Congestion-aware Routing and Content Placement in Elastic Cache Networks

Jinkun Zhang and Edmund Yeh
Department of Electrical and Computer Engineering, Northeastern University, US
zhang.jinku@northeastern.edu eyeh@ece.neu.edu

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621304

***Abstract*—Caching can be leveraged to significantly improve network performance and mitigate congestion. However, characterizing the optimal tradeoff between routing cost and cache deployment cost remains an open problem. In this paper, for a network with arbitrary topology and congestion-dependent nonlinear cost functions, we aim to jointly determine the cache deployment, content placement, and hop-by-hop routing strategies, so that the sum of routing cost and cache deployment cost is minimized. We tackle this mixed-integer nonlinear problem starting with a fixed-routing setting, and then generalize to a dynamic-routing setting. For the fixed-routing setting, a Gradient-combining Frank-Wolfe algorithm with $(\frac{1}{2}, 1)$-approximation is presented. For the general dynamic-routing setting, we obtain a set of KKT conditions, and devise a distributed and adaptive online algorithm based on these conditions. We demonstrate via extensive simulation that our algorithms significantly outperform a number of baselines.**

***Index Terms*—Caching, Routing, Information-centric network**

I. Introduction

Caching, by bringing popular data objects closer to consumers, is recognized as one of the most efficient ways to mitigate bandwidth bottlenecks and reduce delay in modern content delivery networks. However, as a resource, network caches typically neither have prescribed sizes, nor are provided for free. The network operator may pay a cost to deploy caches of elastic sizes across the network (e.g. rent from service providers or purchase and install manually), if doing so yields satisfactory improvement in network performance. Therefore, a rational network operator may seek to quantify the tradeoff between cache deployment costs and network performance metrics. In this paper, by formulating and solving the problem of joint routing and caching with elastic cache sizes, we explicitly investigate this important tradeoff.

The multi-commodity routing problem with arbitrary network topology and nonlinear costs is optimally solved [1], and the caching problem with fixed path and cache capacities is solved with $1 - \frac{1}{e}$ approximation [2]–[5]. Recently, routing and caching have been considered jointly as they are highly coupled. i.e., we wish to cache on the routing path, and to route to a cache site. Jointly-designed routing and caching strategies can reduce routing costs significantly compared to those designed separately [6]. Joint optimization of routing and caching is studied in various networking contexts, such as content delivery networks (CDNs) [7]–[9] and information-centric networks (ICNs) [10], [11]. Using the idea of back-pressure [12], a throughput-optimal dynamic forwarding and caching algorithm for ICN is proposed by [13]. Approximation solutions are proposed for joint routing and caching with linear costs in bipartite graphs [7], or with arbitrary network topology [6]. Nevertheless, the optimal routing and caching with arbitrary topology and nonlinear costs remains an open problem. A heuristic algorithm was proposed for this problem [14], however, without an analytical guarantee.

On the other hand, optimization over elastic cache sizes has drawn significant attention in recent years. This corresponds to the demand of rapidly growing small content providers who tend to lease storage from elastic CDNs, e.g., Akamai Aura, instead of purchasing and maintaining storage by themselves. One line of work focused on jointly optimizing cache deployment and content placement subject to a cache budget constraint [15]–[22]. Another line considered the tradeoff between cache deployment costs and cache utilities, so that the budget itself is optimized to pursue a maximum gain [23]–[28]. In the latter line, almost all cache utilities are defined as functions of cache hit count or ratio. Whereas, in a network of arbitrary topology and congestion-dependent nonlinear costs, a higher cache hit count or ratio does not necessarily yield a lower routing cost. It is more directly in the network operator's interest to achieve lower routing costs, e.g., lower user latency or link usage fee. For example, requests served with a hit ratio 1 at distant servers could incur severer network congestion and thus a higher average delay, compared to requests served locally with a lower hit ratio [29].

To our knowledge, the tradeoff between routing cost and cache deployment cost remains an open problem. In this paper, we fill this gap by minimizing a total cost – the sum of routing cost and cache deployment cost. This paper differs from previous works as we simultaneously (i) assume an arbitrary multi-hop network topology, (ii) adopt a hop-by-hop routing scheme with congestion-dependent nonlinear costs, (iii) consider elastic cache sizes with nonlinear deployment costs, (iv) achieve tradeoff between network performance and cache deployment costs instead of operating with a fixed budget, and (v) directly incorporate routing costs as the performance metric instead of cache utilities.

We consider a cache-enabled content delivery network with stochastic, stationary request arrivals. Requests are routed hop-by-hop until they reach a cache hit. Content items are then sent back to requesters along the reverse path. Costs are incurred on the links due to transmission, and at the nodes due to cache deployment. We aim to minimize the total cost by devising a

distributed and adaptive online algorithm that determines the routing and caching strategies simultaneously.

We study the proposed problem first in a fixed-routing special case, and then in the general dynamic-routing setting. For the fixed-routing case, we achieve a $(\frac{1}{2}, 1)$ approximation by the Gradient-combining Frank-Wolfe algorithm [30]. The general case, to our knowledge, has no known constant factor approximation due to the loss of submodularity. Nevertheless, inspired by [1], we propose a method with strong theoretical insight by presenting and modifying the KKT conditions. It suggests each node handles arrival requests in a way that achieves minimum marginal cost – either by forwarding to a nearby node or by expanding the local cache.

The main contributions of this paper are as follows:

- We propose a novel mathematical framework unifying cache deployment, content placement and routing for arbitrary network topology and nonlinear costs. We formulate a mixed-integer cost minimization problem.
- With fixed-routing, we recast the problem as *submodular + concave* maximization, where a Gradient-Combining Frank-Wolfe algorithm achieves an $1 - \frac{1}{e}$ approximation.
- In the general case, we obtain the KKT necessary condition, and develop a modification to the KKT condition.
- We propose a distributed adaptive gradient projection algorithm that converges to the modified condition, and compare proposed algorithms against baselines in multiple scenarios through extensive simulation.

II. Model and Problem Formulation

Cache-enabled network. We model a cache-enabled network by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of directed links. We assume for any $(i,j) \in \mathcal{E}$, $(j,i) \in \mathcal{E}$ also. For node $i \in \mathcal{V}$, let $\mathcal{N}(i) = \{j \in \mathcal{V} | (i,j) \in \mathcal{E}\} = \{j \in \mathcal{V} | (j,i) \in \mathcal{E}\}$ denote the neighbors of i.

Let $\mathcal{C}$ denote the set of content items, i.e., the *catalog*. We assume all items are of equal size $L_{\text{item}} = 1$.[1] Items are permanently kept at their *designated servers*, without consuming the servers' cache space. Let set $\mathcal{S}_k \subseteq \mathcal{V}$ be the designated server(s) for item $k \in \mathcal{C}$. Nodes have access to caches of elastic size, and can optionally store content items by consuming corresponding cache space. We denote node i's *cache decisions* by $\boldsymbol{x}_i = [x_i(k)]_{k \in \mathcal{C}}$, where the binary decision $x_i(k) \in \{0,1\}$ indicates whether node i chooses to cache item k (i.e., $x_i(k) = 1$ if node i caches item k, and 0 if not). We denote the *global caching decision* by $\boldsymbol{x} = [x_i(k)]_{i \in \mathcal{V}, k \in \mathcal{C}}$.

Request and response routing. Packet transmission in $\mathcal{G}$ is request driven. We use (i,k) to denote the request made by node i for item k, and assume that request packets of (i,k) are generated by i at a (quasi-stationary) rate $r_i(k)$ (request packet/sec) called the *exogenous request input rate*. Let $\boldsymbol{r} = [r_i(k)]_{i \in \mathcal{V}, k \in \mathcal{C}}$. Request packets are routed in a hop-by-hop manner using only local information Let $t_i(k)$ be the total request arrival rate for item k at node i. That is, $t_i(k)$ includes node i's exogenous request input rate $r_i(k)$, and the rate of endogenously arrival requests forwarded from other nodes to node i. Of the request packets for item k that arrive at node i, a fraction of $\phi_{ij}(k) \in [0,1]$ is forwarded to neighbor $j \in \mathcal{N}(i)$. Thus for any $i \in \mathcal{V}$ and $k \in \mathcal{C}$[2],

$$t_i(k) = r_i(k) + \sum\nolimits_{j \in \mathcal{V}} t_j(k) \phi_{ji}(k), \tag{1}$$

where $\phi_{ij}(k) \equiv 0$ if $(i,j) \notin \mathcal{E}$. We denote i's *routing strategy* by $\boldsymbol{\phi}_i = [\phi_{ij}(k)]_{j \in \mathcal{V}, k \in \mathcal{C}}$, and let $\boldsymbol{\phi} = [\phi_{ij}(k)]_{i,j \in \mathcal{V}, k \in \mathcal{C}}$.

