layer 1, all such vertices will be colored, each with a different color.

To further reduce the required number of colors, we use a function called LocalSearch (Algorithm 2, Line 51), which is described in Algorithm 3. It works in an iterative fashion by replacing the current solution with a better one if there exists. It terminates when no better solutions can be found. In particular, the local search algorithm has the purpose of checking the redundancy of each color $c \in \mathcal{C}$, to eventually decrease the current objective function value $|\mathcal{C}|$. In more detail, the local search computes, iteratively for each color $c \in \mathcal{C}$, the set $\mathcal{J}_c$ of all vertices colored with color c, and performs the following steps:

1. For each vertex $i \in \mathcal{J}_c$, if there is a color $c' \in \mathcal{C}$, $c' \neq c$ that is not assigned to any adjacent vertex $j \in Adj(i)$, then assign vertex i with color c';
2. Color c is removed from the set $\mathcal{C}$ if and only if in the previous step it has been possible to replace c with some color $c' \neq c$ for all vertices in $\mathcal{J}_c$.

Finally, in Algorithm 2, Line 52, we compute the local coloring number.

Algorithm 3 LocalSearch($\mathcal{H}_{\mathbf{C},\mathbf{W}}, \mathbf{c}, \mathcal{C}$)

for all $c \in \mathcal{C}$ **do**
  Let $\mathcal{J}_c$ be the set of vertices whose color is c;
  Let $\mathcal{B} = \emptyset$;
  Let $\hat{\mathbf{c}} = \mathbf{c}$;
  for all $i \in \mathcal{J}_c$ **do**
    $\mathcal{A} = \emptyset$;
    for all $j \in \mathcal{N}(i)$ **do**
      $\mathcal{A} = \mathcal{A} \cup \mathbf{c}[j]$;
      if $\mathcal{C} \setminus \mathcal{A} \neq \emptyset$ **then**
        c' is chosen uniformly at random from $\mathcal{C} \setminus \mathcal{A}$;
        $\hat{\mathbf{c}}[i] = c'$;
        $\mathcal{B} = \mathcal{B} \cup \{i\}$;
      end if
    end for
    if $|\mathcal{B}| = |\mathcal{J}_c|$ **then**
      $\mathbf{c} = \hat{\mathbf{c}}$;
      $\mathcal{C} = \mathcal{C} \setminus c$;
    end if
  end for
end for
return $\mathbf{c}$;

To illustrate the RAP-HgLC algorithm, we present the following example.

Example 3. *Consider a shared link network with* $n = 3$ *users:* $\mathcal{U} = \{1, 2, 3\}$, *and* $m = 3$ *files:* $\mathcal{F} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. *Each file is partitioned into* 4 *packets. For example,* $\mathrm{A} = \{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4\}$. *For the caching part, let user* 1 *cache* $\{\mathrm{A}_1, \mathrm{B}_1, \mathrm{B}_2, \mathrm{B}_3, \mathrm{B}_4, \mathrm{C}_2\}$, *user* 2 *cache* $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4, \mathrm{B}_1, \mathrm{C}_2, \mathrm{C}_3\}$, *user* 3 *cache* $\{\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \mathrm{C}_1, \mathrm{B}_2, \mathrm{B}_3\}$. *Then, let users* $\{1, 2, 3\}$ *request files* $\{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$ *respectively. Equivalently, user* 1 *requests* $\mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4$; *user* 2

requests $\mathrm{B}_2, \mathrm{B}_3, \mathrm{B}_4$*; user* 3 *requests* $\mathrm{C}_2, \mathrm{C}_3, \mathrm{C}_4$*. Then, we have* $\mathcal{K}_{\mathrm{A}_2} = \{1, 2\ 3\}$*;* $\mathcal{K}_{\mathrm{A}_3} = \{1, 2\ 3\}$*;* $\mathcal{K}_{\mathrm{A}_4} = \{1, 2\}$*;* $\mathcal{K}_{\mathrm{B}_2} = \{2, 1\ 3\}$*;* $\mathcal{K}_{\mathrm{B}_2} = \{2, 1\ 3\}$*;* $\mathcal{K}_{\mathrm{B}_4} = \{2, 1\}$*;* $\mathcal{K}_{\mathrm{C}_2} = \{3, 1\ 2\}$*;* $\mathcal{K}_{\mathrm{C}_3} = \{3, 2\}$*;* $\mathcal{K}_{\mathrm{C}_4} = \{3\}$ *(here* C_4 *is requested by user* 3 *and not cached anywhere).*

The RAP-HgLC algorithm works as follows. For $i = n = 3$*,* $\mathcal{G}_3 = \{\mathrm{A}_2, \mathrm{A}_3, \mathrm{B}_2, \mathrm{B}_3, \mathrm{C}_2\}$*, let* $v = \mathrm{A}_2$*, then it can be found that* B_2 *and* C_2 *would be in* $\mathcal{I}$*, hence* $\mathcal{I} = \{\mathrm{A}_2, \mathrm{B}_2, \mathrm{C}_2\}$*. Now since* $|\mathcal{I}| = n = 3$*, we color* $\mathrm{A}_2, \mathrm{B}_2, \mathrm{C}_2$ *by black (see Figure 4). Then* $\mathcal{G}_i = \mathcal{G}_i \setminus \mathcal{I} = \{\mathrm{A}_3, \mathrm{B}_3\}$*. In the following loop, since we cannot find a set* $\mathcal{I}$ *with* $|\mathcal{I}| = n = 3$*, we move to Line 19. Then since we cannot find a* $\mathcal{I}$ *with* $|\mathcal{I}| \geq n = 3$*, then we do* $\mathcal{G}_2 = \mathcal{G}_2 \cup \{\mathrm{A}_3\}$*, and then* $\mathcal{G}_2 = \mathcal{G}_2 \cup \{\mathrm{B}_3\}$*. Therefore, we obtain* $\mathcal{G}_2 = \{\mathrm{A}_3, \mathrm{A}_4, \mathrm{B}_3, \mathrm{B}_4, \mathrm{C}_3\}$*. Now we go to Line 5 (start next loop). For* $i = n - 1 = 2$*, in this loop, we first pick* $v = \mathrm{A}_4$*, then we can find* $\mathcal{I} = \{\mathrm{A}_4, \mathrm{B}_4\}$*. We color* $\{\mathrm{A}_4, \mathrm{B}_4\}$ *by blue (see Figure 4). Now* $\mathcal{G}_2 = \mathcal{G}_2 \setminus \{\mathrm{A}_4, \mathrm{B}_4\} = \{\mathrm{A}_3, \mathrm{B}_3, \mathrm{C}_3\}$*. Then in Line 19, we find the vertex with smallest length of* $\mathcal{K}_v$ *(let* $a = 0$*), which is* C_3 *with* $\mathcal{K}_{\mathrm{C}_3} = \{3, 2\}$*, then we have* $\mathcal{I} = \{\mathrm{C}_3\}$ *and* $\mathcal{Q}_2 = \{\mathrm{A}_3, \mathrm{B}_3\}$*, then in the next loop, we can find* $\mathcal{I} = \{\mathrm{C}_3, \mathrm{B}_3\}$*. We color* $\mathcal{I} = \{\mathrm{C}_3, \mathrm{B}_3\}$ *by red (see Figure 4). Now* $\mathcal{G}_2 = \mathcal{G}_2 \setminus \{\mathrm{C}_3, \mathrm{B}_3\} = \{\mathrm{A}_3\}$*. Since there is no* $\mathcal{I}$ *with* $|\mathcal{I}| \geq 2$*, then we do* $\mathcal{G}_1 = \mathcal{G}_1 \cup \{\mathrm{A}_3\} = \{\mathrm{C}_4, \mathrm{A}_3\}$*. Then we go to next loop* $i = n - 2 = 1$*. Then we can see that* $\mathcal{I} = \{\mathrm{C}_4\}$*, and we color* $\{\mathrm{C}_4\}$ *by purple (see Figure 4). Then* $\mathcal{G}_1 = \mathcal{G}_1 \setminus \{\mathrm{C}_4\} = \{\mathrm{A}_3\}$*. Hence, we can find* $\mathcal{I} = \{\mathrm{A}_3\}$ *and we color* $\{\mathrm{A}_3\}$ *by brown.*

