Joint Caching and Routing in Cache Networks With Arbitrary Topology

Tian Xie, *Student Member, IEEE*, Sanchal Thakkar, Ting He, *Senior Member, IEEE*, Patrick McDaniel, *Fellow, IEEE*, and Quinn Burke

***Abstract*—In-network caching and flexible routing are two of the most celebrated advantages of next generation network infrastructures. Yet few solutions are available for jointly optimizing caching and routing that provide performance guarantees for networks with arbitrary topology. We take a holistic approach towards this fundamental problem by analyzing its complexity in all the cases and developing polynomial-time algorithms with approximation guarantees in important special cases. We also reveal the fundamental challenge in achieving guaranteed approximation in the general case and propose an alternating optimization algorithm with good empirical performance and fast convergence. Our algorithms have demonstrated superior performance in both routing cost and congestion compared to the state-of-the-art solutions in evaluations based on real topology and request traces.**

***Index Terms*—Approximation algorithm, cache network, joint caching and routing, unsplittable flow problem.**

I. Introduction

As TWO of the most well-studied topics in computer communication networks, caching and routing play complementary roles: caching brings content closer to the users, and routing optimizes the performance of the communication paths. It is thus natural to explore the benefits of combining these control options via *joint caching and routing*.

While joint caching and routing applies to many network scenarios, it is particularly relevant in next generation networks which provide services beyond data transfer. For example, Information-Centric Network (ICN) promises to offer pervasive content caching at routers [2], [3], [4], and next generation cellular network proposes to offer content caching at various types of base stations [5], [6], [7], [8], [9], [10]. The challenge, however, is in solving the optimization problem designed to jointly optimize content placement and routing, which has received significant attention.

To this end, many tailor-made solutions have been developed for specific systems, e.g., a hierarchical IPTV system [11] or a heterogeneous cellular network with small-cell and macro-cell base stations [5], [6]. The hierarchical structure of these systems offers very limited routing options involving only a couple of hops, greatly simplifying the routing problem but making the solutions inapplicable in general networks.

Meanwhile, few works have addressed the fundamental problem of joint content placement and multi-hop routing in networks with arbitrary topology. Due to the huge solution space, most existing solutions either relied on heuristics or resorted to the generic branch-and-bound method with an exponential worst-case complexity [12]. Polynomial-time algorithms with approximation guarantees were not available until recently, when [3] proposed an approximation algorithm for minimizing the routing cost in the underloaded regime and [12] proposed an approximation algorithm for maximizing the number of served requests in the overloaded regime.

In this work, we address joint caching and routing in arbitrary topology, with the objective of minimizing the routing cost as in [3]. However, our work differs from [3] in that: (i) while [3] ignored link capacity constraints, we consider both limited and unlimited link capacities, where the limited link capacities significantly complicate the routing problem; (ii) while [3] only optimized routing among a limited set of candidate paths (e.g., k-shortest paths to origin servers), we optimize routing among all possible paths while maintaining a polynomial complexity. As shown later (Section VI), these differences allow our solutions to achieve substantially lower routing cost and link congestion.

A. Related Work

As caching and routing were each studied extensively with large numbers of related works, we will only review works addressing their joint optimization below.

Joint caching and routing: The problem of joint caching and routing has been studied in a number of network scenarios: ICN [2], [3], [4], Content Delivery Network (CDN) [13], [14], [15], [16], content provider networks [11], [17], cellular networks [5], [6], [7], [8], [9], [10], and IoT networks [12]. Majority of existing works focused on specific topologies, e.g., a 3-tier hierarchical topology [8], [9], [11], or a 2-tier hierarchical

Manuscript received 9 October 2022; revised 20 March 2023; accepted 9 May 2023. Date of publication 25 May 2023; date of current version 14 June 2023. This work was supported in part by National Science Foundation under Grants CNS-1946022 and CNS-2106294 and in part by Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement W911NF-13-2-0045 (ARL Cyber Security CRA). A preliminary version of this work was presented at ICDCS'22. [DOI: 10.1109/ICDCS54860.2022.00015]. Recommended for acceptance by L. Y. Chen. *(Corresponding author: Tian Xie.)*

Tian Xie, Sanchal Thakkar, and Ting He are with Pennsylvania State University, University Park, PA 16802 USA (e-mail: tbx5027@psu.edu; sjt5721@psu.edu; th.tinghe@gmail.com).