Request packets for k terminate at i if $i \in \mathcal{S}_k$ or $x_i(k) = 1$. When a request packet terminates, a response packet is generated. The response packet delivers the requested item back to requester along the same path taken by the request packet, in the reverse direction.[3] An example network is shown in Fig. 1. The following holds for all $i \in \mathcal{V}$ and $k \in \mathcal{C}$,

$$x_i(k) + \sum\nolimits_{j \in \mathcal{V}} \phi_{ij}(k) = \begin{cases} 1, & \text{if } i \notin \mathcal{S}_k, \\ 0, & \text{if } i \in \mathcal{S}_k. \end{cases} \tag{2}$$

Routing and cache costs. Costs are incurred on the links due to packet transmission or packet queueing, and at the nodes

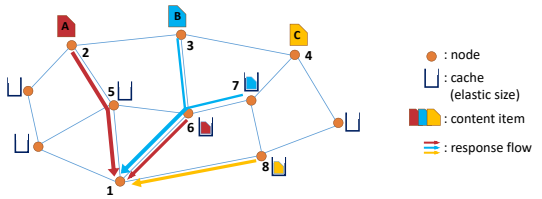

Fig. 1: An example network. Node 1 makes requests for item A, B and C, designated servers are at node 2, 3 and 4, respectively. Node 6, 7, 8 cache the items. Requests are forwarded hop-by-hop. Responses shown by colored arrows fetch the items back to 1 along the reverse path of request.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Network graph $\mathcal{G}$, set of nodes $\mathcal{V}$ and links $\mathcal{E}$
$\mathcal{N}(i)$	Set of neighbors of node i
$\mathcal{C}$,$\mathcal{S}_k$	Set of content items; Designated servers of item k
$\phi_{ij}(k)$	Routing variable from node i to j for item k
$r_i(k)$	Exogenous input request rate for item k at node i
$t_i(k)$	Total request arrival rate of item k at node i
$fij(k)$,F_{ij}	Flow rate of item k and total flow rate on link (i,j)
$D_{ij}(F_{ij})$	Communication cost (e.g. queueing delay) on (i,j)
$x_i(k)$	Cache decision of node i for item k
X_i, $B_i(X_i)$	Cache occupancy and deployment cost at node i
$y_i(k)$, Y_i	Continuous cache variables and occupancy
$T(\boldsymbol{\phi}, \boldsymbol{y})$	Network aggregated cost
$G(\boldsymbol{y})$	Caching gain for fixed-routing case
$\delta_{ij}(k)$	Marginal cost due to increase of $f_{ji}(k)$
$\delta_i(k)$	Minimum marginal cost for i to handle requests of k.

TABLE I: Major notations

[1] Contents files of non-equal sizes can be partitioned into file chunks of equal size, and each file chunk is treated as a different item in $\mathcal{C}$.

[2] We remark that the decision variables are only $\boldsymbol{x}$ and $\boldsymbol{\phi}$, whereas $t_i(k)$ is not decision variable. In the absence of caching and for a given $\boldsymbol{\phi}$, the uniqueness of $t_i(k)$, i.e., the existence of a unique solution to equation (1), is shown by Theorem 1 [1] with mild assumptions.

[3] Such mechanism is implemented in ICN with the Forwarding Interest Base (FIB) and Pending Interest Table (PIT) [13]. We assume different request packets for the same item are recorded and routed separately.

due to cache deployment. Since the size of request packets is typically negligible compared to the size of responses carrying content items, we consider only the link cost caused by responses. Let $f_{ij}(k)$ be the rate of responses (response packet/sec) traveling through link (i,j) carrying item k. Recall that $L_{\text{item}} = 1$, then $f_{ij}(k)$ equals to the flow rate (bit/sec) on link (i,j) due to item k. Let F_{ij} be the total flow rate on (i,j). Since each request packet forwarded from i to j must fetch a response packet travelling through (j,i), we have $f_{ji}(k) = t_i(k)\phi_{ij}(k)$, and $F_{ij} = \sum_{k\in\mathcal{C}} f_{ij}(k)$. We denote by $D_{ij}(F_{ij})$ the flow-dependent routing cost on link (i,j), and assume the cost function $D_{ij}(\cdot)$ is continuously differentiable, monotonically increasing and convex, with $D_{ij}(0) = 0$. Such $D_{ij}(\cdot)$ subsumes a variety of existing cost functions, including linear cost (transmission delay), link capacity constraints. It also incorporates congestion-dependent performance metrics. For example, let μ_{ij} be the service rate of an M/M/1 queue, then $D_{ij}(F_{ij}) = F_{ij}/\left(\mu_{ij} - F_{ij}\right)$ gives the average number of packets waiting in the queue or being served [31]. By Little's Law, the aggregated cost is proportional to the expected system delay of packets. The cache occupancy at node i is given by $X_i = \sum_{k\in\mathcal{C}} x_i(k)$. We denote by $B_i(X_i)$ the cache deployment cost at node i, and assume $B_i(\cdot)$ to be continuously differentiable, monotonically increasing and convex, with $B_i(0) = 0$. The cache deployment cost can represent the expense to buy/rent storage (e.g., [23], [24], [26]), or be used to approximate traditional hard cache capacity constraints.

Problem formulation. We aim to jointly optimize cache decisions and routing strategies to minimize the total cost. To construct a relaxation to this mixed-integer problem, suppose that variables $x_i(k)$ are independent Bernoulli random variables. Let ν be the corresponding joint probability distribution defined over matrices in $\{0,1\}^{|\mathcal{V}|\times|\mathcal{C}|}$, and denote by $\boldsymbol{P}_\nu[\cdot]$, $\mathbb{E}_\nu[\cdot]$ the probability and expectation w.r.t. ν, respectively. Let $y_i(k) \in [0,1]$ be the (marginal) probability that node i caches item k, namely, $y_i(k) = \boldsymbol{P}_\nu[x_i(k) = 1] = \mathbb{E}_\nu[x_i(k)]$, and let $\boldsymbol{y} = [y_i(k)]_{i\in\mathcal{V},k\in\mathcal{C}}$. Taking expectation w.r.t. ν, (2) becomes

$$y_i(k) + \sum\nolimits_{j\in\mathcal{V}} \phi_{ij}(k) = \begin{cases} 1, & \text{if } i \notin \mathcal{S}_k, \\ 0, & \text{if } i \in \mathcal{S}_k. \end{cases} \tag{3}$$

Let $f_{ij}(k)|_{\boldsymbol{y}} = \mathbb{E}_\nu[f_{ij}(k)]$ and $F_{ij}|_{\boldsymbol{y}} = \mathbb{E}_\nu[F_{ij}]$ denote the expected link flow rates, and $t_i(k)|_{\boldsymbol{y}} = \mathbb{E}_\nu[t_i(k)]$. Then, as long as (3) holds, we have $f_{ij}(k)|_{\boldsymbol{y}} = t_i(k)|_{\boldsymbol{y}}\phi_{ij}(k)$, and $t_i(k)|_{\boldsymbol{y}} = r_i(k) + \sum_{j\in\mathcal{V}} t_j(k)|_{\boldsymbol{y}}\phi_{ji}(k)$, sharing the same form as (1). Therefore, without ambiguity, we use F_{ij} to denote $F_{ij}|_{\boldsymbol{y}}$ in the rest of the paper, and let $Y_i = \sum_{k\in\mathcal{C}} y_i(k)$ denote the expected cache occupancy. The relaxed joint routing and content placement problem is cast as

$$\begin{aligned} \min_{\boldsymbol{\phi},\boldsymbol{y}} \quad & T(\boldsymbol{\phi},\boldsymbol{y}) = \sum_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}) + \sum_{i\in\mathcal{V}} B_i(Y_i) \\ \text{subject to} \quad & 0 \le \phi_{ij}(k) \le 1, \quad \forall (i,j)\in\mathcal{E}, k\in\mathcal{C} \\ & 0 \le y_i(k) \le 1, \quad \forall i\in\mathcal{V}, k\in\mathcal{C} \\ & \text{(3) holds.} \end{aligned} \tag{4}$$

Note that we do not explicitly impose any constraints for link or cache capacity in (4), since they are already incorporated in the cost functions. We tackle (4) first in a fixed-routing setting (Section III), and then a general dynamic-routing setting (Section IV).