According to Figure 4, the total number of required colors is 5*, while the maximum number of colors required locally by each user is* 4*. For the naive multicasting, since it only allows the vertices represented the same packet to be colored by the same color, the total number of required colors is* 9*. The corresponding rate is given by* $9/4$*. Hence, the final rate achieved by RAP-HgLC with local coloring is no more than* $\min\{4/4, 9/4\} = 1$*. For the interested reader, it can be verified that if the GCC algorithm, designed for* $B \to \infty$*, as proposed in [10], is used, the corresponding number of required colors is* 6*.*

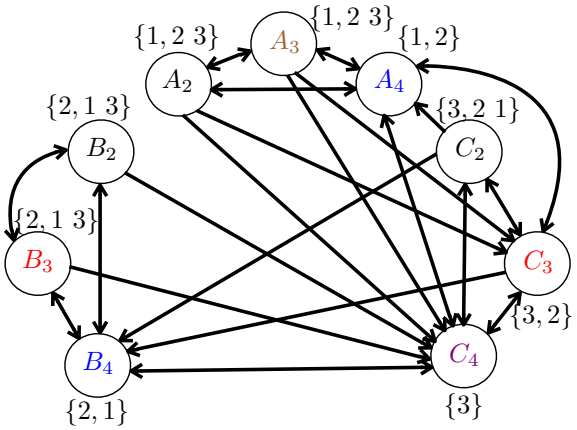

Figure 4. One example for the RAP-HgLC algorithm (this figure needs to be viewed in color).

The complexity of RAP-HgLC can be computed as follows. For the hierarchical coloring procedure (Line 5–50 in Algorithm 2), the complexity is $O(n|\mathcal{V}|^2)$, and the complexity of local search procedure is $O(|\mathcal{E}|)$. Therefore, the running time complexity of RAP-HgLC is given by $O(n|\mathcal{V}|^2 + |\mathcal{E}|) = O(n|\mathcal{V}|^2)$. Since $|\mathcal{V}| \leq nB$, the running time complexity of RAP-HgLC is $O\left(n^3B^2\right)$.

4.2.2. RAP-HgLC Performance Analysis

For the general heterogeneous network setting, tight upper bounds on the asymptotic ($B \to \infty$) average achievable rate of RAP-HgLC are quite complex to derive, even though a simple (but not necessarily tight) upper bound on the asymptotic performance can be obtained considering the asymptotic average rate of RAP-GLC (see Remark 2).

Regarding the finite-length regime, in [21] we derived a tight upper bound on the performance of RAP-HgLC for the simpler case of homogenous networks under worst-case demands. Specifically, the bound in [21], requires B to be $\tilde{O}\left(\left(\frac{m}{M}\right)^{g+2}\right)$ (where $\tilde{O}$ hides some poly log terms) to achieve a worst-case rate of at most $\frac{n}{g}$. This approximately matches a lower bound of $\tilde{O}(\left(\frac{m}{M}\right)^{g})$ derived in the same work for any coloring algorithm, showing, for the simpler homogenous network setting, the optimality of RAP-HgLC among all graph-coloring-based algorithms.

For the more complex setting where demands arise from popularity distributions and every user requests multiple files, the finite-length performance of RAP-HgLC is investigated in Section 6 via numerical analysis, where we show how the RAP-HgLC is able to recover most of the multiplicative caching gain even with very moderate packetization order.

5. Tradeoff between Number of Requests and Code Length

As mentioned earlier, in the simpler homogenous scenario, the authors in [21] showed that under worst-case demands, to achieve a gain g over conventional naive multicasting, it is necessary for B to grow exponentially with g. Intuitively, this is because a sufficiently large B is needed to create coded multicast transmissions that are useful for multiple users. However, when each user makes multiple requests, the number of requests $L_u = L$ can play a similar role to that of B, such that the requirement for B, and hence the resulting computational complexity can be reduced. For ease of analysis, in this section, we assume that all users place the same number of requests ($L_u = L$).

In the following, under either worst-case or uniform demands, we show the sufficient conditions on B and L that guarantee achieving a gain $g = \frac{Mn}{m}$. From this result, we can obtain the regime where B and L are interchangeable (L plays an equivalent role to B). Note that it can be shown that the number of file transmissions under both worst-case and uniform demands have the same order.

We consider two cases for the range of B: the case of $B = 1$, and the case of $B = \omega\left(\frac{m}{M}\right)$. The regime where $1 < B = O\left(\frac{m}{M}\right)$ is out of the scope of this paper.

When $B = 1$, the cache placement algorithm becomes scalar uniform cache placement (SUP), in which each user caches M entire files chosen uniformly at random. For simplicity, we let M be a positive integer. Then, as shown in [14], letting $L \to \infty$ as a function of n, m, M, we obtain the following theorem.

Theorem 2. *When $B = 1$ and $M = \omega(1)$, for the shared link caching network with n users, library size m, storage capacity M, and L distinct per-user requests ($nL \leq m$), if (i) $\frac{M}{m} \leq \frac{1}{2}$ and*

$$L = \omega\left(\max\left\{\frac{nM}{m}\left(\frac{m}{M}\right)^{n}\frac{1}{\left(1-\frac{M}{m}\right)}, \left(\frac{(nM)^{\frac{1}{2(1-\varepsilon)}}}{\left(\frac{m}{M}-1\right)\left(1-\left(1-\frac{M}{m}\right)^{n}\right)}\right)\right\}\right), \tag{20}$$

or (ii) $\frac{M}{m} \geq \frac{e}{1+e}$ and

$$L = \omega\left(\max\left\{\left(\frac{m}{m-M}\right)^{n}, \left(\frac{(nM)^{\frac{1}{2(1-\varepsilon)}}}{\left(\frac{m}{M}-1\right)\left(1-\left(1-\frac{M}{m}\right)^{n}\right)}\right)\right\}\right), \tag{21}$$

where ε is an arbitrarily small number,

then, the achievable rate of RAP-GLC is upper bounded by

$$\lim_{n,m\to\infty} \mathbb{P}\left(R^{\text{SUP}-\text{GLC}} \leq (1+o(1))\min\left\{L\left(\frac{m}{M}-1\right), Ln, m-M\right\}\right) = 1.$$

Proof. See Appendix B. □

From Theorem 2, we can see that when L and M are large enough, instead of requiring a large B and packet-level coding, a simpler file-level coding scheme is sufficient to achieve the same order-optimal rate. We remark, however, that the range of the parameter regimes in which this result holds is limited due to the requirement of a large M and L. Next, we focus on another parameter regime, when $B = \omega\left(\frac{m}{M}\right)$, and find the achievable tradeoff between B and L.

