Joint Optimization of Storage and Transmission via Coding Traffic Flows for Content Distribution

Derya Malak*, Yuanyuan Li†, Stratis Ioannidis†, Edmund M. Yeh†, and Muriel Médard‡
*EURECOM, Sophia Antipolis, FRANCE, derya.malak@eurecom.fr
†Northeastern University, Boston, MA, USA, {yuanyuanli,ioannidis,eyeh}@ece.neu.edu
‡RLE, MIT, Cambridge, MA, USA, medard@mit.edu

***Abstract*—We provide a flow-based coded caching framework for information centric networks. We jointly optimize delivery rates, cross coding, and cache contents allocation as a function of demand and the network's topology. Our model accounts for storage and transmission costs, demand asymmetry, and arbitrary multi-hop topologies, and relies on an ordered flow-based decoding schedule for the transmissions created by pairwise coded flows. Through extensive experiments over multiple topologies, we observe that our coded caching scheme reduces transmission costs over competitors by several orders of magnitude.**

***Index Terms*—Coded caching, wireless, cross-coding, memory-bandwidth tradeoff, demand asymmetry, unicast, multicast.**

I. Introduction

Caching has been used extensively to alleviate backhaul capacity bottlenecks by moving the content closer to the edge. Determining the optimal placement of files for maximizing the cache hit rate or caching gain is NP-hard [1], and approximation algorithms have been studied extensively in this setting [1], [2]. Recently, joint optimization frameworks have been proposed to understand the tradeoffs between caching and routing [3], [4], computing [5], scheduling [6], and power control [7]. In this work, we leverage coding to facilitate file transmission in both wired and wireless caching networks. Exploiting coding relaxes the combinatorial structure of existing NP-hard solutions, as it inherently converts the integer optimization problems into continuous optimization problems, simultaneously eliminating the need for rounding techniques [8]. Moreover, via both caching and cross-coding, users can obtain the required number of degrees of freedom (DoF) to decode a file via fewer transmissions, providing additional resource savings.

After the publication of the landmark paper by Maddah-Ali and Niesen [9], various facets of coded caching have been investigated. These include benefits over uncoded caching [10], memory-bandwidth tradeoffs [11], cache size versus the catalog size [12], demand distribution, and multicasting and unicasting opportunities [13], to name a few. We depart from [9] and follow-up works by assuming asymmetric demand over an arbitrary topology; as a result, our request schemes do not conform to a scenario that favors multicasting. This comes at the cost of restricting our coding schedule to be ordered over pairwise coded traffic. However, our coded cache scheme exploits the utility of both unicast and multicast transmissions, jointly optimizing caching, delivery rates, and cross-coding across contents. Hence, our design lies somewhere between the fully uncoded and fully coded frameworks.

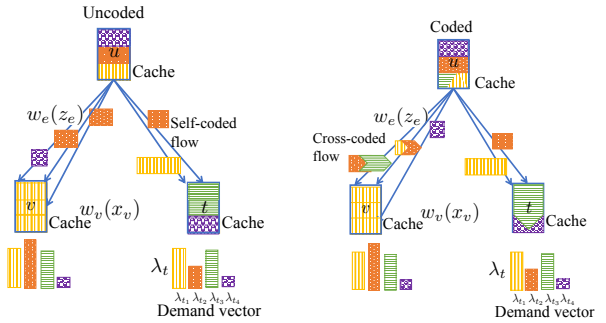
More specifically, we consider a flow-based coded caching network model that incorporates costs accrued by both delivery and caching. Both coded as well as cross-coded content can be stored at arbitrary nodes in the network, and subsequently transmitted and routed to meet demand. The demand distribution is asymmetric, and our model thus requires a mixture of unicast and multicast transmissions. In the phase of delivery, each flow can be either self-coded or cross-coded across different files using network coding techniques [14]. Our cost model is general: it encompasses different cost functions for both transmission and caching, enabling us to quantify the relative cost of caching versus delivery. Our coded caching scheme aims to serve demand by optimizing across the possible ways of generating self-coded and cross-coded transmission flows, leveraging the benefits of both uncoded and coded caching.

Our main contributions can be summarized as follows:

- We provide a flow-based coded caching framework over arbitrary network topologies by incorporating transmission as well as storage costs.
- In the proposed framework, traffic flows are self-coded or cross-coded in pairs, as shown in Fig. 1. Coding enables the end user to extract the required number of degrees of freedom with the help of caching at the local storage, which naturally eliminates the redundancy in models that rely on uncoded transmissions. Cross-coding increases the delivery capacity of the system, while introducing additional system optimization parameters (namely, the rates of the delivered and stored cross-coded traffic).
- By restricting the cardinality of cross-coded traffic, and imposing and exploiting an ordered decoded schedule, our system design that amounts to a tractable, convex optimization problem. The latter jointly optimizes caching, delivery, and cross-coding over the (arbitrary) network topology, under heterogeneous demand.
- We extensively evaluate our proposed framework over several synthetic and real-life topologies. Our framework significantly outperforms competitors, reducing transmission costs by several orders of magnitude.

This work was supported in part by the National Science Foundation under Grants CNS 2008639 and CNS 2107062.

Co-funded by the European Union (ERC, SENSIBILITÉ, 101077361). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.



(a) Self-coded setting with no cross-coded files.

(b) Coded setting with self-coded or cross-coded files.

Fig. 1: A general coded caching network with asymmetric demands. Caches are indicated as tall boxes where each distinct file has a different pattern and each cache can contain self-coded or cross-coded files, i.e., bars with different patterns. Demand vectors are shown as bar graphs. Flows are indicated by arrows. The parameter $m_e(z_e)$ denotes the delivery cost of flow rate z_e across edge e, and $w_v(x_v)$ represents the caching cost of storage load x_v at node v.

The rest of the paper is organized as follows. In Section II we provide a comprehensive review of the related literature. In Section III we detail our cache network model. We propose a coded caching scheme by jointly minimizing the costs of delivery and caching in Section IV. In Section V, we run extensive simulations and demonstrate the gains of the flow-based coded caching framework versus the competitors over various topologies.