III. Special case: fixed-routing

The fixed-routing case refers to scenarios where the routing path of a request is fixed or pre-determined regardless of caching schemes. Namely, if node i is not a designated server of item k, the request packets of k arriving at i can only be forwarded to one pre-defined next-hop neighbor of i. We denote such a next-hop node of i for k as $j_i(k)$.[4]

In the fixed-routing case, problem (4) reduces to

$$\begin{aligned} \min_{\boldsymbol{y}} \quad & T(\boldsymbol{y}) = \sum_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}) + \sum_{i\in\mathcal{V}} B_i(Y_i) \\ \text{subject to} \quad & 0 \le y_i \le 1, \quad \forall k\in\mathcal{C}, i\notin\mathcal{S}_k, \\ & \phi_{ij}(k) = \begin{cases} 1 - y_i, \text{ if } i\notin\mathcal{S}_k, j = j_i(k), \\ 0, \quad \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Let p_{vk} be the routing path from node v to a designated server $s_k \in \mathcal{S}_k$. Path p_{vk} is a node sequence $(p_{vk}^1, p_{vk}^2, \cdots, p_{vk}^{|p_{vk}|})$, where $p_{vk}^1 = v$, $p_{vk}^{|p_{vk}|} = s_k$, and $p_{vk}^{l+1} = j_{p_{vk}^l}(k)$ for $l = 1,\cdots,|p_{vk}| - 1$. We say $(i,j) \in p_{vk}$ for a link (i,j) if i and j are two consecutive nodes in p_{vk}. If $i \in p_{vk}$, let $l_{p_{vk}}(i)$ denote the position of i on path p_{vk}, i.e., $p_{vk}^{l_{p_{vk}}(i)} = i$. We assume every path p_{vk} is well-routed, i.e., no routing loop is formed, and no intermediate node is a designated server of k. Therefore, in terms of item k, the rate of request packets that are generated by node v and arrive at node i is given by $r_v(k)\prod_{l'=1}^{l_{p_{vk}}(i)-1}\left(1 - y_{p_{vk}^{l'}}(k)\right)$ if $i \in p_{vk}$, and 0 if $i \notin p_{vk}$. Then the link flow rates are given by

$$f_{ji}(k) = \sum\nolimits_{v:(i,j)\in p_{vk}} r_v(k)\prod\nolimits_{l'=1}^{l_{p_{vk}}(i)}\left(1 - y_{p_{vk}^{l'}}(k)\right). \tag{6}$$

We denote by $T(\mathbf{0})$ the cost when $\boldsymbol{y} = \mathbf{0}$, i.e., the total routing costs when no cache is deployed, and we assume $T(\mathbf{0})$ is finite. Then problem (5) is equivalent to maximizing a "caching gain" $G(\boldsymbol{y})$:

$$\begin{aligned} \max_{\boldsymbol{y}} \quad & G(\boldsymbol{y}) = A(\boldsymbol{y}) - B(\boldsymbol{y}) \\ \text{subject to} \quad & 0 \le y_i(k) \le 1, \quad \forall k\in\mathcal{C}, i\notin\mathcal{S}_k \end{aligned} \tag{7}$$

where $A(\boldsymbol{y})$ and $B(\boldsymbol{y})$ are given by

$$A(\boldsymbol{y}) = T(\mathbf{0}) - \sum\nolimits_{(i,j)\in\mathcal{E}} D_{ij}(F_{ij}), \quad B(\boldsymbol{y}) = \sum\nolimits_{i\in\mathcal{V}} B_i(Y_i).$$

Lemma 1. *Problem* (7) *is a "DR-submodular + concave" maximization problem. Specifically,* $A(\boldsymbol{y})$ *is non-negative monotonic DR-submodular* [5] *in* $\boldsymbol{y}$*, and* $B(\boldsymbol{y})$ *is convex in* $\boldsymbol{y}$*.*

[4]For simplicity, we only discuss fixed single-path routing. The solution can be seamlessly generalized to fixed multipath routing, i.e., when $\phi_{ij}(k)$ are fixed to non-integer values satisfying (2), by splitting a multipath request into multiple single-path requests.

[5]DR-submodular function is a continuous generalization of submodular functions with diminishing return. See [32] for more information.

Algorithm 1: Gradient-Combining Frank-Wolfe

Input: Integer $N > 1$
Result: Cache strategy $\boldsymbol{y}^{\text{out}}$ for fixed-routing case
Start with $n = 0$, let $\varepsilon = N^{-\frac{1}{3}}$.
Set $\boldsymbol{y}^{(0)}$ to be $y_i(k) = 0$ for all i,k.
do
  Let $\boldsymbol{s}^{(n)} = \arg\max_{\mathbf{0} \le \boldsymbol{y} \le \mathbf{1}} \left\langle \boldsymbol{y}, \nabla A\left(\boldsymbol{y}^{(n)}\right) - 2\nabla B(\boldsymbol{y}^{(n)}) \right\rangle$.
  Let $\boldsymbol{y}^{(n+1)} = (1-\varepsilon^2)\boldsymbol{y}^{(n)} + \varepsilon^2 \boldsymbol{s}^{(n)}$.
for $n = 0, 1, \cdots, N-1$;
Find the best among $\left\{\boldsymbol{y}^{(0)}, \cdots, \boldsymbol{y}^{(N)}\right\}$, let $\boldsymbol{y}^{\text{out}} = \arg\max_{\boldsymbol{y} \in \left\{\boldsymbol{y}^{(0)}, \cdots, \boldsymbol{y}^{(N)}\right\}} G(\boldsymbol{y})$.

DR-submodular + concave maximization problems have been systematically studied recently by Mitra et al. [30]. Problem (7) falls into one of the categories in [30], where a Gradient-Combining Frank-Wolfe (GCFW) algorithm (Algorithm 1) guarantees a $(\frac{1}{2}, 1)$ approximation, i.e., the solution achieves a $\frac{1}{2}$ ratio of the submodular part and a 1 ratio of the concave part of the objective at the optimal solution.

Theorem 1 (Theorem 3.10 [30])**.** *We assume G is L-smooth, i.e., ∇G is Lipschitz continuous. For $N > 1$, let $\boldsymbol{y}^*$ be an optimal solution to* (7) *and $\boldsymbol{y}^{out}$ be the solution generated by Algorithm 1, then it holds that*

$$G(\boldsymbol{y}^{out}) \ge \frac{1-\varepsilon}{2} A(\boldsymbol{y}^*) - B(\boldsymbol{y}^*) - \varepsilon \cdot O\left(L|\mathcal{V}||\mathcal{C}|\right).$$

where L is the Lipschitz constant of ∇G, $\varepsilon = N^{-1/3}$ and N is the total step number.

By (6), the gradient $\nabla B(\boldsymbol{y})$ in Algorithm 1 can be calculated as $\frac{\partial B(\boldsymbol{y})}{\partial y_z(k)} = B_z'(Y_z)$, and $\nabla A(\boldsymbol{y})$ is given by

$$\frac{\partial A(\boldsymbol{y})}{\partial y_z(k)} = t_z(k) \sum\nolimits_{(i,j) \in p_{zk}} D_{ji}'(F_{ji}) \prod\nolimits_{l'=2}^{l_{p_{zk}}(i)} \left(1 - y_{p_{zk}^{l'}}(k)\right).$$

The linear program in Algorithm 1 can be solved by selecting z and k with $\frac{\partial A(\boldsymbol{y})}{\partial y_z(k)} - 2\frac{\partial B(\boldsymbol{y})}{\partial y_z(k)} > 0$ and letting corresponding elements in $\boldsymbol{s}^{(n)}$ be 1, while keeping others 0.

IV. General case: dynamic-routing

The analysis in Section III unfortunately does not apply to the general dynamic-routing case (i.e., when routing is dynamically and adaptively adjusted jointly with caching), since the DR-submodularity no longer holds when $\boldsymbol{\phi}$ is not fixed. In this section, we tackle the general case of problem (4) with a node-based perspective first used in [1] and subsequently in [14], [33]. We first present a set of KKT necessary optimality conditions for (4), then give a modification to the KKT conditions that overcome certain saddle points. We show that the modified conditions yield a bounded gap from the global optimum, then provide further discussion and corollaries.

KKT necessary condition. Following [1], we start by giving closed-form partial derivatives of $T(\boldsymbol{\phi}, \boldsymbol{y})$. For caching strategy $\boldsymbol{y}$, it holds that $\frac{\partial T}{\partial y_i(k)} = B_i'(Y_i)$.

For routing strategy $\boldsymbol{\phi}$, the marginal cost due to increase of $\phi_{ij}(k)$ equals a sum of two parts, (1) the marginal cost due to increase of F_{ji} since more responses are sent from j to i, and (2) the marginal cost due to increase of $r_j(k)$ since node j needs to handle more request packets. Formally,[6]

$$\frac{\partial T}{\partial \phi_{ij}(k)} = t_i(k) \left(D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)} \right), \tag{8}$$

where the term $\partial T / \partial r_i(k)$ is the marginal cost for i to handle unit rate increment of request packets for k. This equals a weighted sum of marginal costs on out-going links and neighbors, namely,

$$\frac{\partial T}{\partial r_i(k)} = \sum\nolimits_{j \in \mathcal{N}(i)} \phi_{ij}(k) \left(D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)} \right). \tag{9}$$

By (3), value of $\partial T / \partial r_i(k)$ is implicitly affected by $y_i(k)$, e.g., it holds that $\partial T / \partial r_i(k) = 0$ if $i \in \mathcal{S}_k$ or $y_i(k) = 1$.[7]

Theorem 2. *Let $(\boldsymbol{\phi}, \boldsymbol{y})$ be an optimal solution to problem* (4)*, then for any $i \in \mathcal{V}$, $j \in \mathcal{N}(i)$ and $k \in \mathcal{C}$,*

$$\begin{aligned} B_i'(Y_i) &\begin{cases} = \lambda_{ik}, \text{ if } y_i(k) > 0, \\ \ge \lambda_{ik}, \text{ if } y_i(k) = 0, \end{cases} \\ t_i(k) \left(D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)} \right) &\begin{cases} = \lambda_{ik}, \text{ if } \phi_{ij}(k) > 0, \\ \ge \lambda_{ik}, \text{ if } \phi_{ij}(k) = 0, \end{cases} \end{aligned} \tag{10}$$

where λ_{ik} is given by

$$\lambda_{ik} = \min \left\{ B_i'(Y_i), \min_{j \in \mathcal{N}(i)} t_i(k) \left(D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)} \right) \right\}. \tag{11}$$

Theorem 2 presents the KKT necessary conditions for problem (4). The proof is omitted due to space limitation.