Theorem 3. *When $B = \omega\left(\frac{m}{M}\right)$, for the shared link caching network with n users, library size m, storage capacity M, and L distinct per-user requests ($nL \leq m$), if (i) $\frac{M}{m} \leq \frac{1}{2}$, and*

$$B = \omega\left(\max\left\{\frac{nM}{Lm}\left(\frac{m}{M}\right)^n \frac{1}{\left(1-\frac{M}{m}\right)}, \frac{(nM)^{\frac{1}{2(1-\varepsilon)}}}{L\left(\frac{m}{M}-1\right)\left(1-\left(1-\frac{M}{m}\right)^n\right)}\right\}\right), \tag{22}$$

or (ii) $\frac{M}{m} \geq \frac{e}{1+e}$, and

$$B = \omega\left(\max\left\{\frac{1}{L}\left(\frac{m}{m-M}\right)^n, \frac{(nM)^{\frac{1}{2(1-\varepsilon)}}}{L\left(\frac{m}{M}-1\right)\left(1-\left(1-\frac{M}{m}\right)^n\right)}\right\}\right), \tag{23}$$

where ε is an arbitrarily small number,

Then, the achievable rate of RAP-GLC is upper bounded by

$$\lim_{n,m\to\infty} \mathbb{P}\left(R^{\text{SUP}-\text{GLC}} \leq (1+o(1)) \min\left\{L\left(\frac{m}{M}-1\right), Ln, m-M\right\}\right) = 1. \tag{24}$$

Proof. See Appendix C. □

If we particularize Theorem 1 to the homogenous network setting under uniform demands, we see that the rate achieved by RAP-GLC is upper bounded by the same expression given in (24). Hence, from Theorem 2, we can see that when L is large enough, instead of requiring a very large B, an intermediate value of $B = \omega\left(\frac{m}{M}\right)$ is sufficient to achieve the same order-optimal rate. In practice, it is important to find the right balance and tradeoff between B and L given the remaining system parameters. In Section 6, we show via simulation that a similar tradeoff holds also for RAP-HgLC.

6. Simulations and Discussions

In this section, we numerically evaluate the performance of the two polynomial-time algorithms described in Section 4, RAP-GLC and RAP-HgLC, in the finite-length regime characterized by the number of packets per file B.

Recall that the caching distribution $\mathbf{P}^*$ is to be optimized to minimize the number of transmissions. Since the distribution $\mathbf{P}^*$ resulting from minimizing the right-hand side of (12) may not admit an analytically tractable expression in general, in the following numerical results, we restrict the caching distribution to take the form of a *truncated uniform distribution* $\widetilde{\mathbf{p}}_u$, as described in [12]:

$$\begin{aligned} \widetilde{p}_{f,u} &= \frac{1}{\widetilde{m}_u}, \quad f \leq \widetilde{m}_u \\ \widetilde{p}_{f,u} &= 0, \quad f \geq \widetilde{m}+1 \end{aligned} \tag{25}$$

where the cut-off index $\widetilde{m}_u \geq M$ is a function of the system parameters that is optimized to minimize the right-hand side of (12). The intuition behind the form of $\widetilde{\mathbf{p}}_u$ in (25) is that each user caches the same fraction of (randomly selected) packets from each of the most $\widetilde{m}_u$ popular files, and does not cache any packet from the remaining $m - \widetilde{m}_u$ least popular files. We point out that when $\widetilde{m}_u = M$, this cache placement coincides with the LFU (Least Frequently Used) caching policy. Thus, this cache placement is referred to as *Random LFU* (RLFU) [12], and the corresponding caching algorithms as RLFU-GLC and RLFU-HgLC. Recall that LFU discards the least frequently requested file upon the arrival of a new file to a full cache of size M_u files. In the long run, this is equivalent to caching the M_u most popular files [69].

In Figures 5 and 6, we plot the average achievable rate, i.e., the average number of transmissions (normalized by the file size) as a function of the cache size for RLFU-GLC and RLFU-HgLC. For comparison, we also simulate the following algorithms:

- LFU, which has been shown to be optimal in single cache networks;
- RLFU-GLC with infinite file packetization ($B \to \infty$), whose performance guarantee is given in Theorem 1, and it is shown to be order optimal.

Regarding the LFU algorithm, the average achievable rate is given by

$$\mathbb{E}[R^{\mathrm{LFU}}] = \sum_{f=\min_u\{M_u\}+1}^{m} \left(1 - \prod_{u \in \mathcal{U}_{\{M_u<f\}}} (1-q_{f,u})^{L_u}\right), \tag{26}$$

where $\mathcal{U}_{\{M_u<f\}}$ denotes the set of users with $M_u < f$.

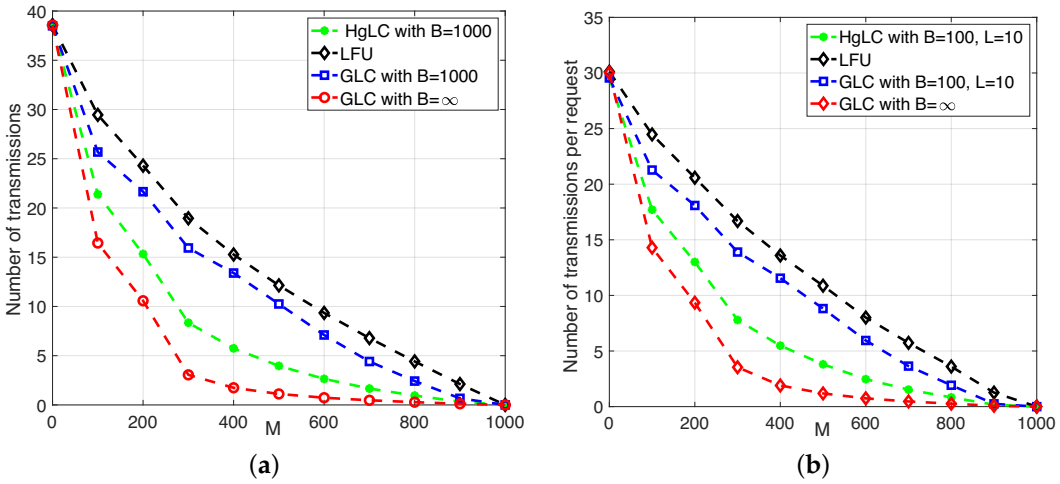

Figure 5. Average number of transmissions in a heterogeneous shared link caching network with $m = 1000$. (**a**) $n = 40$, $L = 1$, $\gamma = 0.5$; (**b**) $n = 40$, $L = 10$, $\gamma = 0.5$.

For simplicity, and to better illustrate the effectiveness of the proposed algorithms, especially under multiple per-user requests, we consider a scenario in which all users request files according to a Zipf demand distribution with parameter $\gamma \in \{0.2, 0, 4, 0.5\}$, and all caches have size M files. Under Zipf demands, file f is requested with probability $\frac{f^{-\gamma}}{\sum_{i=1}^{m} i^{-\gamma}}$.

We consider two types of users. In Figures 5a and 6a, users represent end devices requesting only one file each ($L = 1$); while in Figures 5b and 6b, they represent helpers/small-cells, each serving 10 end user devices, and consequently collecting $L = 10$ requests.