Patrick McDaniel and Quinn Burke are with the University of Wisconsin-Madison, Madison, WI 53706 USA (e-mail: mcdaniel@cse.psu.edu; qkb@cs.wisc.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPDS.2023.3276724, provided by the authors.

Digital Object Identifier 10.1109/TPDS.2023.3276724

topology [5], [6], [7]. These special topologies have very limited routing options, thus simplifying the problem.

Among works considering general network topology, only a few provided performance guarantees [3], [10], [12]. However, [3], [10] did not consider link capacity constraints, which greatly simplifies the routing problem as it suffices to route each request to the nearest replica of the requested content. While [12] considered link capacity constraints, it assumed an overloaded regime where not all requests can be served, and focused on maximizing the number of served requests. In contrast, we consider routing cost minimization in the underloaded regime as in [3], which represents the normal operation state of most networked systems, but we tackle a much more general problem than [3] and also improve it in the special case of unlimited link capacities.

Other related joint optimizations: Besides caching and routing, other joint optimizations have also been studied, e.g., joint content placement, server selection, and storage capacity allocation [18], and joint cache deployment, request routing, and content placement [14]. The content placement problem is also similar to the placement of virtual network functions, which is usually jointly optimized with routing [19], [20]. However, due to the high complexity of these problems, existing solutions are mostly based on heuristics. The few existing solutions that provide performance guarantees [21], [22], [23] address optimization problems that are very different from ours, and are thus not comparable with our work. Specifically, [21] optimized request rates and content placement, but assumed predetermined routes; [22] optimized VM allocation, content placement, and request routing, but ignored link capacities; [23] optimized request routing and content retention time, but only provided performance bounds in the case of uncapacitated caches.

B. Summary of Contributions

We consider the problem of joint caching and routing for minimizing the total routing cost under cache and link capacity constraints. After formulating the problem as a comprehensive optimization that covers both simple content replication (*integral caching*) with single-path routing (*integral routing*) and caching fractions of coded content (*fractional caching*) with multi-path routing (*fractional routing*), we make the following contributions:

1) We analyze the complexity of the optimization in all the cases through connections to known NP-hard problems.
2) We develop efficient algorithms, with focus on the hardest case of integral caching and integral routing. In the special case of unlimited link capacities, we develop a truly polynomial-time algorithm based on pipage rounding that achieves the same constant-factor approximation as the pseudo polynomial-time algorithm in [3]. In another special case of binary cache capacities, we reduce our problem to the minimum-cost single-source unsplittable flow problem (MSUFP) and develop a polynomial-time bicriteria approximation algorithm that improves the state-of-the-art MSUFP algorithm when each demand is much smaller than link capacities. We then apply the ideas from these special cases to develop a heuristic algorithm for the general case that alternatingly optimizes caching and routing.
3) We further extend the above solutions developed for equal-sized content items to the case of heterogeneous-sized items. While caching items of heterogeneous sizes can no longer be solved by pipage rounding, we show that the problem still has the desirable properties of submodular objective and p-independence constraints, which allows the greedy algorithm to achieve a constant approximation.
4) We evaluate our solutions against state-of-the-art benchmarks in the practical application scenario of edge caching. Our results based on real topology and request traces show that: (i) when given perfect knowledge of the demand, our algorithms can significantly improve the state-of-the-art solutions in both routing cost and congestion, (ii) the advantage remains when the decisions are based on predicted demand produced by a realistic prediction method, and (iii) dividing files into equal-sized chunks can significantly improve the performance of caching and routing.

Roadmap: Section II formulates our optimization problem, Section III analyzes its complexity, Section IV presents our algorithms and approximation analysis, Section V addresses the extension to heterogeneous item sizes, Section VI provides evaluation results, and finally Section VII concludes the paper. *All the proofs can be found in Appendix A of the supplementary file, available online.*

II. Problem Formulation

A. Network Model

We model the cache network as a directed graph $G = (V, E)$, where V is the set of nodes, and E the set of links. Collectively, the nodes serve a catalog C of content items (e.g., file chunks), which are assumed to be of equal size as in [3], [12], [21]; this assumption will be relaxed later (see Section V). To serve the content, each node v is equipped with a cache, which can store up to c_v content items ($c_v = 0$ if v has no cache). A node that does not store a content item can request it from other nodes. Each link $(u, v) \in E$ can transfer c_{uv} content items per unit time (assuming that the size of a request is negligible). We model each type of requests by a pair $(i, s) \in C \times V$, meaning that node s requests a content item i. Let $R \subseteq C \times V$ denote the set of all types of requests, and $\lambda_{(i,s)}$ (unit: requests per unit time) denote the arrival rate of requests of type (i, s).