Modified condition. Note that condition (10) is not sufficient for global optimality (even for the pure-routing problem, i.e., with $\boldsymbol{y}$ fixed to $\mathbf{0}$). A counterexample is provided in [1]. A cause of such non-sufficiency is the degenerate case where $t_i(k) = 0$, in which λ_{ik} is always 0 and (10) always holds, regardless of routing strategies $[\phi_{ij}(k)]_{j \in \mathcal{V}}$. Note that when $t_i(k) = 0$, it is always optimal to set $y_i(k) = 0$ since no request packets for item k ever arrive at node i. Note also that $t_i(k)$ appears repeatedly in (11) for all $j \in \mathcal{N}(i)$. To this end, we propose condition (12) as a modification to (10), and show that a bounded gap on the total cost is promised if (12) holds.

Theorem 3. *Let $(\boldsymbol{\phi}, \boldsymbol{y})$ be feasible to problem* (4)*, such that for all $i \in \mathcal{V}$, $j \in \mathcal{N}(i)$ and $k \in \mathcal{C}$,*

$$\begin{aligned} B_i'(Y_i) &\begin{cases} = t_i(k)\delta_i(k), & \text{if } y_i(k) > 0, \\ \ge t_i(k)\delta_i(k), & \text{if } y_i(k) = 0, \end{cases} \\ D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)} &\begin{cases} = \delta_i(k), & \text{if } \phi_{ij}(k) > 0, \\ \ge \delta_i(k), & \text{if } \phi_{ij}(k) = 0, \end{cases} \end{aligned} \tag{12}$$

[6]To formally derive (8) and (9) in detail, please refer to Theorem 2 in [1].
[7]If no routing loops are formed, $\partial T / \partial r_i(k)$ can be computed recursively by (9), staring from nodes $i \in \mathcal{S}_k$ or with $y_i(k) = 1$.

where $\delta_i(k)$ is given by [8]

$$\delta_i(k) = \min\left\{\frac{B_i'(Y_i)}{t_i(k)}, \min_{j\in\mathcal{N}(i)}\left(D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)}\right)\right\}. \tag{13}$$

Let $(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger)$ be any feasible solution to (4)*. Then,*

$$T(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger) - T(\boldsymbol{\phi}, \boldsymbol{y}) \geq \sum_{i\in\mathcal{V}}\sum_{k\in\mathcal{C}} \delta_i(k)\left(y_i(k) - y_i^\dagger(k)\right)\left(t_i^\dagger(k) - t_i(k)\right). \tag{14}$$

Proof sketch. It holds that $T(\boldsymbol{\phi}, \boldsymbol{y}) = D(\boldsymbol{\phi}) + B(\boldsymbol{y})$ where $D(\boldsymbol{\phi}) \equiv \sum_{(i,j)} D_{ij}(F_{ij})$ and $B(\boldsymbol{y}) \equiv \sum_i B_i(Y_i)$. It can be shown that $B(\boldsymbol{y})$ is convex in $\boldsymbol{y}$, and $D(\boldsymbol{\phi})$ is geodesically convex in $\boldsymbol{\phi}$ if $\boldsymbol{r} > \mathbf{0}$. For any feasible $(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger)$, following the technique in Theorem 3 [1], we show that

$$\sum_{(j,i)} D_{ji}'(F_{ji})\left(F_{ji}^\dagger - F_{ji}\right) \geq \sum_{i,k}\left(y_i(k) - y_i^\dagger(k)\right) t_i^\dagger(k)\delta_i(k).$$

By the convexity of $B(\boldsymbol{y})$, we show that

$$\sum_i B_i'(Y_i)\left(Y_i^\dagger - Y_i\right) \geq \sum_{i,k} t_i(k)\delta_i(k)\left(y_i^\dagger(k) - y_i(k)\right).$$

Then (14) holds by combining the above. □

To the best of our knowledge, (14) is the first provable bound for minimizing the sum of a convex function and a geodesic convex function. Note that (12) implies $y_i(k) = 0$ if $t_i(k) = 0$, since the increasing and convex assumption of $B_i(\cdot)$ requires $B_i'(Y_i) > 0$ if $Y_i > 0$. Condition (12) is a more restrictive version of the necessary condition (10). Any feasible $(\boldsymbol{\phi}, \boldsymbol{y})$ satisfying (12) must also satisfy (10).

Unlike [1], [33], condition (12) is still not sufficient for global optimality. Nevertheless, it is practically efficient to minimize the total cost in a distributed manner according to (12). We next provide further discussion upon condition (12).

Intuitive interpretation. To provide an intuitive interpretation of the modified condition, let $\delta_{ij}(k)$ denote the marginal cost due to increase of flow rate $f_{ji}(k)$, that is, the marginal cost if node i forwards additional requests of unit rate to node j. Then similar to (8) and (9), $\delta_{ij}(k)$ is given by

$$\delta_{ij}(k) = \frac{\partial T}{\partial f_{ji}(k)} = D_{ji}'(F_{ji}) + \frac{\partial T}{\partial r_j(k)}. \tag{15}$$

Next, we define a virtual *cached flow* as $f_{i0}(k) = t_i(k)y_i(k)$, i.e., the expected rate of request packets for item k that terminate at node i due to i's caching strategy. Let $\delta_{i0}(k)$ denote the marginal cost due to increase of $f_{i0}(k)$, namely,

$$\delta_{i0}(k) = \frac{\partial T}{\partial f_{i0}(k)} = \frac{\partial T}{t_i(k)\partial y_i(k)} = \frac{B_i'(Y_i)}{t_i(k)}. \tag{16}$$

By (3), $\delta_{i0}(k)$ gives the marginal cache deployment cost if i wishes to increase $y_i(k)$ so that the total request packets forwarded to its neighbors is reduced by unit rate. Therefore, by (13), we have

$$\delta_i(k) = \min_{j\in\{0\}\cup\mathcal{N}(i)} \delta_{ij}(k). \tag{17}$$

[8] In the calculation of $\delta_i(k)$, we assume $B_i'(Y_i)/t_i(k) = \infty$ if $t_i(k) = 0$.

That is, $\delta_i(k)$ gives the minimum marginal cost for node i to handle request packets for item k. Condition (12) then suggests that each node handles incremental arrival requests in the way that achieves its minimum marginal cost – either by forwarding to neighbors, or by expanding its own cache. In other words, we say it is "worthwhile" to deploy cache for k at i if $\delta_{i0}(k) < \min_{j\in\mathcal{N}(i)}\delta_{ij}(k)$, and "not worthwhile" otherwise.

Even though condition (12) is neither a necessary condition nor a sufficient condition, Corollary 1 implies that it must have non-empty intersection with the global optima of (4).

Corollary 1. *For any optimal solution $(\boldsymbol{\phi}^*, \boldsymbol{y}^*)$ to* (4)*, there exists a corresponding $(\boldsymbol{\phi}, \boldsymbol{y})$ satisfying condition* (12)*, such that $\boldsymbol{y} = \boldsymbol{y}^*$ and $\phi_{ij}(k) = \phi_{ij}^*(k)$ for all i, k with $t_i^*(k) > 0$.*

Proof sketch. The corresponding $(\boldsymbol{\phi}, \boldsymbol{y})$ can be constructed by letting $\boldsymbol{y} = \boldsymbol{y}^*$, and $\phi_{ij}(k) = \phi_{ij}^*(k)$ for i, k such that $t_i^*(k) > 0$. For i, k with $t_i^*(k) = 0$, pick one $j \in \mathcal{N}(i)$ such that $j \in \arg\min_{j'\in\mathcal{N}(i)}\left(D_{j'i}'(F_{j'i}^*) + \frac{\partial T^*}{\partial r_{j'}(k)}\right)$ and let $\phi_{ij}(k) = 1$. □

Corollary 2. *Let $(\boldsymbol{\phi}, \boldsymbol{y})$ be a feasible solution to* (4) *and satisfy* (12)*. Let $(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger)$ be a feasible solution to* (4)*, such that for all $i \in \mathcal{V}$ and $k \in \mathcal{C}$, either $y_i^\dagger(k) = y_i(k)$ or $t_i^\dagger(k) = t_i(k)$. Then it holds that $T(\boldsymbol{\phi}, \boldsymbol{y}) \leq T(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger)$.*

Corollary 2 follows from Theorem 3. It implies that (12) is sufficient for optimizing $\boldsymbol{\phi}$ when $\boldsymbol{y}$ is fixed, and for optimizing $\boldsymbol{y}$ when $t_i(k)$ are unchanged. An example is shown in Fig. 2, where the caches always receive the same amount of request packets (i.e., unchanged $t_i(k)$), and the routers can not cache at all (i.e., unchanged $y_i(k)$).