In Figure 5a,b, we fix the total number of users n and the product between L and B ($L \times B = 1000$). Figure 5a plots the average rate for a network with $n = 40$ users, $\gamma = 0.5$, $L = 1$, and $B = 1000$. It is immediate to observe the impact of finite packetization on the multiplicative caching gain. In fact, as predicted by the theory (see [21]), the significant caching gain (with respect to LFU) quantified by the asymptotic performance of RAP-GLC (GLC with $B = \infty$) is completely lost when using RAP-GLC with finite packetization (GLC with $B = 1000$). On the other hand, RAP-HgLC remarkably preserves, at the expense of a slight increase in computational complexity, most of the multiplicative caching gain for the same value of file packetization. For example, in Figure 5a, if M doubles from $M = 200$ to $M = 400$, then the rate achieved by RAP-HgLC reduces from 15 to 5.7. Furthermore, RAP-HgLC can achieve a factor of 3.5 rate reduction from LFU for $M = 500$. For the same regime, it is straightforward to verify that neither RAP-GLC nor LFU exhibit this property. Note from Figure 5a that to guarantee a rate of 10, RAP-GLC requires a cache size of $M = 500$, while RAP-HgLC can reduce the cache size requirement to $M = 250$, a $2\times$ cache size reduction. Furthermore, while LFU can only provide an additive caching gain, additive and multiplicative gains may show indistinguishable when M is comparable to the library size m. Hence, one needs to pick a reasonably small M ($\frac{m}{n} < M \ll m$) to observe the multiplicative caching gain of RAP-HgLC.

Figure 5b shows the average rate for a network with $n = 40$ helpers/small-cells, each serving 10 users making requests according to a Zip distribution with $\gamma = 0.5$. Hence, the total number of distinct requests per helper is up to $L_u = 10, \forall u \in \{1, \ldots, 20\}$. In this case, we assume $B = 100$ (instead of $B = 1000$ in Figure 5a). In order to make easier the comparison with Figure 5a, we normalize the achievable rate (number of transmissions) by the file size and the number of requests.

Note from Figure 5a,b that as predicted by Theorem 3, when L_u increases (from $L_u = 1$ to $L_u = 10$), almost the same multiplicative caching gain can be achieved with a smaller B (from $B = 1000$ to $B = 100$). In fact, from Figure 5a,b, we see that under RAP-HgLC, the average rate per request for $B = 100$ and $L = 10$ is almost the same as the average rate per request for $B = 1000$ and $L = 1$. This confirms the interesting tradeoff between B and L established in Theorem 3.

We can observe a similar behavior in Figure 6a,b. Figure 6a plots the average rate for a network with $n = 80$ users, $\gamma = 0.4$, $L = 1$, and $B = 200$. RAP-HgLC is able to preserve most of the multiplicative caching gain for the same values of file packetization. For example, in Figure 6a, if M doubles from $M = 200$ to $M = 400$, then the rate achieved by RAP-HgLC essentially halves from 20 to 10. Furthermore, RAP-HgLC can achieve a factor of 5 rate reduction from LFU for $M = 500$. Note from Figure 6a that to guarantee a rate of 20, RAP-GLC requires a cache size of $M = 500$, while RAP-HgLC can reduce the cache size requirement to $M = 200$, a $2.5\times$ cache size reduction.

Figure 6b plots the average rate for a network with $n = 20$ helpers/small-cells, each serving 10 users making requests according to a Zip distribution with $\gamma = 0.2$. Hence, the total number of distinct requests per helper is up to $L_u = 10, \forall u \in \{1, \ldots, 20\}$. In this case, we assume $B = 100$. Differently from Figure 5b, here we plot the average rate without normalizing it by the number of requests.

Note from Figure 6a,b that, as predicted by Theorem 3, when L_u increases (from $L_u = 1$ to $L_u = 10$), almost the same multiplicative caching gain can be achieved with a smaller B (from $B = 200$ to $B = 100$). In fact, from Figure 6a,b, we see that under RAP-HgLC , the average rate per request for $B = 100$ and $L = 10$ is almost the same as the average rate per request for $B = 200$ and $L = 1$. For example, for $M = 200$, $B = 100$, and $L = 10$, the per request average rate achieved by RAP-HgLC is 0.3, while for $M = 200$ and $B = 200$, is 0.25. This again confirms the tradeoff between B and L stated in Theorem 3.

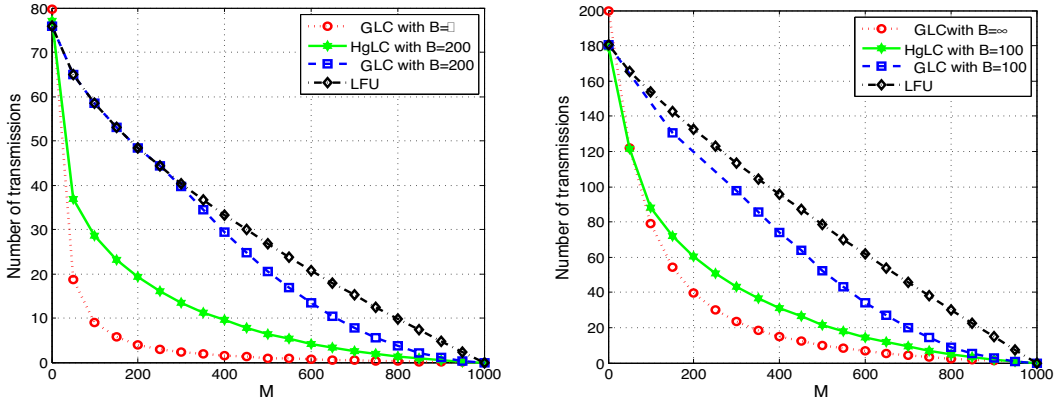

Figure 6. Average number of transmissions in a heterogeneous shared link caching network with $m = 1000$. (**a**) $n = 80$, $L = 1$, $\gamma = 0.4$; (**b**) $n = 20$, $L = 10$, $\gamma = 0.2$.

Furthermore, from Figures 5 and 6, we notice that increasing the Zip parameter reduces the gains with respect to LFU. This is explained by the fact that when aggregating multiple requests, there is a higher number of overlapping requests, which increases the opportunities for naive multicasting (as clearly characterized in [13]). Note, however, that RAP-HgLC can remarkably keep similar gains with respect to LFU in this multiple-request setting, and approach the asymptotic performance even with just $B = 100$ packets per file, confirming the effectiveness of the local graph coloring and extra processing procedures in RAP-HgLC.

7. Conclusions

Coded multicasting has been shown to be a promising approach to significantly reduce the traffic load in wireless caching networks. However, most existing schemes require the number of packets per file to grow exponentially with the number of users. To address this challenge, in this paper we focused on a heterogeneous shared link caching network model and designed novel coded multicast algorithms based on local graph coloring that exhibit polynomial-time complexity in all the system parameters, and preserve the asymptotically proven multiplicative caching gain for finite file packetization. We also demonstrated that the number of packets per file can be traded-off with the number of requests collected by each cache, such that the same multiplicative caching gain can be preserved. Simulation results confirm the superiority of the proposed schemes and illustrate the tradeoff between request aggregation and computational complexity (driven by the packetization order), shedding light into the practical achievability of the promising multiplicative caching gain in next generation wireless networks.

Author Contributions: All authors have contributed in equal part to the results of this paper.

Funding: This research was funded in part by NSF grants #1619129, #1817154, #1824558, and by the Alexander von Humboldt Professorship.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Theorem 1

To analytically characterize the performance of RAP-GLC, we consider two specific cases, where $t = n$ (i.e., the coded multicast only scheme) and $t = 0$ (i.e., the naive multicasting only scheme), and refer to these schemes as RAP-GLC$_1$ and RAP-GLC$_2$, respectively. In the following, we will compute the performance of

these two cases respectively and take the minimum rate between these two cases. Obviously, this rate can serve as an upper bound of RAP-GLC.