II. Related Work

Erasure coding has been extensively studied in the context of storage systems. Examples include regenerating codes to reduce repair bandwidth [14], storage-bandwidth trade offs [11], erasure coded distributed system design and maximizing the service rate region [15], and erasure coded atomic (strongly consistent) distributed read and write storage service [16]. We depart by using random linear network codes, that lead to different feasibility regions than above works.

Network coding has been shown to achieve capacity in multicast transmissions with polynomial encoding/decoding complexity [17]. It is exploited for minimum cost multicasting [18], and to minimize total cost of edge using flow splitting [19]. Inter-session coding, i.e., coding of packets belonging to different sessions, has been proposed for enhancing distributed operation with simple scheduling and adaptability to unknown topologies [20]. The tradeoff between delay versus coding efficiency has been studied to minimize transmission cost and packet delays via a control policy that relies on queue lengths [21]. None of these works considers the impact of caching.

Physical layer caching with or without coding has been widely studied to optimize wireless networks. The gain offered by local caching and broadcasting is characterized in the landmark paper by Maddah-Ali and Niesen (MAN) [9], where a single multicast transmission suffices to meet the demand which is assumed to be symmetric, and the placement cost is not accounted for. Extensions of MAN include the analysis of the scaling of the per-user throughput and collaboration distance [22], the determination of the wireless caching capacity region [23], as well as single-hop and device-to-device delay [9], [22], [24]. Coded caching has been leveraged for wireless networks [25], and network coded storage has been devised for scheduling [26]. We depart by studying complex network topologies, in both the wired and wireless regimes, restricting however our coding scheme to pair-wise cross-coding under a pre-defined order.

Alternatively, there has been research focusing on jointly optimizing the caching gain and resource usage. These efforts include decentralized optimization via femtocaching to minimize the download delay [1], distributed caching for content distribution networks (CDNs) [27], and distributed optimization of caching gain for given routing [2]. Throughput and delay scaling in content-centric networking (CCN) has been analyzed [28], and the cost of storing and accessing objects has been minimized by building Steiner trees [29], or mixing Steiner trees [30], where the throughput benefits of network coding equal the integrality gap of the Steiner tree formulation. Complementing the host-centric paradigms, information-centric networking (ICN) architectures have been explored to optimize both bandwidth and storage for efficient content distribution [31], minimizing network latency [32]. Other works include jointly optimizing caching and routing for latency guarantees [33], caching for minimizing delay in the presence of congestion [34], and energy-efficient caching with multipath routing to balance the cost of transmissions and caching via the multicast and unicast delivery modes [13]. However, these works have not incorporated coding.

Several works extend the MAN single-file-retrieval problem in [9]. The MAN scheme under the constraint of uncoded cache placement to achieve the minimum worst-case load among all possible demands has been studied in [12]. An improved delivery scheme when files are demanded multiple times to minimize the worst case load under uncoded placement has been proposed in [35]. The memory and rate tradeoff for coded caching has been characterized within a factor of 2 in [10]. Other extensions of [9] include its application to D2D caching [36], private coded caching [37], coded distributed computing [38], requesting multiple files [38], meeting linear and polynomial queries [39], retrieving general functions and understanding how the optimal worst-case load increases [40]. However, to the best of our knowledge, none of these extensions considers jointly the effects of delivery, demand asymmetry, mixing of self and cross-coded files, along with exploiting the benefits of unicast and multicast transmissions.

III. Problem Formulation

We consider a network of interconnected nodes forming a caching network. Individual nodes retrieve files from a finite catalog. All nodes in the network are capable of (a) storing files, (b) receiving and forwarding encoded traffic, but also (c) decoding and encoding traffic prior to forwarding. In particular, with respect to (c), nodes are also able to cross-code traffic across pairs of files, as shown in Fig. 1. Subject to costs

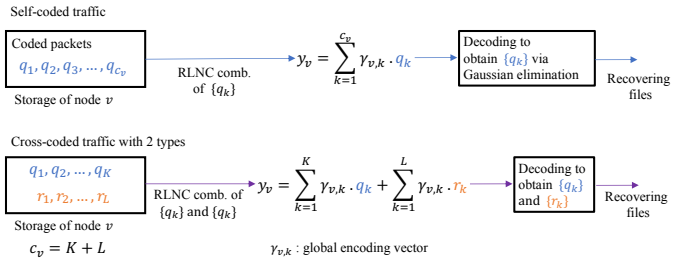


Fig. 2: RLNC for self-coded and cross-coded flow types.

associated with transmitting traffic, as well as storing files, the purpose of our system design is to jointly determine (a) how to cross-code traffic, (b) traffic flows, and, most crucially (c) where to store files, in order to meet demand. We elaborate on each of these issues in detail below.

A. Cache Network and Demand

More formally, we represent a cache network as a directed graph $G(V, E)$, with nodes V and edges $E \subseteq V \times V$. Nodes in the network wish to retrieve files from a catalog $\mathcal{C}$. Every node in the network has storage capabilities, and uses a random linear network code (RLNC) [41] to store (parts of) files in the catalog: that is, we assume that every file in $\mathcal{C}$ is partitioned sequentially into smaller *stages*; each stage is further partitioned into *(uncoded) packets* or *chunks*, and nodes can store random linear combinations of these uncoded packets. These random linear combinations, i.e., the *coded packets*, represent the DoF necessary to retrieve the stage. We assume that files and stages are large enough that a flow model, determined by mean transmission rates, is a good approximation of the system dynamics.[1] We illustrate the coding and decoding principles of RLNC via a simple example in Fig. 2, where coded packets in the storage of nodes are represented by variables q_k or r_k.