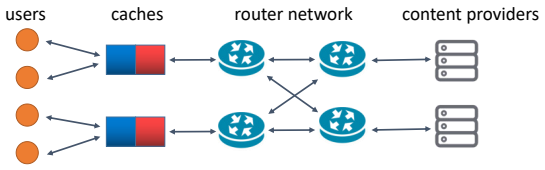

Fig. 2: A special scenario that (12) yields a global optimal solution. Single-layered caches are equipped near users.

Corollary 3. *Let $(\boldsymbol{\phi}, \boldsymbol{y})$ be a feasible solution to* (4) *and satisfy* (12)*. Let $(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger)$ be a feasible solution to* (4)*, such that either $\boldsymbol{\phi}^\dagger \geq \boldsymbol{\phi}$ or $\boldsymbol{\phi}^\dagger \leq \boldsymbol{\phi}$.*[9] *Then it holds that $T(\boldsymbol{\phi}, \boldsymbol{y}) \leq T(\boldsymbol{\phi}^\dagger, \boldsymbol{y}^\dagger)$.*

Corollary 3 also follows from Theorem 3. For i, k such that $y_i(k) \neq 1$, let $\rho_{ij}(k) = \phi_{ij}(k)/(1 - y_i(k))$ be the *conditional routing variable*, i.e., the probability of a request packet being forwarded to j given the requested item is not cached at i. In practical networks, the routing and caching mechanisms are usually implemented separately, and the routing is only based on $\rho_{ij}(k)$ instead of $\phi_{ij}(k)$. Corollary 3 contains a special case where $\rho_{ij}^\dagger(k) = \rho_{ij}(k)$ for all i, j, k, but $y_i(k)^\dagger \geq y_i(k)$ for all i, k (or $y_i(k)^\dagger \leq y_i(k)$ for all i, k). This special case

[9] For $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ of same dimension, we denote by $\boldsymbol{v}_1 \geq \boldsymbol{v}_2$ if every element of $\boldsymbol{v}_1$ is no less than the corresponding element in $\boldsymbol{v}_2$. Similarly as $\boldsymbol{v}_1 \leq \boldsymbol{v}_2$.

implies that if condition (12) is met, the total cost cannot be lowered by only caching more items (i.e., only increasing $y_i(k)$ for some i and k), or only removing items from caches (i.e., only decreasing $y_i(k)$ for some i and k), while keeping the conditional routing variables unchanged.

V. Online algorithm

Although Algorithm 1 achieves a constant factor approximation for the fixed-routing case, it is offline and centralized. In practical networks, request patterns are unknown a prior and may vary with time, thus an online adaptive and preferably distributed algorithm is needed. In this section, we present a distributed online algorithm for the general dynamic routing case that converges to condition (12). The algorithm does not require prior knowledge of exogenous rates $r_i(k)$ and designated servers $\mathcal{S}_k$, and is adaptive to slow variation in $r_i(k)$ and cost metrics $D_{ij}(\cdot)$, $B_i(\cdot)$.

Algorithm overview. We partition time into *periods* of duration $L * T_{\text{slot}}$. A period consists of L *slots*, each of duration T_{slot}. In t-th period, node i keeps its routing and caching strategies $(\boldsymbol{\phi}_i^t, \boldsymbol{y}_i^t)$ unchanged. At the m-th slot of t-th period ($1 \le m \le L$), node i rounds $\boldsymbol{y}_i^t$ into integer caching decisions $\boldsymbol{x}_i^{t,m}$ with $\mathbb{E}[\boldsymbol{x}_i^{t,m}] = \boldsymbol{y}_i^t$.[10] The last slot in each period is called *update slot*, during which nodes update their routing and caching strategies in a distributed manner. We postpone the discussion of randomized rounding techniques to Section V, and now focus on the update of strategies $\boldsymbol{\phi}^t$ and $\boldsymbol{y}^t$.

Our algorithm is a gradient projection variant. Each node updates its strategies during the update slot of t-th period by

$$\boldsymbol{\phi}_i^{t+1} = \boldsymbol{\phi}_i^t + \Delta\boldsymbol{\phi}_i^t, \quad \boldsymbol{y}_i^{t+1} = \boldsymbol{y}_i^t + \Delta\boldsymbol{y}_i^t. \tag{18}$$

The update vectors $\Delta\boldsymbol{\phi}_i^t$ and $\Delta\boldsymbol{y}_i^t$ are calculated by

$$\Delta\phi_{ij}^t(k) = \begin{cases} -\phi_{ij}^t(k), & \text{if } j \in \mathcal{B}_i^t(k) \\ -\min\left\{\phi_{ij}^t(k), \alpha e_{ij}^t(k)\right\}, & \text{if } j \notin \mathcal{B}_i^t(k),\ e_{ij}^t(k) > 0 \\ S_i^t(k)/N_i^t(k), & \text{if } j \notin \mathcal{B}_i^t(k),\ e_{ij}^t(k) = 0 \end{cases}$$

$$\Delta y_i^t(k) = \begin{cases} -\min\left\{y_i^t(k), \alpha e_{i0}^t(k)\right\}, & \text{if } e_{i0}^t(k) > 0 \\ S_i^t(k)/N_i^t(k), & \text{if } e_{i0}^t(k) = 0 \end{cases} \tag{19}$$

where $\mathcal{B}_i^t(k)$ is the set of *blocked nodes* to suppress routing loops, α is the stepsize, and[11]

$$\begin{aligned} &e_{i0}^t(k) = \delta_{i0}^t(k) - \delta_i^t(k), \quad e_{ij}^t(k) = \delta_{ij}^t(k) - \delta_i^t(k),\ \forall j \notin \mathcal{B}_i^t(k), \\ &N_i^t(k) = \left| \left\{ j \in \mathcal{N}(i) \backslash \mathcal{B}_i^t(k) \middle| e_{ij}^t(k) = 0 \right\} \right| + \mathbb{1}_{\delta_{i0}^t(k) > 0}, \\ &S_i^t(k) = \sum\nolimits_{j \notin \mathcal{B}_i^t(k) : e_{ij}^t(k) > 0} \Delta\phi_{ij}^t(k) + \Delta y_i^t(k) \mathbb{1}_{\delta_{i0}^t(k) > 0}, \end{aligned} \tag{20}$$

The intuitive idea is to transfer routing/caching fractions from non-minimum-marginal directions to the minimum-marginal ones. $\delta_{ij}^t(k)$ and $\delta_{i0}^t(k)$ are calculated as in (15) and (16). But slightly different from (17), due to the existence of $\mathcal{B}_i^t(k)$,

$$\delta_i^t(k) = \min\left\{\delta_{i0}^t(k), \min\nolimits_{j \in \mathcal{N}(i) \backslash \mathcal{B}_i^t(k)} \delta_{ij}^t(k)\right\}. \tag{21}$$

[10] We suggest refreshing caching decisions multiple times in each period to better estimate theoretical costs and marginals from actual measurements. Nevertheless, the algorithm applies to any $L \ge 1$.

[11] $\mathbb{1}_A$ is the indicator function of A. i.e., $\mathbb{1}_A = 1$ if A is true, and 0 if not.

Algorithm 2: Gradient Projection (GP)

Input: Loop-free $(\boldsymbol{\phi}^0, \boldsymbol{y}^0)$ with $T^0 < \infty$, stepsize α
Start with $t = 0$.
do
  Each node i round $\boldsymbol{y}_i^t$ into $\boldsymbol{x}_i^{t,m}$ with Distributed Randomized Rounding (DRR).
during *the m-th slot of t-th period*;
do
  Each node updates $\partial T/\partial r_i(k)$ for all k via a a message broadcasting mechanism.
  Each node calculates (20).
  Each node updates strategies $(\boldsymbol{\phi}_i^t, \boldsymbol{y}_i^t)$ by (18) (19).
during *update slot of t-th period*;

In each update slot, to calculate $\delta_{ij}^t(k)$ and $\delta_{i0}^t(k)$, the value $\partial T/\partial r_i(k)$ is updated throughout the network with a control message broadcasting mechanism (see, e.g., [33]). Specifically, node i receives $\partial T/\partial r_j(k)$ from all downstream neighbors (i.e., the nodes $j \in \mathcal{N}(i)$ with $\phi_{ij}(k) > 0$), calculates[12] its $\partial T/\partial r_i(k)$ according to (9), and broadcasts $\partial T/\partial r_i(k)$ to all upstream neighbors. Such broadcast starts at the designated servers or nodes with $y_i(k) = 1$, where $\partial T/\partial r_i(k) = 0$. The proposed algorithm is summarized in Algorithm 2. Next, we discuss the set $\mathcal{B}_i^t(k)$.