Appendix A.1. Average Total Number of Colors for RAP-GLC$_1$

To compute the average total number of colors provided by RAP-GLC$_1$, we first see that for all $v \in \mathcal{I}$ obtained in this algorithm, $\mathcal{T}_v$ are identical. Based on Algorithm 1, by construction the independent sets $\mathcal{I} \subset \mathcal{V}$ generated by RAP-GLC$_1$ have the same (unordered) label of users requesting or caching the packets $\{\rho(v) : v \in \mathcal{I}\}$. We shall refer to such unordered label of users as the user label of the independent set. Hence, we count the independent sets by enumerating all possible user labels, and upperbounding how many independent sets $\mathcal{I}$ Algorithm 1 generates for each user label. Consider a user label $\mathcal{U}^\ell \subset \mathcal{U}$ of size ℓ, and let $\mathcal{I}(\mathcal{U}^\ell, \mathbf{f}, i)$ the i-th independent set generated by Algorithm 1 with label $\mathcal{U}^\ell$ and while let $\mathcal{J}(\mathcal{U}^\ell, \mathbf{f}) = \{\mathcal{I}(\mathcal{U}^\ell, \mathbf{f}, i) : \forall i\}$.

Following Algorithm 1, for each $\mathcal{U}^\ell$, the number of used colors is $|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|$. Given $\mathbf{f}$, we can see that $|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|$ is a random variable which is a function of $\mathbf{C}$. Let the indicator $1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}$ denote the event that vertex v_{f_u} from file f_u requested by user $u \in \mathcal{U}^\ell$ is available in all the users in $\mathcal{U}^\ell$ but u and the rest of the vertices $\mathcal{U} \setminus \mathcal{U}^\ell$, then $1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}$ follows a Bernoulli distribution with parameter

$$\lambda(u, f_u) \quad = \quad (1 - p_{f_u,u} M_u) \prod_{k \in \mathcal{U}^\ell \setminus \{u\}} (p_{f_u,k} M_k) \prod_{k \in \mathcal{U} \setminus \mathcal{U}^\ell} (1 - p_{f_u,k} M_k) \tag{A1}$$

such that its expectation is $\lambda(u, f_u)$. Then, we can see that given f, $\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\} = \lambda(u, f_u)B + o(B)$ with high probability [70]. Thus, as $B \to \infty$, we have that with high probability,

$$\begin{aligned} |\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})| &= \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\} \\ &= \max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \quad \lambda(u, f_u)B + o(B), \end{aligned} \tag{A2}$$

where $\mathbf{f}(\mathcal{U}^\ell)$ represent the set of files requested by $\mathcal{U}^\ell$.

Then, by averaging over the demand's distribution, we obtain that with high probability:

$$\begin{aligned} \mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})] &\leq \mathbb{E}\left[\sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \left|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})\right|\right] \\ &= \sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \mathbb{E}\left[\left|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})\right|\right] \\ &\overset{(a)}{=} \sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \mathbb{E}\left[\max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \lambda(u, f_u)B + o(B)\right] \\ &\overset{(b)}{=} \sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} \lambda(u, f) + \delta_1(B), \end{aligned} \tag{A3}$$

where (a) is by using (A2) and (b) is obtained by computing the probability that the requested file f_u in $\mathbf{f}(\mathcal{U}^\ell)$ maximizes $\lambda(u, f)$. $\delta_1(B)$ denotes a smaller order term of $\sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} \lambda(u, f)$. For any $\mathcal{U}^\ell$, we obtain that $\sum_f \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} = 1$, and $\rho_{f,u,\mathcal{U}^\ell}$ denotes the probability that file f is the file

with memory assignment $p_{f,u}$ such that $\rho_{f,u,\mathcal{U}^\ell} \triangleq \mathbb{P}(f = \arg\max_{f_u \in \mathbf{f}(\mathcal{U}^\ell)} \lambda(u, f_u)$, where $\mathbf{f}(\mathcal{U}^\ell)$ denotes the set of files requested by a subset users $\mathcal{U}^\ell$. Thus, we normalize (A3) by B and obtain that

$$\begin{aligned} R^{\text{RAP}-\text{GLC}_1} &= \frac{\mathbb{E}[\chi(\mathsf{H}_{\mathsf{M},\mathsf{W}})]}{B} \\ &\leq \sum_{j=1}^{L} \sum_{\ell=1}^{n} \sum_{\mathcal{U}^\ell \subset \mathcal{U}_{n_j}} \sum_{f=1}^{m} \sum_{u \in \mathcal{U}^\ell} \rho_{f,u,\mathcal{U}^\ell} \lambda(u, f), \\ &= \psi(\mathbf{P}, \mathbf{Q}), \end{aligned} \tag{A4}$$

which is the first term inside the minimum in (12).

Appendix A.2. Average Total Number of Colors for RAP-GLC$_2$

As described in Section 4.1, RAP-GLC$_2$ computes the minimum coloring of $\mathcal{H}_{\mathbf{C},\mathbf{W}}$ subject to the constraint that only the vertices representing the same packet can have the same color. In this case, the total number of colors is equal to the number of distinct requested packets, and the coloring can be found in $O(|\mathcal{V}|^2)$. Starting from this valid coloring, GCLC$_2$ computes $\max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))|$. To show that the performance of GCLC$_2$ are upper bounded by $\bar{m} - \bar{M}$ with $\bar{m}$ and $\bar{M}$ given as in (13) and (14) respectively, we note that:

$$\begin{aligned} \max_{v \in \mathcal{V}} |\mathbf{c}(\mathcal{N}^+(v))| &\overset{(a)}{\leq} \sum_{\mathbf{f} \in \mathcal{F}^n} \left(\prod_{u=1}^{n} q_{f,u} \right) \sum_{\hat{f}=1}^{m} 1\{\hat{f} \in \mathbf{f}\} B_f \\ &= \sum_{\hat{f}=1}^{m} \sum_{\mathbf{f} \in \mathcal{F}^n} \left(\prod_{u=1}^{n} q_{f,u} \right) 1\{\hat{f} \in \mathbf{f}\} B_{\hat{f},\mathbf{f}} \\ &\overset{(b)}{\leq} \sum_{f} \mathbb{P}(\text{file } f \text{ is requested})(B - \min_u p_{u,f} M_u B) \\ &= \sum_{f=1}^{m} \left(1 - \prod_{u=1}^{n} \left(1 - q_{f,u} \right)^{L_u} \right) (1 - \min_u p_{u,f} M_u) B, \end{aligned} \tag{A5}$$

where $B_{\hat{f},\mathbf{f}}$ is number of chucks that are going to be transmitted of file $\hat{f}$ given that the demand vector is equal to $\mathbf{f}$, and (a) is due to the observation that given a file $\hat{f}$,

$$\begin{aligned} \sum_{\mathbf{f} \in \mathcal{F}^n} \left(\prod_{u=1}^{n} q_{f,u} \right) 1\{\hat{f} \in \mathbf{f}\} &= \mathbb{E}[1\{\hat{f} \in \mathbf{f}\}] \\ &= \mathbb{P}(\text{file } \hat{f} \text{ is requested}). \end{aligned} \tag{A6}$$

Normalizing (A5) by B, we obtain that:

$$\begin{aligned} R^{\text{RAP}-\text{GLC}_2} &\leq \sum_{f=1}^{m} \left(1 - \prod_{u=1}^{n} \left(1 - q_{f,u} \right)^{L_u} \right) \\ &\quad \cdot (1 - \min_u p_{u,f} M_u) \\ &= \bar{m} - \bar{M}, \end{aligned} \tag{A7}$$

which is the second term inside the minimum in (12).