In this work, we consider the under-loaded regime, where there are generally multiple ways to place and route content items such that all the requests can be satisfied. Our objective is to find a feasible solution that minimizes the total routing cost. To this end, we associate each link $(u, v) \in E$ with a routing cost $w_{uv} \geq 0$ (w_{uv} may not equal w_{vu}), denoting the cost of transferring a content item over this link. The routing cost can model any *additive metric*. For example, if w_{uv} denotes monetary cost (e.g., for leasing bandwidth), then minimizing the total routing cost minimizes the monetary cost in serving the requests; if w_{uv} denotes $-\log(\text{link reliability})$, then under the assumption of independent link failures, minimizing the total routing cost minimizes the average of $-\log(\text{path reliability})$, which maximizes the success rate of content retrieval. The specific choice of routing costs is not our focus; instead, our focus is on designing the *caching strategy* and the *routing strategy* based on a given cost per link such that the total routing cost can be minimized subject to the above resource constraints.

B. Model of Caching

We use x_{vi} to denote the caching decision regarding storing content item i at node v. If a content item can only be replicated as a whole, we require *integral caching* $x_{vi} \in \{0, 1\}$ (1: storing the item, 0: not storing the item). If a cache can store (a coded version of) a fraction of an item, we allow *fractional caching* $x_{vi} \in [0, 1]$. For example, using random linear code, we can divide each item into small chunks and store linear combinations of these chunks at caches such that the original item can be recovered with high probability as long as sufficiently many coded chunks are retrieved [24], where x_{vi} denotes the fraction of coded chunks for content item i that are stored at node v.

C. Model of Routing

Due to the possibility of multiple nodes storing a requested item, the routing decision contains both *source selection* that selects the source(s) to retrieve the content from, and *routing* that selects the path(s) to retrieve the content through. Depending on how items are cached, we may require *integral source selection* $r_v^{(i,s)} \in \{0, 1\}$, where $r_v^{(i,s)} = 1$ indicates that v is selected as the only source for serving request (i, s), or we may allow *fractional source selection* $r_v^{(i,s)} \in [0, 1]$, where $r_v^{(i,s)}$ is the fraction of item i served from node v to node s. Depending on whether multi-path routing is supported, we may require *integral routing* $f_{uv}^{(i,s)} \in \{0, 1\}$, where $f_{uv}^{(i,s)} = 1$ indicates that link (u, v) is on the only path serving request (i, s), or we may allow *fractional routing* $f_{uv}^{(i,s)} \in [0, 1]$, where $f_{uv}^{(i,s)}$ is the fraction of the flow serving request (i, s) that traverses link (u, v).

D. Problem: Optimal Joint Caching and Routing

We now formally define the joint caching and routing problem we want to solve in the form of an optimization:

$$\min_{\boldsymbol{f},\boldsymbol{x},\boldsymbol{r}} \sum_{(i,s)\in R} \lambda_{(i,s)} \sum_{(u,v)\in E} w_{uv} f_{uv}^{(i,s)} \tag{1a}$$

$$\text{s.t.} \sum_{(i,s)\in R} \lambda_{(i,s)} f_{uv}^{(i,s)} \le c_{uv}, \quad \forall (u,v) \in E, \tag{1b}$$

$$\sum_{w:(u,w)\in E} f_{uw}^{(i,s)} - \sum_{w:(w,u)\in E} f_{wu}^{(i,s)} = r_u^{(i,s)} - \mathbb{1}_{u=s}, \quad \forall (i,s) \in R, u \in V, \tag{1c}$$

$$\sum_{u\in V} r_u^{(i,s)} = 1, \qquad \forall (i,s) \in R, \tag{1d}$$

$$r_v^{(i,s)} \le x_{vi}, \qquad \forall (i,s) \in R, v \in V, \tag{1e}$$

$$\sum_{i\in C} x_{vi} \le c_v, \qquad \forall v \in V, \tag{1f}$$

$$x_{vi} \in \begin{cases} \{0, 1\} & \text{if integral caching,} \\ [0, 1] & \text{if fractional caching,} \end{cases} \quad \forall v \in V, i \in C, \tag{1g}$$

$$f_{uv}^{(i,s)}, \, r_v^{(i,s)} \in \begin{cases} \{0, 1\} & \text{if integral routing,} \\ [0, 1] & \text{if fractional routing,} \end{cases} \quad \forall (i,s) \in R, (u,v) \in E, v \in V. \tag{1h}$$

The decision variables are $\boldsymbol{f} := (f_{uv}^{(i,s)})_{(i,s)\in R,(u,v)\in E}$ (routing), $\boldsymbol{x} := (x_{vi})_{v\in V, i\in C}$ (caching), and $\boldsymbol{r} := (r_v^{(i,s)})_{(i,s)\in R, v\in V}$ (source selection).