We define a set of *targets* $\mathcal{T} \subseteq V$ that act as content receivers (or traffic sinks). Every node $t \in \mathcal{T}$ is associated with a demand vector:

$$\lambda_t = [\lambda_{t,i}]_{i \in \mathcal{C}},$$

where $\lambda_{t,i} \in \mathbb{R}^+$ is the *request rate*, i.e., the intensity of the request process for coded packets of type $i \in \mathcal{C}$ at $t \in \mathcal{T}$. Intuitively, if $[0, T]$ is the duration of the entire operation period, then $\lambda_{t,i} \times T$ is the total number of coded packets/DoF that t needs to acquire during this period. Hence, if t is to acquire the entire file i, we can think of $\lambda_{t,i}$ as being proportional to the file size. Nevertheless, it is possible that the rates are higher for certain targets, to capture stringent delivery time requirements.

B. Caching Decisions and Storage Costs

As stated above, every node in the network can store and transmit coded packets of files in $\mathcal{C}$. Our use of an RLNC allows us to abstract caching decisions as follows: for every $i \in \mathcal{C}$ and $v \in V$, we denote by $x_{v,i} \in \mathbb{R}^+$ the *caching rate*, i.e., the rate with which node v can be used to produce coded packets of file i. The caching rates admit two possible physical interpretations:

1) Assuming that caches are *memory-bound*, meaning that storage is a limited resource, the caching rate $x_{v,i}$ can be thought of as being proportional to the number of coded packets (across stages) that node v stores. In other words, if $[0, T]$ is the total operation period, then $x_{v,i} \times T$ is the total storage consumed at v to be able to produce coded packets for file i at rate $x_{v,i}$.
2) If caches are *I/O-bound*, $x_{v,i}$ can be thought of as the rate with which coded packets can be *read* from the storage device. As memory becomes increasingly cheap, it may be possible to store *the entire file at* v (i.e., v stores equal to or even more DoF than the ones required to reconstruct every stage), and $x_{v,i}$ is simply the throughput of the I/O connection to storage.

We stress here that, in either interpretation $x_{v,i}$ are rates, and correspond to, e.g., coded packets per second.

Beyond coded packets for individual files, we also allow storing *cross-coded packets*: these are random linear combinations of chunks across a stage of file i and a corresponding stage of file j. The total storage load of node $v \in V$ is then:

$$x_v = \sum_{i \in \mathcal{C}} x_{v,i} + \sum_{i,j \neq i,(i,j) \in \mathcal{C}^2} x_{v,(i,j)}. \tag{1}$$

We assume that storing content incurs the aggregate cost:

$$\sum_{v \in V} w_v (x_v), \tag{2}$$

where $w_v : \mathbb{R}_+ \to \mathbb{R}_+$ is a convex non-decreasing function capturing storage costs.

Alternatively, we can introduce hard constraints. E.g., for memory-bound caches, we can have constraints of the form $x_v \leq c_v / T$, where $c_v \in \mathbb{R}_+$ is the storage capacity of $v \in V$. Similarly, in I/O bound caches, we can have constraints of the type $x_v \leq \mu_v$, where μ_v is the capacity of the I/O link. Both such hard constraints and soft-penalties of the form in Eq. (2) are permissible in our model.[2]

C. Traffic Flow and Bandwidth Costs

Traffic carrying flow for file i can be potentially generated at every node storing file i, traverse the network, and be consumed by target nodes requesting i. We consider two types of traffic. The first type corresponds to the coded traffic flow associated with file $i \in \mathcal{C}$. The second type of flow corresponds to cross-coded traffic flow associated with files $i, j \in \mathcal{C}$. In this case, cross-coded traffic flow corresponds to a (linear) combination of flows corresponding to files i and $j \neq i$. The amount of cross-coded traffic that can be generated at node v is determined by the quantities of incoming cross-coded traffic and coded traffic flows associated with single files.

We denote by $z_{e,i}$ the physical *traffic rate* on an edge $e = (u, v) \in E$ associated with (non-cross-coded) traffic for file

[1] As is typical [17], we assume that random weights used in coded packets are stored and transmitted along with coded packets, at a small (polylogarithmic) overhead compared to the payload. These weights can be used at a receiver to decode the packets.

[2] Though hard constraints may make meeting demand infeasible.

$i \in \mathcal{C}$. Similarly, we denote by $z_{e,(i,j)}$ the physical rate on an edge $e \in E$ associated with traffic resulting from cross-coding between i and j. The total physical flow over $e \in E$ is then

$$z_e = \sum_{i \in \mathcal{C}} z_{e,i} + \sum_{i,j \neq i,(i,j) \in \mathcal{C}^2} z_{e,(i,j)}. \tag{3}$$

Traffic flowing over an edge $e \in E$ incurs a cost. The total network cost due to delivery is given by: $\sum_{e \in E} m_e(z_e)$, where $m_e : \mathbb{R}_+ \to \mathbb{R}_+$ is a convex non-decreasing function that maps the load at edge $e \in E$ to the corresponding cost.

D. Problem Statement

Generally, we aim to solve the following problem:

$$\underset{z_e,\, x_v}{\text{Minimize:}} \quad \sum_{e \in E} m_e(z_e) + \sum_{v \in V} w_v(x_v) \tag{4a}$$

$$\text{subj. to:} \quad \text{flow and demand constraints.} \tag{4b}$$

That is, we wish to minimize costs due to caching and transmission across edges, while (a) respecting flow preservation constraints across nodes, and (b) meeting demand, i.e., ensuring that coded packets arrive at the target nodes at the desired rates. We will describe both types of constraints in great detail. In doing so, we determine both (i) where self-coded and cross-coded file contents are to be stored at/served from, via the respective caching rate variables, (ii) how self-coded and cross-coded traffic are to be routed along the network, and, concurrently, (iii) what is the relative balance in resource usage across self-coded and cross-coded traffic w.r.t. both resources (caching and transmission).