Appendix B. Proof of Theorem 2

In this section, we prove Theorem 2. Recall that for each $\mathcal{U}^\ell$, the number of used colors is given by $|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|$, then we have we have $|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})| = \max_{u\in\mathcal{U}^\ell} \sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}$, where $\mathbf{f}(\mathcal{U}^\ell)$ represent the set of files requested by $\mathcal{U}^\ell$. In this case, it is clear that

$$\mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right] = L\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}. \tag{A8}$$

Then let

$$Y_l \triangleq |\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})| - \mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right]. \tag{A9}$$

The goal is to find a condition of L such that

$$|\mathbb{E}[Y_l]| = o\left(L\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right), \tag{A10}$$

which implies

$$\mathbb{E}\left[|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|\right] \leq (1+o(1))L\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}. \tag{A11}$$

Then following the similar step from Theorem 1 in [12], we can obtain the concentration result shown in Theorem 2, where we require $L = \omega\left(\frac{(nM)^{\frac{1}{2(1-\epsilon)}}}{\left(\frac{m}{M}-1\right)\left(1-\left(1-\frac{M}{m}\right)^n\right)}\right)$ and $\epsilon > 0$ is an arbitrarily small number.

To compute $|\mathbb{E}[Y_l]|$, we have

$$\begin{aligned}
(\mathbb{E}[Y_l])^2 &\leq \mathbb{E}[(Y_l)^2] \\
&= \mathbb{E}\left[\left(|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})| - \mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right]\right)^2\right] \\
&= \mathbb{E}\left[\left(\max_{u\in\mathcal{U}^\ell}\left(\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\} - \mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right]\right)\right)^2\right] \\
&\leq \sum_{u\in\mathcal{U}^\ell} \mathbb{E}\left[\left(\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\} - \mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right]\right)^2\right] \\
&= l\mathbb{E}\left[\left(\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\} - \mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right]\right)^2\right] \\
&\overset{(a)}{=} lL\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right) + \delta_1,
\end{aligned} \tag{A12}$$

where (a) is because $M = \omega(1)$ such that δ_1 is a smaller order term compared to the first term in (A12) and use the variance for the Binomial distribution. Then, we let

$$\left(lL\left(\tfrac{M}{m}\right)^{l-1}\left(1-\tfrac{M}{m}\right)^{n-l+1}\left(1-\left(\tfrac{M}{m}\right)^{l-1}\left(1-\tfrac{M}{m}\right)^{n-l+1}\right)\right)^{\frac{1}{2}} = o\left(L\left(\tfrac{M}{m}\right)^{l-1}\left(1-\tfrac{M}{m}\right)^{n-l+1}\right), \tag{A13}$$

then, we can obtain

$$\begin{aligned} L &= \omega\left(\frac{l\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right)}{\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}}\right) \\ &= \omega\left(\frac{\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right)}{\frac{1}{l}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}}\right). \end{aligned} \tag{A14}$$

For sufficient condition, let

$$L = \omega\left(\frac{1}{\frac{1}{l}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}}\right). \tag{A15}$$

Do the derivative of $s(l) = \frac{1}{l}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}$ with respect to l, we obtain

$$\begin{aligned} \frac{ds(l)}{dl} = \frac{1}{l}\Bigg(&\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\log\frac{M}{m} \\ &-\log\left(1-\frac{M}{m}\right)\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\Bigg) \\ &-\frac{1}{l^2}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1} \\ = \frac{1}{l}&\log\left(\frac{\frac{M}{m}}{1-\frac{M}{m}}\right)\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1} \\ &-\frac{1}{l^2}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}. \end{aligned} \tag{A16}$$

- When $\frac{M}{m} < \frac{1}{2}$, then we have $\frac{ds(l)}{l} < 0$. Hence, $s(l)$ is a decreasing function such that the minimum value of $s(l)$ take place when $l = n$. Thus, by using (A15), we obtain the sufficient condition is given by

$$L = \omega\left(\frac{n\left(\frac{m}{M}\right)^{n-1}}{1-\frac{M}{m}}\right) = \omega\left(\frac{nM}{m}\left(\frac{m}{M}\right)^{n}\frac{1}{\left(1-\frac{M}{m}\right)}\right). \tag{A17}$$

- When $\frac{M}{m} \geq \frac{e}{1+e}$, then we have $\frac{ds(l)}{l} > 0$. Hence, $s(l)$ is an increasing function such that the minimum value of $s(l)$ take place when $l = 1$. Thus, by using (A15) we obtain the sufficient condition is given by

$$L = \omega\left(\frac{1}{\left(1-\frac{M}{m}\right)^n}\right) = \omega\left(\left(\frac{m}{m-M}\right)^n\right). \tag{A18}$$

Thus, we finished the proof of Theorem 2.

Appendix C. Proof of Theorem 3

In this proof, we follow the similar procedure in the proof of Theorem 2 in Appendix B, and obtain

$$\mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right] = LB\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}. \tag{A19}$$

Then let

$$Y_l \triangleq |\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})| - \mathbb{E}\left[\sum_{\forall v_{f_u}} 1\{\mathcal{T}_{v_{f_u}} = \mathcal{U}^\ell\}\right]. \tag{A20}$$

The goal again is to find a condition of B and L such that

$$|\mathbb{E}[Y_l]| = o\left(LB\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right), \tag{A21}$$

which implies

$$\mathbb{E}\left[|\mathcal{J}(\mathcal{U}^\ell, \mathbf{f})|\right] \leq (1+o(1))LB\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}. \tag{A22}$$

Then following the similar step from Theorem 1 in [12], we can obtain the concentration result shown in Theorem 2, where we require $LB = \omega\left(\frac{(nM)^{\frac{1}{2(1-\varepsilon)}}}{\left(\frac{m}{M}-1\right)\left(1-\left(1-\frac{M}{m}\right)^n\right)}\right)$ and $\epsilon > 0$ is an arbitrarily small number.

Similar to (A12), to compute $|\mathbb{E}[Y_l]|$, we have

$$\begin{aligned}(\mathbb{E}[Y_l])^2 &\leq \mathbb{E}[(Y_l)^2] \\ &\overset{(a)}{=} lLB\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\cdot\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right)+\delta_2,\end{aligned} \tag{A23}$$

where (a) is because $B = \omega\left(\frac{m}{M}\right)$ such that δ_2 is a smaller order term compared to the first term in (A23) and use the variance for the Binomial distribution. Then, we let

$$\begin{aligned}&\left(lLB\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right)\right)^{\frac{1}{2}} \\ &= o\left(LB\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right),\end{aligned} \tag{A24}$$

then, we can obtain

$$\begin{aligned} LB &= \frac{\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right)}{\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}} \\ &= \frac{\left(1-\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}\right)}{\frac{1}{l}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}}. \end{aligned} \tag{A25}$$

For sufficient condition, let

$$LB = \omega\left(\frac{1}{\frac{1}{l}\left(\frac{M}{m}\right)^{l-1}\left(1-\frac{M}{m}\right)^{n-l+1}}\right). \tag{A26}$$

Then following the similar steps as the proof of Theorem 2 in Appendix B, we fished the proof of Theorem 3.