The objective (1a) is to minimize the total routing cost (per unit time). Constraints (1b) and (1c) are the link capacity and the flow conservation constraints as in the multicommodity flow problem. In our context, each *commodity* (i, s) represents the responses to requests of type (i, s), and $r_u^{(i,s)} - \mathbb{1}_{u=s}$ is the fraction of commodity (i, s) emitted from node u ($\mathbb{1}_{\cdot}$ denotes the indicator function). Constraint (1d) ensures that each request is served by sufficient sources, and constraint (1e) ensures that each selected source stores (a sufficient fraction of) the requested content. Constraint (1f) models the cache capacity constraint at each node. Constraints (1g) and (1h) specify the allowable caching/routing decisions. Note that integral routing implies integral source selection, as modeled by (1c). While the reverse is not strictly true, we only consider cases that routing and source selection are simultaneously integral/fractional, as they can often be combined into a pure routing problem in an auxiliary graph as shown later.

Based on the choices in constraints (1g) and (1h), (1) models the joint optimization of caching and routing in three cases:

1) *fractional caching and fractional routing (FC-FR)*,
2) *integral caching and fractional routing (IC-FR)*, and
3) *integral caching and integral routing (IC-IR)*.

In theory, there is a fourth case, *fractional caching and integral routing (FC-IR)*. However, under integral routing, the source selection must also be integral, which means that there is no value for caching partial content items. Therefore, there must be an optimal solution for FC-IR that is feasible (and optimal) for IC-IR, and thus it suffices to consider the above three cases.

Clearly, IC-IR is the most constrained case with the worst routing cost (under the optimal solution) among the three cases, but it also has the least requirement on implementation, by storing uncoded content and performing single-path routing. Meanwhile, FC-FR is the least constrained case with the best routing cost, but its solution is the most complicated to implement, requiring content encoding/decoding and support of multi-path routing. It is thus of interest to investigate all three cases to understand the tradeoff among computational complexity, routing cost, and implementation requirements.

III. Complexity Analysis

The optimization (1) is a linear programming (LP), integer linear programming (ILP), or mixed integer linear programming

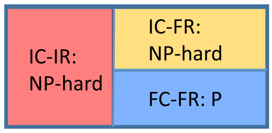

Fig. 1. Complexity analysis for the joint caching and routing problem (1).

(MILP) problem, depending on the choices in constraints (1g) and (1h)). We start by analyzing the complexity in solving (1) optimally in various cases.

Complexity of IC-IR: It is easy to see that the optimization (1) incorporates the multicommodity flow problem as a subproblem, as even if the optimal caching and source selection decision $(\boldsymbol{x}, \boldsymbol{r})$ is given, the remaining problem is still a multicommodity flow problem. Specifically, each commodity corresponds to a type of request (i, s), with a source v such that $r_v^{(i,s)} = 1$, a destination s, and a demand $\lambda_{(i,s)}$, and we need to find a single path for each commodity such that all the demands can be satisfied at the minimum cost within the link capacities, which is the *minimum-cost unsplittable flow problem* that is NP-hard [25], [26]. Therefore, (1) under IC-IR is NP-hard.

Complexity of IC-FR: It has been shown that integral caching is already NP-hard. Specifically, in the special case of $c_{uv} = \infty$ $(\forall(u, v) \in E)$, (1) reduces to the MinCost-SR problem in [3], which is known to be NP-hard due to a reduction from the 2-Disjoint Set Cover Problem. Therefore, (1) under IC-FR remains NP-hard.

Complexity of FC-FR: In this case, (1) becomes an LP, which is polynomial-time solvable by existing LP algorithms (e.g., Karmarkar's algorithm [27]).

Summary: Fig. 1 summarizes the complexity of the joint caching and routing problem (1) in all the cases. Except for the case of FC-FR, the problem is always NP-hard, which motivates our search for efficient approximation algorithms.

IV. Algorithm Design

We now study efficient algorithms for solving (1) approximately. Since FC-FR is polynomial-time solvable, we will focus on the cases of integral caching and/or integral routing.