Formally stating the constituent constraints (a)-(b), and accomplishing (i)-(iii), poses several challenges. First, allowing for cross-coding across multiple files can potentially lead to a combinatorial explosion of variables. Second, cross-coded traffic can be utilized in a variety of ways: for example, cross-coded packets of type (i, j) can be used along with (decoded) self-coded packets of type i to produce decoded packets of type j, or combined together to produce decoded packets of both i and j. The RLNC solution, as shown in Fig. 2, captures the cross-coding mechanism, through encoding different traffic types and decoding these mixtures. However, packets/DoF that were used for the former type of decoding cannot be used for the latter type of decoding. This creates a complicated set of flow preservation constraints, and a need for careful management of flows at each node to determine the flow preservation rules as well as how demand can be met.

We address the above issues in the following ways. First, the combinatorial explosion of variables is addressed by restricting cross-coding to *pairs of files*, as introduced so far. Second, the management of cross-coded traffic is accomplished by imposing and exploiting an ordering on how traffic for different files is decoded. These assumptions yield a tractable formulation of the optimization in Eq. (4), which we describe next.

IV. Joint Caching, Cross-Coding, and Flow Optimization

A. Book-keeping and Meeting Demand

We first introduce additional variables to capture the amount of physical flow that can be used to satisfy demand at targets. We denote by $\rho^t_{e,i}$ and $\rho^t_{e,(i,j)}$ the portion of the self-coded and cross-coded traffic on an edge $e \in E$ that can be used to serve demand at target t, respectively. Note that physical traffic can be reused across targets. These variables should satisfy:

$$\rho^t_{e,i} \leq z_{e,i}, \quad \text{for all } e \in E, i \in \mathcal{C}, t \in \mathcal{T}, \tag{5}$$

$$\rho^t_{e,(i,j)} \leq z_{e,(i,j)}, \text{for all } e \in E, (i,j) \in \mathcal{C}^2, i \neq j, t \in \mathcal{T}. \tag{6}$$

Similarly, portions of the caching rates in nodes can be used to serve demands at targets. We indicate this via variables:

$$\xi^t_{v,i} \leq x_{v,i}, \quad \text{for all } e \in E, i \in \mathcal{C}, t \in \mathcal{T}, \tag{7}$$

$$\xi^t_{v,(i,j)} \leq x_{v,(i,j)}, \text{for all } e \in E, (i,j) \in \mathcal{C}^2, i \neq j, t \in \mathcal{T}. \tag{8}$$

These portions characterize the (potential) amount of caching rate traffic for file i, present at node v, which can be used in service of target t. Intuitively, these "book-keeping" variables will help ensure that the rates $z_{e,\cdot}$ and $x_{v,\cdot}$ are sufficient to meet demand in every target.

Correspondingly, demand at a target node is met by decoded traffic for requested files. We assume that every node, including non-targets, decodes incoming traffic (to serve demand in case of targets), but also to recode it and forward new random linear combinations towards other nodes. Both incoming traffic and stored content can be used to decode files at different nodes. In particular, let $\mu^t_{v,i}$ be the rate with which node v can decode content file i, that could subsequently be used to serve t. Then, decoded content can be used to generate outgoing traffic. In particular, outgoing traffic flow at every node should satisfy: for all $v \in V, i \in \mathcal{C}, t \in \mathcal{T}$.

$$\mu^t_{v,i} \geq \sum_{u:(v,u) \in E} \rho^t_{(v,u),i}. \tag{9}$$

Similarly, outgoing cross-coded traffic flow is governed by:

$$2 \min(\mu^t_{v,i}, \mu^t_{v,j}) \geq \sum_{u:(v,u) \in E} \rho^t_{(v,u),(i,j)}, \tag{10}$$

for all $v \in V, i, j \in \mathcal{C}, t \in \mathcal{T}$. This is due to the fact that any pair of decoded packets of i and j can be used to generate a pair of cross-coded packets of flow (i, j).

Finally, demand should be met; to that end:

$$\mu^t_{t,i} \geq \lambda_{t,i}, \quad \text{for all } t \in \mathcal{T}, \tag{11}$$

i.e., the decoding rate at each target should exceed the demand. We next turn our attention to how incoming traffic and stored content can be used to decode files.

B. Decoding Traffic

We assume that, the transmission of each stage is associated with a timeslot, where the schedule of when and where each packet is injected is given a priori [17]. Within each timeslot, every node follows an ordered decoding scheme: if $i < j$, then i is decoded before j. This leads to four different types of decoding to serve $t \in \mathcal{T}$ with respect to file $i \in \mathcal{C}$:

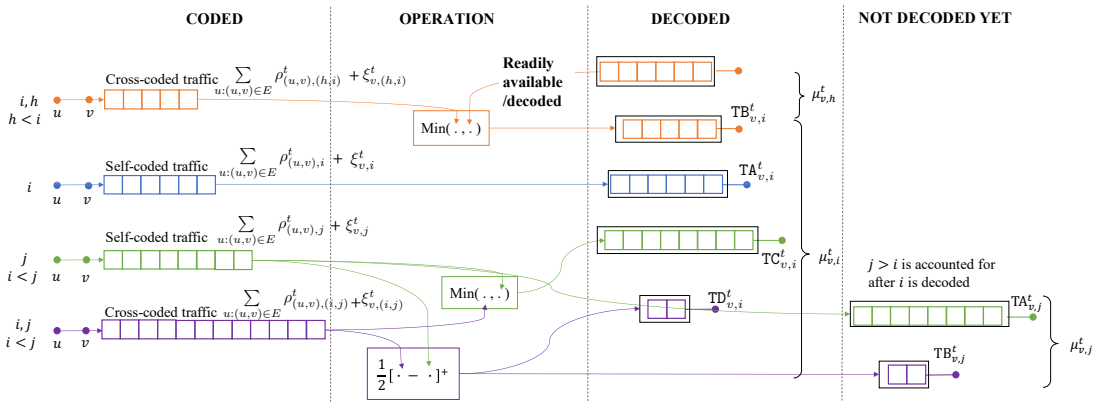


Fig. 3: Different types of decodings to serve t with respect to file i. Types $\mathtt{TA}^t_{v,i}$-$\mathtt{TD}^t_{v,i}$ are listed in the order of decoding.