Loops and blocked nodes. A routing loop refers to node sequence $(l_1, l_2, \cdots, l_{|l|})$, such that $l_1 = l_{|l|}$ and for some $k \in \mathcal{C}$, $\phi_{l_p l_{p+1}}(k) > 0$ for $p = 1, \cdots, |l| - 1$. A loop implies that a strictly positive portion of requests for item k forwarded from node l_1 is sent back to l_1 itself. The existence of loops should be forbidden, as it gives rise to redundant flow circulation and wastes network resources. Loop-free routing can be guaranteed by a higher layer [16][26] (e.g., FIB in ICN). If so, we simply set $\mathcal{B}_i^t(k) = \mathcal{V} \backslash \mathcal{N}(i)$ for all t and k.

In case loop-free routing is not guaranteed by a higher layer protocol (e.g., in ad-hoc networks), our algorithm can still prevent the formation of loops by employing a method called "blocked node set" $\mathcal{B}_i^t(k)$, assuming a loop-free initial state $\boldsymbol{\phi}^0$ is given. Specifically, during $(t+1)$-th period, node i should not forward any request of item k to nodes in the set $\mathcal{B}_i^t(k)$. The construction of sets $\mathcal{B}_i^t(k)$ falls into two types, *static* and *dynamic*. We next introduce both.

(1) *Static sets.* The blocked node sets can be pre-determined and kept unchanged throughout the algorithm, i.e., $\mathcal{B}_i^t(k) = \mathcal{B}_i(k)$ for all $t \ge 0$. A directed acyclic subgraph of $\mathcal{G}$ is constructed for every item at the beginning of the algorithm, in which every node has at least one path to a designated server. We denote the subgraph w.r.t. $k \in \mathcal{C}$ as $\mathcal{G}_{(k)} = (\mathcal{V}, \mathcal{E}_{(k)})$ with $\mathcal{E}_{(k)} \subseteq \mathcal{E}$. Then the blocked node sets are constructed as $\mathcal{B}_i(k) = \left\{ j \in \mathcal{N}(i) \middle| (i,j) \notin \mathcal{E}_{(k)} \right\}$.

The idea of static blocked node set is commonly adopted, e.g., in the FIB construction of ICN. The subgraphs $\mathcal{G}_{(k)}$ can

[12] Node i should estimate $B_i'(Y_i)$ and $D_{ij}'(F_{ij})$ from Y_i and F_{ij}. Flow rate F_{ij} is measured by averaging the first $(L-1)$ slots of t-th period.

be calculated efficiently at the network initialization [6], either in a centralized way (e.g., Bellman-Ford algorithm), or in a distributed manner (e.g., distance-vector protocol).

(2) *Dynamic sets.* The sets $\mathcal{B}_i^t(k)$ can also be dynamically calculated as the algorithm proceeds. Compared with the fixed case, dynamically determined sets may give nodes more routing options and, therefore, potentially better performance in terms of total cost. It requires a more elaborate node blocking mechanism, preferably distributed and efficient. A classic dynamic node blocking mechanism is presented in [1] for a multi-commodity routing problem,

Convergence. Since the node blocking mechanism is implemented to suppress loops, Algorithm 2 may not converge to condition (12) if solution $(\boldsymbol{\phi}, \boldsymbol{y})$ satisfying (12) contains loops. Nevertheless, Theorem 4 states that the convergence limit still satisfies a version of (12) with $\mathcal{N}(i)$ replaced by $\mathcal{N}(i)\backslash\mathcal{B}_i(k)$.

Theorem 4. *Assume the network starts at $(\boldsymbol{\phi}^0, \boldsymbol{y}^0)$ with $T^0 < \infty$, and $(\boldsymbol{\phi}^t, \boldsymbol{y}^t)$ are updated by Algorithm 2 with a sufficiently small stepsize α. Then, if static blocked node sets are used, the sequence $\left\{(\boldsymbol{\phi}^t, \boldsymbol{y}^t)\right\}_{t=0}^{\infty}$ converges to a limit point $(\boldsymbol{\phi}, \boldsymbol{y})$, and $(\boldsymbol{\phi}, \boldsymbol{y})$ satisfies* (12)*, with $\mathcal{N}(i)$ being replaced by $\mathcal{N}(i)\backslash\mathcal{B}_i(k)$.*

Proof sketch. The proof follows the outline of Theorem 2 in [34]. We first show with sufficiently small stepseize, it holds $T^{t+1} < T^t$, unless condition (12) with $\mathcal{N}(i)\backslash\mathcal{B}_i(k)$ is satisfied. This is due to the convexity of T w.r.t. a single $\boldsymbol{\phi}_i(k)$ given all other variables fixed. The stepsize can be found in [1], [35]. Then, since the feasible set can be shown compact, the sequence $\left\{(\boldsymbol{\phi}^t, \boldsymbol{y}^t)\right\}_{t=0}^{\infty}$ must have a limit point $(\boldsymbol{\phi}, \boldsymbol{y})$, at where the objective T can not be further (strictly) improved. Therefore, condition (12) with $\mathcal{N}(i)\backslash\mathcal{B}_i(k)$ must hold. □

Algorithm complexity. The computational complexity of GP is $O(|\mathcal{V}|^2|\mathcal{C}|)$ for each update (for each node $O((|d_{\max}|+1)|\mathcal{C}|)$, with space complexity $O((|d_{\max}| + 1)|\mathcal{C}|))$, where $d_{\max}$ is the largest out-degree. An additional $O(|\mathcal{V}|^2|\mathcal{C}|)$ occurs if dynamic blocking is used. As a comparison, typical routing methods have complexity $O(|\mathcal{V}|^2|\mathcal{C}|)$ and caching algorithm [4] has complexity $O(|\mathcal{C}|\log|\mathcal{C}|)$. Thus the complexity is not increased relative to alternating optimization for routing and caching. During each update slot, node i should calculate $\partial T/\partial r_i(k)$ for all k. When the network is large, control messages carrying derivative information may take non-negligible time and bandwidth to percolate through the network. There are $|\mathcal{E}|$ transmissions of broadcast messages corresponding to an item in one update slot, and totally $|\mathcal{C}||\mathcal{E}|$ transmissions each update slot, with on average $|\mathcal{C}|/T_{\text{slot}}$ per link·second and at most $d_{\max}|\mathcal{C}|$ per node. The broadcast messages have $O(1)$ size, and can be sent in an out-of-band channel. Let t_c be the maximum transmission time for a broadcast message, and $\bar{h}$ be the maximum hop number for a request path. Then the broadcast mechanism requires at most $\bar{h}t_c$ for each update.

We remark that single node computation workload is not affected by $|\mathcal{V}|$ if $d_{\max}$ is bounded by a constant. The update may fail if control message delay $t_c\bar{h}$ exceeds T_{slot}, or if per-second control message $|\mathcal{C}|/T_{\text{slot}}$ exceeds the control channel capacity. If so, we can use longer slots, or allow some nodes to perform updates every multiple periods. Meanwhile, if $|\mathcal{C}|$ is large, the computation and communication overhead can be significantly reduced by applying our algorithm only to the most popular items.

Distributed randomized rounding. The continuous caching strategy $\boldsymbol{y}$ is rounded to caching decision $\boldsymbol{x}$ in each slot. The rounding can be done with a naive probabilistic scheme, i.e., $x_i^{t,m}(k)$ is a Bernoulli random variable with $p = y_i^t(k)$. However, such a heuristic rounding method may generate large or drastically changing cache sizes. Various advanced rounding techniques exist (see [36]–[38]). If all Y_i are integer, the deterministic *pipage rounding* [37] and the randomized *swap rounding* [39] guarantee the actual routing cost after rounding is no worse than the relaxed result, while keeping $X_i = Y_i$. However, such techniques are centralized.

A distributed rounding method is proposed by [4], each node independently operates without knowledge of closed-form $T(\boldsymbol{\phi}, \boldsymbol{y})$. We extend this method to non-integer Y_i, and refer to it as Distributed Randomized Rounding (DRR). With such a rounding algorithm, it is guaranteed that the expected flow rates and cache sizes meet the relaxed value, and the actual cache size at each node is within 1 of the expected value.