References

1. Shanmugam, K.; Golrezaei, N.; Dimakis, A.; Molisch, A.; Caire, G. FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers. *IEEE Trans. Inf. Theory* **2013**, *59*, 8402–8413. [CrossRef]
2. Llorca, J.; Tulino, A.; Guan, K.; Kilper, D. Network-Coded Caching-Aided Multicast for Efficient Content Delivery. In Proceedings of the 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013.
3. Fadlallah, Y.; Tulino, A.M.; Barone, D.; Vettigli, G.; Llorca, J.; Gorce, J. Coding for Caching in 5G Networks. *IEEE Commun. Mag.* **2017**, *55*, 106–113. [CrossRef]
4. Liu, D.; Chen, B.; Yang, C.; Molisch, A.F. Caching at the wireless edge: Design aspects, challenges, and future directions. *IEEE Commun. Mag.* **2016**, *54*, 22–28. [CrossRef]
5. Tandon, R.; Simeone, O. Harnessing cloud and edge synergies: Toward an information theory of fog radio access networks. *IEEE Commun. Mag.* **2016**, *54*, 44–50. [CrossRef]
6. Maddah-Ali, M.A.; Niesen, U. Coding for caching: Fundamental limits and practical challenges. *IEEE Commun. Mag.* **2016**, *54*, 23–29. [CrossRef]
7. Paschos, G.; Bastug, E.; Land, I.; Caire, G.; Debbah, M. Wireless caching: Technical misconceptions and business barriers. *IEEE Commun. Mag.* **2016**, *54*, 16–22. [CrossRef]
8. Maddah-Ali, M.; Niesen, U. Fundamental Limits of Caching. *IEEE Trans. Inf. Theory* **2014**, *60*, 2856–2867. [CrossRef]
9. Maddah-Ali, M.; Niesen, U. Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff. *IEEE/ACM Trans. Netw.* **2014**. [CrossRef]
10. Ji, M.; Tulino, A.; Llorca, J.; Caire, G. On the average performance of caching and coded multicasting with random demands. In Proceedings of the 2014 11th International Symposium on Wireless Communications Systems (ISWCS), Barcelona, Spain, 26–29 August 2014; pp. 922–926.
11. Niesen, U.; Maddah-Ali, M.A. Coded Caching With Nonuniform Demands. *IEEE Trans. Inf. Theory* **2017**, *63*, 1146–1158. [CrossRef]
12. Ji, M.; Tulino, A.M.; Llorca, J.; Caire, G. Order-Optimal Rate of Caching and Coded Multicasting with Random Demands. *IEEE Trans. Inf. Theory* **2017**, *63*, 3923–3949. [CrossRef]

13. Ji, M.; Tulino, A.; Llorca, J.; Caire, G. Caching and Coded Multicasting: Multiple Groupcast Index Coding. In Proceedings of the 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Atlanta, GA, USA, 3–5 December 2014; pp. 881–885.
14. Ji, M.; Tulino, A.M.; Llorca, J.; Caire, G. Caching-Aided Coded Multicasting with Multiple Random Requests. In Proceedings of the 2015 IEEE Information Theory Workshop (ITW), Jerusalem, Israel, 26 April–1 May 2015; pp. 1–5.
15. Wan, K.; Tuninetti, D.; Piantanida, P. On caching with more users than files. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 135–139. [CrossRef]
16. Wan, K.; Tuninetti, D.; Piantanida, P. On the optimality of uncoded cache placement. In Proceedings of the 2016 IEEE Information Theory Workshop (ITW), Cambridge, UK, 11–14 September 2016; pp. 161–165. [CrossRef]
17. Ji, M.; Caire, G.; Molisch, A.F. The Throughput-Outage Tradeoff of Wireless One-Hop Caching Networks. *IEEE Trans. Inf. Theory* **2015**, *61*, 6833–6859. [CrossRef]
18. Ji, M.; Caire, G.; Molisch, A.F. Fundamental Limits of Caching in Wireless D2D Networks. *IEEE Trans. Inf. Theory* **2016**, *62*, 849–869. [CrossRef]
19. Ji, M.; Caire, G.; Molisch, A.F. Wireless Device-to-Device Caching Networks: Basic Principles and System Performance. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 176–189. [CrossRef]
20. Cacciapuoti, A.S.; Caleffi, M.; Ji, M.; Llorca, J.; Tulino, A.M. Speeding Up Future Video Distribution via Channel-Aware Caching-Aided Coded Multicast. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 2207–2218. [CrossRef]
21. Shanmugam, K.; Ji, M.; Tulino, A.M.; Llorca, J.; Dimakis, A.G. Finite-Length Analysis of Caching-Aided Coded Multicasting. *IEEE Trans. Inf. Theory* **2016**, *62*, 5524–5537. [CrossRef]
22. Shangguan, C.; Zhang, Y.; Ge, G. Centralized Coded Caching Schemes: A Hypergraph Theoretical Approach. *IEEE Trans. Inf. Theory* **2018**, *64*, 5755–5766. [CrossRef]
23. Chen, Z. Fundamental limits of caching: Improved bounds for users with small buffers. *IET Commun.* **2016**, *10*, 2315–2318. [CrossRef]
24. Karamchandani, N.; Niesen, U.; Maddah-Ali, M.A.; Diggavi, S.N. Hierarchical Coded Caching. *IEEE Trans. Inf. Theory* **2016**, *62*, 3212–3229. [CrossRef]
25. Pedarsani, R.; Maddah-Ali, M.A.; Niesen, U. Online Coded Caching. *IEEE/ACM Trans. Netw.* **2016**, *24*, 836–845. [CrossRef]
26. Sahraei, S.; Gastpar, M. K users caching two files: An improved achievable rate. In Proceedings of the 2016 Annual Conference on Information Science and Systems (CISS), Princeton, NJ, USA, 16–18 March 2016; pp. 620–624. [CrossRef]
27. Wang, C.; Lim, S.H.; Gastpar, M. Information-Theoretic Caching: Sequential Coding for Computing. *IEEE Trans. Inf. Theory* **2016**, *62*, 6393–6406. [CrossRef]
28. G., J. Fundamental limits of caching: Improved bounds with coded prefetching. *arXiv* **2016**, arXiv:1612.09071.
29. Shariatpanahi, S.P.; Motahari, S.A.; Khalaj, B.H. Multi-Server Coded Caching. *IEEE Trans. Inf. Theory* **2016**, *62*, 7253–7271. [CrossRef]
30. Shanmugam, K.; Tulino, A.M.; Dimakis, A.G. Coded caching with linear subpacketization is possible using Ruzsa-Szeméredi graphs. In Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, 25–30 June 2017; pp. 1237–1241. [CrossRef]
31. Ghasemi, H.; Ramamoorthy, A. Improved Lower Bounds for Coded Caching. *IEEE Trans. Inf. Theory* **2017**, *63*, 4388–4413. [CrossRef]
32. Lim, S.H.; Wang, C.; Gastpar, M. Information-Theoretic Caching: The Multi-User Case. *IEEE Trans. Inf. Theory* **2017**, *63*, 7018–7037. [CrossRef]
33. Jeon, S.; Hong, S.; Ji, M.; Caire, G.; Molisch, A.F. Wireless Multihop Device-to-Device Caching Networks. *IEEE Trans. Inf. Theory* **2017**, *63*, 1662–1676. [CrossRef]
34. Sengupta, A.; Tandon, R. Improved Approximation of Storage-Rate Tradeoff for Caching With Multiple Demands. *IEEE Trans. Commun.* **2017**, *65*, 1940–1955. [CrossRef]
35. Hachem, J.; Karamchandani, N.; Diggavi, S.N. Coded Caching for Multi-level Popularity and Access. *IEEE Trans. Inf. Theory* **2017**, *63*, 3108–3141. [CrossRef]