(i) **Type-A (decoding pure i).** This flow type models the uncoded or self-coded available traffic for i at node v:

$$\mathtt{TA}^t_{v,i} = \textstyle\sum_{u:(u,v)\in E} \rho^t_{(u,v),i} + \xi^t_{v,i} \ .$$

This comprises incoming traffic w.r.t. file i as well as DoF stored at v.

(ii) **Type-B (decoding i with previously decoded traffic $h < i$).** This flow type models the cross-coded traffic flow (mixture) that exploits all previously decoded $h < i$, that are presently available at node v:

$$\mathtt{TB}^t_{v,i} = \textstyle\sum_{h<i} \min\left(\sum_{u:(u,v)\in E} \rho^t_{(u,v),(h,i)} + \xi^t_{v,(h,i)}, \mu^t_{v,h}\right) \ .$$

In this case, i is decoded using a mixture of i and h, where $h < i$, and pure (i.e., already decoded) h which is available at a rate $\mu^t_{v,h}$. Hence, $\mathtt{TB}^t_{v,i}$ is the effective amount of traffic for i extracted from the aggregate available mixture of all $h < i$.

(iii) **Type-C (recoding to extract i without decoding pure traffic $j > i$).** This flow type models the traffic for file i to be decoded by cross-coded traffic with files $j > i$, which has arrived at node v but has not yet been decoded:

$$\mathtt{TC}^t_{v,i} = \textstyle\sum_{j>i} \min\Big(\sum_{u:(u,v)\in E} \rho^t_{(u,v),(i,j)} + \xi^t_{v,(i,j)}, \\ \sum_{u:(u,v)\in E} \rho^t_{(u,v),j} + \xi^t_{v,j}\Big) \ .$$

In addition to pure j, node v has a mixture of files i and any j. This, combined with the flexibility of recoding with no need for intermediate decoding, ensures decoding of i. As in Type-B, the effective amount of traffic node v can extract for file i is the sum over all $j > i$ of the minimum of the available mixture of j with i and the incoming flow (as opposed to existing decoded traffic). Meanwhile, pure j is available but not decoded, which contributes to flow $\mathtt{TA}^t_{v,j}$ after i is decoded.

(iv) **Type-D (recoding to extract i from the leftover cross-coded traffic (i,j)).** This type models the traffic for i to be decoded using the mixture with j such that $j > i$. After decoding Types B and C, the remaining cross-coded traffic is used at node v to produce uncoded traffic for both i and j.

$$\mathtt{TD}^t_{v,i} = \textstyle\sum_{j>i} \frac{1}{2}\big[\sum_{u:(u,v)\in E}(\rho^t_{(u,v),(i,j)} + \xi^t_{v,(i,j)}) \\ -(\sum_{u:(u,v)\in E} \rho^t_{(u,v),j} + \xi^t_{v,j})\big]^+,$$

where $[x]^+$ is x if $x > 0$ and zero otherwise. The effective DoF of this residual mixture to the traffic for i is half of the original DoF since two DoFs in the mixture are required for one DoF of i (while also giving one DoF for j). Note that decoding of Type D also produces pure traffic of type j, i.e., $\mathtt{TB}^t_{v,j}$. This is accounted for in $\mu^t_{v,j}$ as Type B traffic.

As a result of combining these processes (Types A-D), we have that for all $v \in V$, $t \in \mathcal{T}$, $i \in \mathcal{C}$:

$$\mathtt{TA}^t_{v,i} + \mathtt{TB}^t_{v,i} + \mathtt{TC}^t_{v,i} + \mathtt{TD}^t_{v,i} \geq \mu^t_{v,i} \ . \tag{12}$$

We illustrate the different types of decodings (Types A-D listed in the order of decoding) in Fig. 3. We note that the asymmetry between Types B and C, D implies that constraints depend on the order of the files, as imposed by their indices $i, j, \ldots \in \mathcal{C}$. This is a design choice: the system is parameterized by which files it decodes first. Moreover, this ordering need not be global: every node could potentially have its own ordering, leading to a different formulation of constraints (12). Finally, the proposed scheme need not be constrained cross coding of pairs; combinations could be extended to triplets, quadruplets, etc., increasing the number of constraints from quadratic (in the number of files) to cubic etc. We nevertheless restrict to pairs for simplicity and for reducing problem complexity.

C. Optimization

Before we formally state the form optimization problem (4) takes, we revisit in detail the decision parameters we have so far. The unknowns are caching $x_{v,i}$, $x_{v,(i,j)}$, transmission loads $z_{e,i}$, $z_{e,(i,j)}$, portions of caching for demands $\xi^t_{v,i}$, $\xi^t_{v,(i,j)}$, portions of transmission loads for demands $\rho^t_{e,i}$, $\rho^t_{e,(i,j)}$, which are self-coded and cross-coded, respectively, and decoding rate $\mu^t_{v,i}$. From a flow-conservation perspective, we obtain the following optimization problem that minimizes the aggregate cost of delivery and caching:

$$\underset{z_e,\, x_v}{\text{Minimize:}} \quad \textstyle\sum_{e\in E} m_e(z_e) + \sum_{v\in V} w_v(x_v) \tag{13a}$$

$$\text{subj. to:} \quad \text{constraints (1), (3), and (5)–(12).} \tag{13b}$$

We note that, if weight functions are convex, the optimization Prob. (13) is convex; the convexity of all constraints is

Graph	$\|V\|$	$\|E\|$	$\|\mathcal{T}\|$	$\|\mathcal{C}\|$	Cost
synthetic topology					
Erdős-Rényi (ER)	50	256	5	25	23
grid	64	224	5	25	34206
hypercube (HC)	64	384	5	25	40
expander	64	444	5	25	37
small-world (SW) [43]	64	306	5	25	52
Barabási and Albert (BA) [44]	50	282	5	25	18
Watts-Strogatz (WS) [45]	50	100	5	25	2826
backbone network [46]					
GEANT	22	66	5	20	17
Abilene	11	28	5	50	0.15
Deutsche Telekom (Dtelekom)	68	546	5	20	4186
hierarchy topology					
Maddah-Ali and Niesen (MAN)	3	2	2	4	-
Tree	14	13	9	20	-

TABLE I: Graph Topologies and Experiment Parameters

easy to verify for all cases except (12). We next show that (12) is also convex. We thus could solve the convex program (13) through classic solvers [42].