Lemma 2. *If $\boldsymbol{x}^{t,m}$ are rounded from $\boldsymbol{y}^t$ by DRR, then*

$$\begin{aligned}
&\mathbb{E}[x_i^t(k)] = y_i^t(k), \quad \forall i \in \mathcal{V}, k \in \mathcal{C},\\
&\left|\sum\nolimits_{k\in\mathcal{C}} x_i^t(k) - \sum\nolimits_{k\in\mathcal{C}} y_i^t(k)\right| < 1, \quad \forall i \in \mathcal{V},\\
&\mathbb{E}[F_{ij}\big|_{(\boldsymbol{\phi}^t, \boldsymbol{x}^{t,m})}] = F_{ij}\big|_{(\boldsymbol{\phi}^t, \boldsymbol{y}^t)}, \quad \forall (i,j) \in \mathcal{E}.
\end{aligned}$$

The proof of Lemma 2 is omitted. Since $D_{ij}(\cdot)$ and $B_i(\cdot)$ are convex, combining with Jensen's Inequality, it holds that $\mathbb{E}\left[T(\boldsymbol{\phi}^t, \boldsymbol{x}^{t,m})\right] \geq T\left(\boldsymbol{\phi}^t, \mathbb{E}[\boldsymbol{x}^{t,m}]\right) = T(\boldsymbol{\phi}^t, \boldsymbol{y}^t)$. Suppose $|D_{ij}(x) - D_{ij}(y)| \leq M_{ij}|x-y|^\alpha$, the performance loss due to Jensen gap is bounded as $|\mathbb{E}[D_{ij}(F_{ij})] - D_{ij}(\mathbb{E}[F_{ij}])| \leq M_{ij}\sigma_\alpha^\alpha$, where σ_α is the α-moment of F_{ij}, determined by packet dispatching mechanism. We demonstrate in Section VI that, with a proper randomized packet forwarding mechanism, the costs measured in the real network will not deviate too much from the theoretical result $T(\boldsymbol{\phi}^t, \boldsymbol{y}^t)$.

VI. Simulation

Simulator setting. We simulate the proposed algorithms and other baseline methods in various network scenarios with a packet-level simulator available at [40]. We denote by $\mathcal{R}$ the set of requests in the network. For each request (i,k), the requester i is uniformly chosen in $\mathcal{V}$, and the requested item k is chosen in the catalog $\mathcal{C}$ with a Zipf-distribution of parameter 1.0. The exogenous request rates r_{ik} for all requests are uniformly random in interval $[1.0, 5.0]$. For each $(i,k) \in \mathcal{R}$, node i sends request packets for item k in a Poisson process of rate $r_i(k)$. For each item $k \in \mathcal{C}$, we assume $|\mathcal{S}_k| = 1$ and choose the designated server uniformly randomly in all nodes.

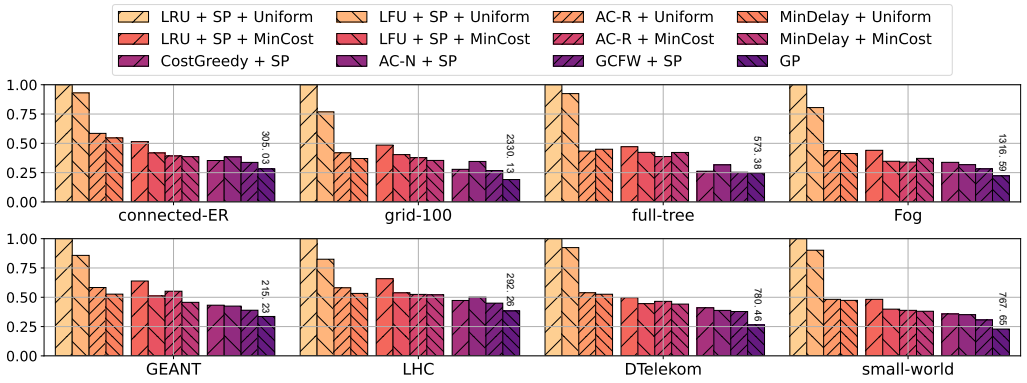

Fig. 3: Normalized total cost T across network scenarios. The actual delay values of GP for each case are marked.

To ensure relatively steady flow rates, we adopt a *token-based* randomized forwarding. i.e., node i keeps a token pool, where the number of tokens for neighbor j is in proportion to $\phi_{ij}(k)$. Each request arriving at i is forwarded consuming one token, and the token pool is refreshed when empty. We monitor the network status every T_{monitor}. Flows F_{ij} are measured by averaging during past T_{monitor}. We let $D_{ij}(F_{ij}) = d_{ij}F_{ij} + d_{ij}^2F_{ij}^2 + d_{ij}^3F_{ij}^3$, which is a 3-order expansion of $F_{ij}/(1/d_{ij} - F_{ij})$. We use $D'_{ij}(0) = d_{ij}$ as link weights, representing the unit-flow cost with no congestion, for non-congestion-dependent methods. We let $B_i(Y_i) = b_iY_i$, where b_i is the unit cache price at i. Parameters d_{ij} and b_i are uniformly selected from the interval in Table II.

Simulated scenarios and baselines. We simulate multiple synthetic or real-world network scenarios in Table II. **connected-ER** is a connectivity-guaranteed Erdős-Rényi graph, where bi-directional links exist for each pair of nodes with probability $p = 0.07$. **grid-100** and **grid-25** are 2-dimensional 10×10 and 5×5 grid networks. **full-tree** is a full binary tree of depth 6. **Fog** is a full 3-ary tree of depth 4, where children of the same parent is concatenated linearly [41]. This topology is dedicated to formulating fog-caching and computing networks. **GEANT** is a pan-European data network for the research and education community [42]. **LHC** (Large Hadron Collider) is a prominent data-intensive computing network for high-energy physics applications. **DTelekom** is a sample topology of Deutsche Telekom company [42]. **small-world** (Watts-Strogatz small world) is a ring graph with additional short-range and long-range edges.

We implement proposed **GCFW** (Algorithm 1), **GP** (Algorithm 2), and multiple baselines summarized in Table III. **LRU** (Least Recently Used [43]) and **LFU** (Least Frequently Used [44]) are traditional cache eviction algorithms. **SP** (Shortest Path) routes request packets on the shortest path to a designated server. **AC-R** (Adaptive Caching with Routing) is a joint routing/caching algorithm proposed by [6]. It uses probabilistic routing among top $k_{\text{SP}} = 3$ shortest paths. **MinDelay** is another hop-by-hop joint routing/caching algorithm with convex costs [14]. It uses the Frank-Wolfe algorithm with stepsize 1 for integer solutions. **Uniform** uniformly adds

TABLE II: Simulated network scenarios

Topologies	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{R}\rvert$	d_{ij}	b_i
connected-ER	50	256	80	200	[0.05, 0.1]	[5, 10]
grid-100	100	358	100	400	[0.05, 0.1]	[20, 40]
full-tree	63	124	50	150	[0.05, 0.1]	[20, 30]
Fog	40	130	50	200	[0.05, 0.1]	[30, 50]
GEANT	22	66	40	100	[0.05, 0.1]	[10, 15]
LHC	16	62	30	100	[0.1, 0.15]	[10, 15]
DTelekom	68	546	100	300	[0.1, 0.2]	[10, 20]
small-world	120	720	100	400	[0.05, 0.1]	[10, 20]
grid-25	25	80	30	100	0.1	10

TABLE III: Implemented algorithms and functionalities

Algorithm	cache deployment	content placement	routing
LRU/LFU		✓	
SP			✓
AC-R		✓	✓
MinDelay		✓	✓
Uniform	✓		
MinCost	✓		
CostGreedy	✓	✓	
AC-N	✓	✓	
GCFW	✓	✓	
GP	✓	✓	✓

cache capacities by 1 at all nodes in each period. **MinCost** is a heuristic cache deployment algorithm. It adds the cache capacity by 1 at the node with the highest total cache miss cost in every period[13]. **CostGreedy** is a heuristic joint cache deployment and content placement method. It greedily sets $y_i(k) = 1$ for the node-item pair (i, k) with the largest single-item cache miss cost in every period. **AC-N** (Adaptive Caching with Network-wide capacity constraint) is a cache deployment and content placement method with a network-wide cache budget [15]. We add budget by 1 and re-run AC-N each period.

We set $T_{\text{slot}} = 10$, $\boldsymbol{y}^0 = \mathbf{0}$, $L = 20$, and start with the shortest path. When using Uniform or MinCost, the corresponding content placement method is re-run every period to accommodate new cache capacities. For methods other than GCFW and GP, we run the simulation for a sufficient duration

[13] MinCost is in fact a cache miss cost-weighted cache hit maximization.