36. Ji, M.; Wong, M.F.; Tulino, A.M.; Llorca, J.; Caire, G.; Effros, M.; Langberg, M. On the fundamental limits of caching in combination networks. In Proceedings of the 2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Stockholm, Sweden, 28 June–1 July 2015; pp. 695–699. [CrossRef]
37. Ji, M.; Tulino, A.M.; Llorca, J.; Caire, G. Caching in combination networks. In Proceedings of the 2015 49th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 8 November 2015; pp. 1269–1273. [CrossRef]
38. Wan, K.; Ji, M.; Piantanida, P.; Tuninetti, D. Novel outer bounds for combination networks with end-user-caches. In Proceedings of the 2017 IEEE Information Theory Workshop (ITW), Kaohsiung, Taiwan, 6–10 November 2017; pp. 444–448. [CrossRef]
39. Wan, K.; Tuninetti, D.; Ji, M.; Piantanida, P. State-of-the-art in cache-aided combination networks. In Proceedings of the 2017 51st Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 29 October–1 November 2017; pp. 641–645. [CrossRef]
40. Wan, K.; Ji, M.; Piantanida, P.; Tuninetti, D. Caching in Combination Networks: Novel Multicast Message Generation and Delivery by Leveraging the Network Topology. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
41. Wan, K.; Jit, M.; Piantanida, P.; Tuninetti, D. On the Benefits of Asymmetric Coded Cache Placement in Combination Networks with End-User Caches. In Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, 17–22 June 2018; pp. 1550–1554. [CrossRef]
42. Wan, K.; Tuninetti, D.; Ji, M.; Piantanida, P. A Novel Asymmetric Coded Placement in Combination Networks with End-User Caches. In Proceedings of the 2018 Information Theory and Applications Workshop (ITA), San Diego, CA, USA, 11–16 February 2018; pp. 1–5. [CrossRef]
43. Tian, C.; Chen, J. Caching and Delivery via Interference Elimination. *IEEE Trans. Inf. Theory* **2018**, *64*, 1548–1560. [CrossRef]
44. Yu, Q.; Maddah-Ali, M.A.; Avestimehr, A.S. The Exact Rate-Memory Tradeoff for Caching With Uncoded Prefetching. *IEEE Trans. Inf. Theory* **2018**, *64*, 1281–1296. [CrossRef]
45. Zhang, K.; Tian, C. Fundamental Limits of Coded Caching: From Uncoded Prefetching to Coded Prefetching. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1153–1164. [CrossRef]
46. Wang, C.; Bidokhti, S.S.; Wigger, M. Improved Converses and Gap Results for Coded Caching. *IEEE Trans. Inf. Theory* **2018**, *64*, 7051–7062. [CrossRef]
47. Yu, Q.; Maddah-Ali, M.A.; Avestimehr, A.S. Characterizing the Rate-Memory Tradeoff in Cache Networks Within a Factor of 2. *IEEE Trans. Inf. Theory* **2019**, *65*, 647–663. [CrossRef]
48. Karat, N.S.; Thomas, A.; Rajan, B.S. Optimal Error Correcting Delivery Scheme for an Optimal Coded Caching Scheme with Small Buffers. In Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, 17–22 June 2018; pp. 1710–1714. [CrossRef]
49. Tian, C. Symmetry, Outer Bounds, and Code Constructions: A Computer-Aided Investigation on the Fundamental Limits of Caching. *Entropy* **2018**, *20*, 603. [CrossRef]
50. Cisco. *The Zettabyte Era-Trends and Analysis*; Cisco White Paper; Cisco: San Jose, CA, USA, 2013.
51. Birk, Y.; Kol, T. Informed-source coding-on-demand (ISCOD) over broadcast channels. In Proceedings of the Conference on Computer Communications, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Gateway to the 21st Century, San Francisco, CA, USA, 29 March–2 April 1998.
52. Breslau, L.; Cao, P.; Fan, L.; Phillips, G.; Shenker, S. Web caching and Zipf-like distributions: Evidence and implications. In Proceedings of the INFOCOM'99: Conference on Computer Communications, New York, NY, USA, 21–25 March 1999; Volume 1, pp. 126–134.
53. Tang, L.; Ramamoorthy, A. Coded Caching Schemes With Reduced Subpacketization From Linear Block Codes. *IEEE Trans. Inf. Theory* **2018**, *64*, 3099–3120. [CrossRef]
54. Yan, Q.; Cheng, M.; Tang, X.; Chen, Q. On the Placement Delivery Array Design for Centralized Coded Caching Scheme. *IEEE Trans. Inf. Theory* **2017**, *63*, 5821–5833. [CrossRef]

55. Vettigli, G.; Ji, M.; Tulino, A.M.; Llorca, J.; Festa, P. An efficient coded multicasting scheme preserving the multiplicative caching gain. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 251–256. [CrossRef]
56. Ji, M.; Shanmugam, K.; Vettigli, G.; Llorca, J.; Tulino, A.M.; Caire, G. An efficient multiple-groupcast coded multicasting scheme for finite fractional caching. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 3801–3806. [CrossRef]
57. Ramakrishnan, A.; Westphal, C.; Markopoulou, A. An Efficient Delivery Scheme for Coded Caching. In Proceedings of the 2015 27th International Teletraffic Congress, Ghent, Belgium, 8–10 September 2015; pp. 46–54. [CrossRef]
58. Jin, S.; Cui, Y.; Liu, H.; Caire, G. Order-Optimal Decentralized Coded Caching Schemes with Good Performance in Finite File Size Regime. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–7. [CrossRef]
59. Wan, K.; Tuninetti, D.; Piantanida, P. Novel delivery schemes for decentralized coded caching in the finite file size regime. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 21–25 May 2017; pp. 1183–1188. [CrossRef]
60. Asghari, S.M.; Ouyang, Y.; Nayyar, A.; Avestimehr, A.S. Optimal Coded Multicast in Cache Networks with Arbitrary Content Placement. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
61. Amiri, M.M.; Yang, Q.; Gündüz, D. Decentralized coded caching with distinct cache capacities. In Proceedings of the 2016 50th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 6–9 November 2016; pp. 734–738. [CrossRef]
62. Amiri, M.M.; Yang, Q.; Gündüz, D. Decentralized Caching and Coded Delivery With Distinct Cache Capacities. *IEEE Trans. Commun.* **2017**, *65*, 4657–4669. [CrossRef]
63. Ibrahim, A.M.; Zewail, A.A.; Yener, A. Centralized Coded Caching with Heterogeneous Cache Sizes. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6. [CrossRef]
64. Wei, Y.; Ulukus, S. Coded caching with multiple file requests. In Proceedings of the 2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 3–6 October 2017; pp. 437–442. [CrossRef]
65. Parrinello, E.; Unsal, A.; Elia, P. Fundamental Limits of Caching in Heterogeneous Networks with Uncoded Prefetching. *arXiv* **2018**, arXiv:1811.06247.
66. Shanmugam, K.; Dimakis, A.G.; Langberg, M. Local graph coloring and index coding. In Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, 7–12 July 2013; pp. 1152–1156. [CrossRef]
67. Lin, S.; Costello, D.J. *Error Control Coding*; Prentice-hall Englewood Cliffs: Upper Saddle River, NJ, USA, 2004; Volume 123.
68. Bar-Yossef, Z.; Birk, Y.; Jayram, T.; Kol, T. Index coding with side information. *IEEE Trans. Inf. Theory* **2011**, *57*, 1479–1494. [CrossRef]
69. Lee, D.; Noh, S.; Min, S.; Choi, J.; Kim, J.; Cho, Y.; Kim, C. LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Trans. Comput.* **2001**, *50*, 1352–1361.
70. Boucheron, S.; Lugosi, G.; Massart, P. *Concentration Inequalities: A Nonasymptotic Theory of Independence*; Oxford University Press: Oxford, UK, 2013.