Lemma 1. *The set of constraints* (12) *is convex.*

Proof. Type A decoding flow is an affine function. The minimum of affine functions is concave, hence Type B decoding flow is also a concave function. Finally, the sum of Type C and Type D is also a concave function due to:

$$\min(a,b) + \tfrac{1}{2}[a-b]^+ = \tfrac{1}{2}[a+\min(a,b)],$$

which is concave since the min operator is concave. □

V. Performance Evaluation

In this section, we provide a comprehensive evaluation of the flow-based coded caching scheme to understand the utility, capacity, and bandwidth tradeoffs for different topologies.

A. Experiment Setup

To evaluate our scheme, we perform simulations over general topologies (synthetic topologies, and backbone network topologies), and two hierarchy topologies. These topologies and their parameters are summarized in Table I.

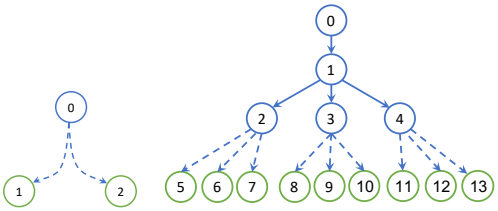
Settings for General Topologies. To simulate a distribution network, we "embed" a backbone/CDN-like topology in arbitrary topologies as follows. First, for every node $v \in V$ we compute the average distance to other nodes $d(v) = \sum_u \frac{d(v,u)}{|V|-1}$ and the *node centrality* $c(v) = 1/d(v)$. Then, we assign the transmission cost function of edge $(u,v) \in E$ as

$$m_e(z_{(u,v)}) = z_{(u,v)}^{\alpha(d(u)+d(v))+\beta},$$

where $\alpha = 10$, and $\beta = -20 \min d(v) + 1$ are selected so as to get a range of exponents larger than 1 (so that the function is convex), and the cache capacity of node $v \in V$ is given as

$$c_v = \alpha' c(v) + \beta',$$

where $\alpha' = 3$ and $\beta' = -3 \min c(v) + 1$ are selected again to get a wide range of cache values larger than 1. Intuitively, the more central a node is, the higher its storage capacity, and the higher the throughput of closeby edges (as captured by more lenient penalties). We uniformly at random (u.a.r.) select target set $\mathcal{T}$. For each target $t \in \mathcal{T}$, its demand for item $i \in \mathcal{C}$ follows a Zipf distribution with parameter 1.2.



(a) MAN (b) Tree

Fig. 4: Topologies for two toy examples, where each solid line represents a unicast transmission, the dotted lines denote hyperedges, and green nodes/leaves denote targets.

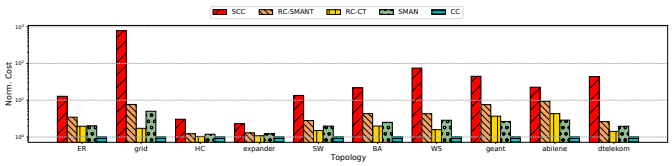


Fig. 5: Transmission costs of various caching algorithms (normalized by CC costs, shown in column "Cost" of Table I) over different topologies. Our proposed CC consistently outperforms competitors by several orders of magnitude.

Settings for Hierarchy Topologies. In order to provide some visualization intuitions and compare to the most related work from Maddah-Ali and Niesen [9], we show how schemes work over these two simpler hierarchy topologies. As shown in Fig. 4, green leaves represent targets $\mathcal{T} \subseteq V$. Furthermore, we capture the broadcasting nature in [9] via dotted hyperedges. For the MAN, we refer the interested readers to [47]. For the Tree topology, we set the cache size for each layer from the server to the leaf to be $\mathbf{c_v} = (20, 8, 4, 2)$. The edge penalty vector between these ordered layers from the server to the leaf nodes is $\mathbf{m_e}(z_e) = (z_e, z_e^2, z_e^4)$, respectively; note that the cost functions of transmission from the same layer is fixed and identical. This is natural: in wireless systems, the server typically has a higher bandwidth or higher transmission power than the leaves, thus penalty of transmission with higher bandwidth/power is lower.

B. Joint Caching, Delivery, and Cross-coding Algorithms

We implement our framework and several competitors:

- Coded Caching (CC) is our proposed scheme, consisting of both self-coded and cross-coded traffic management.
- Self-Coded Caching (SCC) consists of only self-coded traffic. SCC optimizes Problem (13) by setting all cross coded variables to 0.
- 'Simulated' Maddah-Ali and Niesen (SMAN) is a simulated version of the scheme introduced by Maddah-Ali and Niesen [9]. We first take the maximum rate of each file across all targets as the demand rate, so that the symmetry is guaranteed, as in [9]. Then, we optimize both cache and transmission by solving a "uniform demand" version of Prob. (13). Note that the original scheme from [9] is symmetrically designed so as to maximize

multicasting opportunities and does not optimize caching. In contrast, we extend their scheme using an asymmetric demand model under a flow-based coded caching scheme, and optimize caches accordingly.

- Random Caching and Coded Transmission (RC−CT) consists of two steps. First, we set cache rates proportional to file populations. Second, given fixed cache rates, we optimize Problem (13) only w.r.t. z_e.
- Random Caching and SMAN Transmission (RC−SMANT) also consists of two steps similar to RC−CT. The only difference is: before the second step, RC−SMANT takes the maximum rate of each file across all targets as the demand rate, like SMAN.