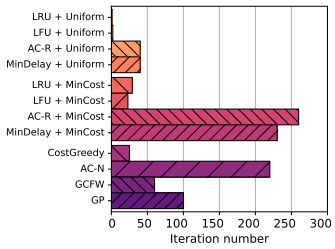

Fig. 4: Number of iterations till convergence in `grid-25`

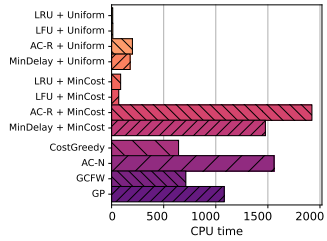

Fig. 5: CPU time till convergence in `grid-25`

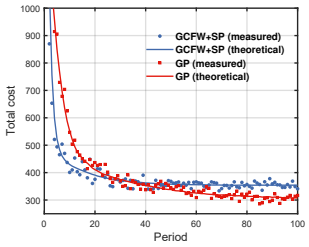

Fig. 6: Measured and theoretical costs in `grid-25`

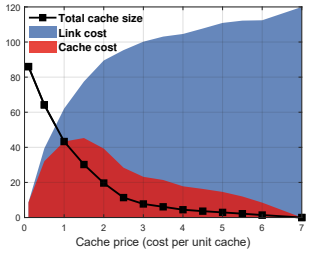

Fig. 7: Trade-off between cache cost and routing cost

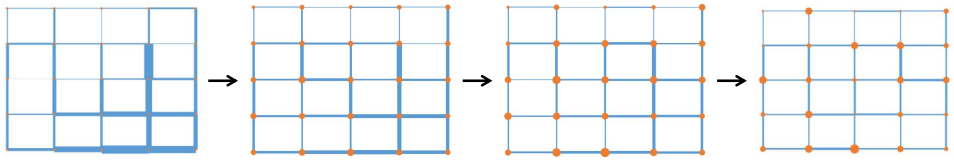

Fig. 8: Link flow and cache size evolution in `grid-25` by `GP`. Link width and node size are respectively in proportion to link flow and node cache size. t is the period number and T is the measured total cost.

and record the total cost of the lowest-cost period. For `GCFW` and `GP`, we measure steady-state total costs after convergence. For `GCFW`, we set $N = 100$. For `GP`, we use dynamic node blocking, set stepsize $\alpha = 0.01$ and run for $T_{\text{sim}} = 20000$.

Results and analysis.

(1) *Cost reduction.* We summarize the (normalized) measured total costs in Fig. 3. We divide the methods into three groups. Fig. 3 shows that the second group outperforms the first group, and is outperformed by the third group, implying that heuristic utility-based cache deployment methods are better than not optimizing, but can be further improved by jointly considering content placement and routing. The proposed algorithm `GP` outperforms other methods in all scenarios, reducing the total cost by up to 44.8% (on average 29.3%). Moreover, the improvement of `GP` is significant in scenarios with more routing choices (e.g., `grid-100`), and diminishes when routing choice is limited (e.g., `full-tree`).

(2) *Convergence.* To analyze the behavior of proposed algorithms, we present more refined experiments on the scenario `grid-25`. Fig. 4 and Fig. 5 compares the iteration number for convergence and algorithm CPU time for `grid-25`, respectively. They imply that `GP` consumes more time compared to simple heuristic caching policies (e.g., `LRU`), and less time compared to nested-looped alternating methods (e.g., `AC-N`).

Fig. 6 shows the convergence trajectory of measured and theoretical total cost by `GCFW+SP` and `GP` in `grid-25`. Measured cost refers to actual link costs measured on links and actual cache costs calculated from cache sizes after rounding. Theoretical cost refers to the flow-level cost T given by (4), calculated using the pre-given input rates $[r_i(k)]$ and $(\boldsymbol{\phi}^t, \boldsymbol{y}^t)$. The consistency of theoretical and measured costs implies that with continuous relaxation and rounding, our theoretical model accurately reflects the real network behavior.

(3) *Congestion mitigation.* The ability to mitigate network congestion is expected to be an important feature of the proposed algorithms, as non-linear traffic-dependent link costs are considered. Fig. 8 illustrates the evolution of link flows and cache sizes across the network as `GP` goes on. The requests and designated servers are randomly generated according to our previous assumptions and Table II. We observe that severe link congestion is gradually mitigated by properly tuning routing and caching strategies.

(4) *Routing-caching tradeoff.* As a fundamental motivation of this paper, we investigate the tradeoff between routing cost and cache deployment cost in `grid-25`. We set the link cost to be linear and plot in Fig. 7 the optimized link and cache costs as well as the corresponding total cache size against different unit cache cost b_i. We observe that, with a very high unit cache cost, no cache is deployed. As the unit cache cost dropping, the total cache size increases, the total cost decreases, and the cache cost takes a gradually more significant portion of the total cost.

VII. Conclusion

We minimize the sum of routing cost and cache deployment cost over caching and routing strategies for arbitrary topology and nonlinear costs. In the fixed-routing special case, the objective is a *DR-submodular + concave* function, and propose a Gradient-combining Frank-Wolfe algorithm with $(\frac{1}{2}, 1)$ approximation. For the general case, we propose the KKT condition and a modification. The modified condition suggests each node handles arrival requests in a way that achieves minimum marginal cost. We propose a distributed and adaptive online algorithm for the general case that converges to the modified condition. We demonstrate in simulation that our proposed algorithms significantly outperform baseline methods in multiple network scenarios.

REFERENCES

[1] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE transactions on communications*, vol. 25, no. 1, pp. 73–85, 1977.

[2] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[3] K. Poularakis and L. Tassiulas, "On the complexity of optimal content placement in hierarchical caching networks," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 2092–2103, 2016.

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

[5] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

[6] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 77–87.

[7] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1635–1648, 2016.

[8] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[9] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5g wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2995–3007, 2016.

[10] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[11] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, 2014.

[12] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *29th IEEE Conference on Decision and Control*. IEEE, 1990, pp. 2130–2132.

[13] E. 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 *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014.

[14] M. Mahdian and E. Yeh, "Mindelay: Low-latency joint caching and forwarding for multi-hop networks," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[15] V. S. Mai, S. Ioannidis, D. Pesavento, and L. Benmohamed, "Optimal cache allocation under network-wide capacity constraint," in *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019.

[16] H. Liu and R. Han, "A hierarchical cache size allocation scheme based on content dissemination in information-centric networks," *Future Internet*, vol. 13, no. 5, p. 131, 2021.

[17] J. Kwak, G. Paschos, and G. Iosifidis, "Elastic femtocaching: Scale, cache, and route," *IEEE Transactions on Wireless Communications*, 2021.

[18] J. Yao and N. Ansari, "Joint content placement and storage allocation in c-rans for iot sensing service," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1060–1067, 2018.

[19] B. Dai, Y.-F. Liu, and W. Yu, "Optimized base-station cache allocation for cloud radio access network with multicast backhaul," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1737–1750, 2018.

[20] X. Peng, J. Zhang, S. Song, and K. B. Letaief, "Cache size allocation in backhaul limited wireless networks," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[21] A. Liu and V. K. Lau, "How much cache is needed to achieve linear capacity scaling in backhaul-limited dense wireless networks?" *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 179–188, 2016.

[22] Y. Xu, Y. Li, T. Lin, Z. Wang, W. Niu, H. Tang, and S. Ci, "A novel cache size optimization scheme based on manifold learning in content centric networking," *Journal of Network and Computer Applications*, vol. 37, pp. 273–281, 2014.

[23] W. Chu, M. Dehghan, J. C. Lui, D. Towsley, and Z.-L. Zhang, "Joint cache resource allocation and request routing for in-network caching services," *Computer Networks*, vol. 131, pp. 1–14, 2018.

[24] M. Dehghan, L. Massoulie, D. Towsley, D. S. Menasche, and Y. C. Tay, "A utility optimization approach to network cache design," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1013–1027, 2019.

[25] R. T. Ma and D. Towsley, "Cashing in on caching: On-demand contract design with linear pricing," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, 2015, pp. 1–6.

[26] J. Ye, Z. Li, Z. Wang, Z. Zheng, H. Hu, and W. Zhu, "Joint cache size scaling and replacement adaptation for small content providers," in *IEEE Conference on Computer Communications*. IEEE, 2021.

[27] D. Carra, G. Neglia, and P. Michiardi, "Elastic provisioning of cloud caches: A cost-aware ttl approach," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1283–1296, 2020.

[28] G. Xiong, S. Wang, G. Yan, and J. Li, "Reinforcement learning for dynamic dimensioning of cloud caches: A restless bandit approach," *IEEE/ACM Transactions on Networking*, 2023.

[29] W. Chu, Z. Yu, J. C. Lui, and Y. Lin, "Jointly optimizing throughput and content delivery cost over lossy cache networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3846–3863, 2021.

[30] S. Mitra, M. Feldman, and A. Karbasi, "Submodular+ concave," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 577–11 591, 2021.

[31] D. Bertsekas and R. Gallager, *Data networks*. Athena Scientific, 2021.

[32] A. A. Bian, B. Mirzasoleiman, J. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 111–120.

[33] J. Zhang, Y. Liu, and E. Yeh, "Optimal congestion-aware routing and offloading in collaborative edge computing," in *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 2022, pp. 1–8.

[34] Y. Xi and E. M. Yeh, "Node-based optimal power control, routing, and congestion control in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 4081–4106, 2008.

[35] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[36] M. Mahdian, A. Moharrer, S. Ioannidis, and E. Yeh, "Kelly cache networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1130–1143, 2020.

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

[38] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[39] C. Chekuri, J. Vondrák, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 575–584.

[40] J. Zhang, "Elastic-Caching-Networks." [Online]. Available: https://github.com/JinkunZhang/Elastic-Caching-Networks

[41] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation scheduling, caching, and communication in data-intensive computing networks," *IEEE/ACM Transactions on Networking*, 2021.

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

[43] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.

[44] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, "Evaluating content management techniques for web proxy caches," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 3–11, 2000.