We implement all these schemes using CVXPY. Our code is publicly available.[3]

C. Results for General Topologies

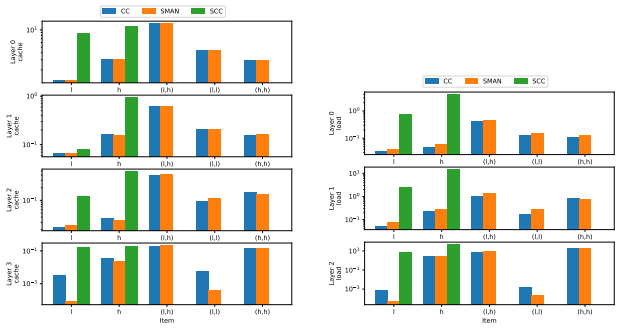
We present normalized cost results in Fig. 5. The normalized cost is the transmission cost normalized by the one yielded by CC, shown in the last column of Table I. In all topologies, our CC *consistently outperforms all competitors by several orders of magnitude*. SCC always performs the worst, which verifies the importance of cross coding. Also, schemes considering asymmetric demands (CC and RC−CT) always have better performance than ones only considering symmetric demands (SMAN and RC−SMANT), correspondingly. This is expected, as symmetric demands pessimistically assume they need to uniformly serve the maximum of asymmetric demands.

D. Tree Topology

We next consider the Tree network topology with four multi-hop layers, as shown in Fig. 4-(b). In the Tree topology, solid lines between nodes represent unicast transmissions (e.g., from the server to the radio area network), and the dotted lines represent the hyperedges from the radio area network to the leaves (e.g., wireless on tablet, phone). The ordering of layers is from root to leaf. The demand for the Tree network is modeled by a Zipf distribution, where the catalog size is $|\mathcal{C}| = 20$ and the Zipf exponent is 1.2. Although the demands from different targets all follow the same distribution, they might take different values. We group files for easier visualization and denote the collection of the 10 most popular files by h (stands for High popular), and the least popular 10 files by l (stands for Low popular). There are 5 possible coded combinations denoted by the set $\{l, h, (l,h), (l,l), (h,h)\}$, where l and h denote the self-coded files, (l,h) represents the cross-coding across l and h files, and so forth.

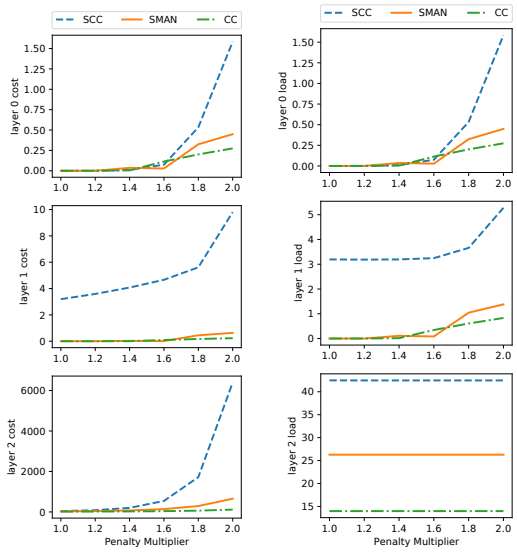
Transmission and Cache Placement. In Fig. 6, we illustrate the performance across layers. In this setting, we observe that the combination of the edge caches and edge transmissions is not sufficient to meet the demands, i.e., all intermediate caches and transmissions help satisfy the demands. Our CC has the least transmission costs. We list the total transmission costs at the captions of the Figs. 6. Our CC scheme achieves more than $3\times$ gains compared to SMAN, while achieving more than $250\times$ gains compared to SCC.

[3]https://github.com/neu-spiral/CodedCaching



(a) Cache (b) Transmission

Fig. 6: (a) Cache storage and (b) transmission load for different algorithms with symmetric demands. The indices l and h represent low and high popularity files. The total transmission cost of three algorithms is 3.2×10^4, 1.1×10^5 and 8.7×10^6.



(a) Cost (b) Transmission Load

Fig. 7: Cost and transmission load with different cost functions, where larger penalty multiplier indicates higher penalization for transmission.

Effect of Edge Load Penalties. In Fig. 7 we consider the cost across each layer with varying edge penalty. This helps illustrate how the relative cost of delivery versus caching affects the behavior of the aggregate cost of the scheme. The label penalty multiplier PenaltyMultiplier on the x-axis is a coefficient in transmission cost function $m_e(z_e)$, where edges in layer i have $m_e(z_e) = z_e^{\text{PenaltyMultiplier}^i}$. With the increment of penalty multiplier, the load of edge on layer 2 stays almost the same, while the load on the lower layer edges increases. The increment rate of SCC approach grows the fastest, then SMAN, and our CC scheme grows the slowest.

VI. Conclusions

We proposed a flow-based coded caching framework. To the best of our knowledge, this is the first comprehensive

work that considers asymmetric demand models, a mixture of unicast and multicast transmissions, self-coded and cross-coded delivery, via minimizing the joint cost of delivery and placement in general wireless network topologies. Our numerical experiments show that, for various arbitrary topologies, our coding scheme outperforms the widely accepted algorithms by several orders of magnitude. Moreover, under cross-coding, we observe a $2\times$ reduction in transmission costs versus the self-coded model in Maddah-Ali and Niesen topology. Furthermore, the reduction of delivery cost is even more dramatic (up to more than $3\times$ versus the Maddah-Ali and Niesen scheme and up to more than $250\times$ versus self-coded) in hierarchical (tree) transmission models. Extensions of this work include devising a general delivery scheme, where the link costs are coupled, accounting for, e.g., interference, and providing a constant factor approximation of the optimal solution of the joint placement and delivery problem without imposing a decoding schedule. Designing adaptive and distributed techniques is critical, especially when the demand is dynamic and unknown. Other extensions include designing joint caching and routing schemes and scheduling transmissions via incorporating congestion.

REFERENCES

[1] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, Sep. 2013.

[2] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 737–750, Feb. 2018.

[3] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc., IEEE Infocom*, Hong Kong, Apr. 2015, pp. 936–944.

[4] S. Ioannidis and E. M. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," in *Proc., ACM Conf. Inf.-Centric Netw.*, Berlin, Germany, Sep. 2017.

[5] A. Khalesi and P. Elia, "Multi-user linearly-separable distributed computing," *arXiv preprint arXiv:2206.11119*, Jun. 2022.

[6] M. Mahdian, A. Moharrer, S. Ioannidis, and E. Yeh, "Kelly cache networks," in *Proc., IEEE Infocom*, Honolulu, HI, Apr. 2019.

[7] D. Malak, F. V. Mutlu, J. Zhang, and E. M. Yeh, "Transmission delay minimization via joint power control and caching in wireless HetNets," *arXiv preprint arXiv:2105.14380*, May 2021.

[8] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *J. Comb. Optim.*, vol. 8, no. 3, pp. 307–328, Sep. 2004.

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

[10] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 647–663, Sep. 2018.

[11] N. Prakash, V. Abdrashitov, and M. Médard, "The storage versus repair-bandwidth trade-off for clustered storage systems," *IEEE Trans. Inf. Theory*, vol. 64, no. 8, pp. 5783–5805, Feb. 2018.

[12] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *Proc., IEEE Inf. Theory Wksh.*, Cambridge, UK, Sep. 2016, pp. 161–165.

[13] Y. Wang and V. Friderikos, "Energy-efficient proactive caching with multipath routing," *Computer Networks*, vol. 216, p. 109272, 2022.

[14] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Aug. 2010.

[15] M. Aktaş, G. Joshi, S. Kadhe, F. Kazemi, and E. Soljanin, "Service rate region: A new aspect of coded distributed system design," *IEEE Trans. Inf. Theory*, vol. 67, no. 12, pp. 7940–7963, Oct. 2021.

[16] N. Nicolaou, V. Cadambe, N. Prakash, K. Konwar, M. Medard, and N. Lynch, "ARES: adaptive, reconfigurable, erasure coded, atomic storage," pp. 2195–2205, Jul. 2019.

[17] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Physical Commun.*, vol. 1, no. 1, pp. 3–20, Mar. 2008.

[18] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. IEEE Joint Conf. Computer and Commun. Socs.*, vol. 3, Miami, FL, Mar. 2005, pp. 1607–1617.

[19] Y. Cui, M. Médard, E. Yeh, D. Leith, and K. R. Duffy, "Optimization-based linear network coding for general connections of continuous flows," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2033–47, Sep. 2018.

[20] A. Eryilmaz, D. S. Lun, and B. Swapna, "Control of multi-hop communication networks for inter-session network coding," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 1092–1110, Jan. 2011.

[21] Y.-P. Hsu, N. Abedini, N. Gautam, A. Sprintson, and S. Shakkottai, "Opportunities for network coding: To wait or not to wait," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1876–1889, Sep. 2014.

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

[23] U. Niesen, D. Shah, and G. W. Wornell, "Caching in wireless networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6524–6540, Jun. 2012.

[24] D. Malak, M. Al-Shalash, and J. G. Andrews, "Optimizing content caching to maximize the density of successful receptions in device-to-device networking," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4365–4380, Oct. 2016.

[25] J. Hachem, N. Karamchandani, S. Diggavi, and S. Moharir, "Coded caching for heterogeneous wireless networks," *arXiv preprint arXiv:2006.01025*, Jun. 2020.

[26] U. J. Ferner, P. Sadeghi, N. Aboutorab, and M. Médard, "Scheduling advantages of network coded storage in point-to-multipoint networks," in *Proc., IEEE Int. Symp. Netw. Coding*, Aalborg Oest, Denmark, Jun. 2014, pp. 1–6.

[27] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc., IEEE Infocom*, San Diego, CA, Mar. 2010.

[28] M. Mahdian and E. M. Yeh, "Throughput and delay scaling of content-centric ad hoc and heterogeneous wireless networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3030–3043, 2017.

[29] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–29, Aug. 2008.

[30] C. Fragouli and E. Soljanin, *Network coding fundamentals*. Now Publishers Inc, 2007.

[31] E. M. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "Vip: A framework for joint dynamic forwarding and caching in named data networks," in *Proc., ACM Conf. Inf.-Centric Netw.*, Paris, France, Sep. 2014.

[32] M. Mahdian and E. Yeh, "Mindelay: Low-latency forwarding and caching algorithms for information-centric networks," *arXiv preprint arXiv:1710.05130*, Oct. 2017.

[33] J. Li, T. K. Phan, W. K. Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "Dr-cache: Distributed resilient caching with latency guarantees," in *Proc., IEEE Infocom*, Honolulu, HI, Apr. 2018, pp. 441–449.

[34] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 864–874, May 2013.

[35] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 1281–1296, Dec. 2017.

[36] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Dec. 2015.

[37] K. Wan and G. Caire, "On coded caching with private demands," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 358–372, Nov. 2020.

[38] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–28, Sep. 2017.

[39] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2722–2734, Feb. 2020.

[40] K. Wan, H. Sun, M. Ji, D. Tuninetti, and G. Caire, "On optimal load-memory tradeoff of cache-aided scalar linear function retrieval," *IEEE Trans. Inf. Theory*, vol. 67, no. 6, pp. 4001–4018, Mar. 2021.

[41] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[42] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[43] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc., Annu. ACM Symp. Theory Comp.*, 2000, pp. 163–170.

[44] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[45] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[46] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, pp. 1–6, 2011.

[47] D. Malak, Y. Li, S. Ioannidis, E. Yeh, and M. Medard, "Extended version of coded caching networks," 2023. [Online]. Available: https://Sherry6352.github.io/files/Coded_Caching.pdf