A. Approximation Under Unlimited Link Capacities

If the network is lightly loaded, i.e., each link has sufficient capacity to serve all the demands $(\sum_{(i,s)\in R} \lambda_{(i,s)} \leq \min_{(u,v)\in E} c_{uv})$, then the routing decision becomes easy. Specifically, given the content placement in caches, we should always serve each request (i, s) from the nearest (i.e., least-cost) node storing the requested content. If the nearest node only stores a fraction of (the coded sub-chunks of) the content, then we should also retrieve from the second nearest node storing the content and so on, until the request is fully satisfied. This is a generalization of the *route-to-nearest-replica (RNR)* strategy in ICN [3], and will be referred to as RNR in the sequel. The focus is therefore on finding a good content placement. As explained in Section II-D, if either routing or caching is limited by integer constraints, then the optimal caching solution is integral. We thus consider the problem of finding the optimal integral content placement under RNR.

This problem has been considered in [3], which developed a pseudo polynomial-time algorithm that achieves a constant-factor approximation to the optimal solution. However, the algorithm's complexity is polynomial in the total number of possible routing paths, which is generally exponential in the network size.[1] Below, we will develop a truly polynomial-time algorithm that achieves the same constant-factor approximation.

1) Equivalent Formulation: The key in circumventing the high complexity for considering all possible paths is to recognize that only the least-cost paths between nodes *may* be used under the optimal solution. Let $w_{v\to s}$ denote the minimum routing cost from node v to node s, and $w_{\max}$ be an upper bound on the maximum $w_{v\to s}$ over all $v, s \in V$. It is well-known that $(w_{v\to s})_{v,s\in V}$ and the associated paths can be computed in polynomial time by shortest path algorithms (e.g., Dijkstra). Given a content placement $\boldsymbol{x}$ and a source selection $\boldsymbol{r}$, we define a proxy objective function:

$$C_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) := \sum_{(i,s)\in R} \lambda_{(i,s)} \sum_{v\in V} r_v^{(i,s)} (x_{vi} w_{v\to s} + (1-x_{vi}) w_{\max}),$$

based on which we formulate the following optimization:

$$\min_{\boldsymbol{x},\boldsymbol{r}} \quad C_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \tag{2a}$$

$$\text{s.t.} \quad \sum_{v\in V} r_v^{(i,s)} = 1, \qquad \forall(i, s) \in R, \tag{2b}$$

$$\sum_{i\in C} x_{vi} \leq c_v, \qquad \forall v \in V, \tag{2c}$$

$$x_{vi},\ r_v^{(i,s)} \in \{0,\ 1\}, \qquad \forall v \in V, (i, s) \in R. \tag{2d}$$

As requesting content item i from a node v not storing it (i.e., $x_{vi} = 0$) will incur a large cost $w_{\max}$, the optimal solution to $\boldsymbol{r}$ must only select the source for each request among the nodes storing the requested content, and must select the source with the least routing cost to the requester (i.e., RNR). Thus, the optimal solution to (2) will minimize the cost in serving all the requests (due to (2b)) subject to cache capacity constraints (2c) and integer constraints (2d), which makes (2) a special case of (1) under IC-IR when $c_{uv} = \infty$ $(\forall(u, v) \in E)$.

Next, we convert the problem into an equivalent maximization problem. Define a complementary objective function

$$F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) := C_{\text{RNR}}^{(0)} - C_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \tag{3}$$

that represents the "cost saving" due to content placement $\boldsymbol{x}$ and source selection $\boldsymbol{r}$, where $C_{\text{RNR}}^{(0)} := |V| w_{\max} \sum_{(i,s)\in R} \lambda_{(i,s)}$ is a constant. It is easy to see that minimizing $C_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ is equivalent to maximizing $F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$. As will be shown below, the maximization problem accepts a constant-factor approximation.

2) Submodularity of Objective: As an explanation of why the maximization of $F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ is easier to solve, we will show that

[1]The issue was addressed in [3] by heuristically selecting a polynomial number of candidate paths (e.g., k shortest paths to the server), but the loss of optimality due to ignoring the other possible paths was not addressed.

F_{RNR} can be written as a *monotone submodular function* [28] of content placement. To this end, we rewrite F_{RNR} as a set function: for any $X \subseteq V \times C$,

$$\widetilde{F}_{\text{RNR}}(X) := \max_{\boldsymbol{r} \text{ s.t. } (2b),(2d)} F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}), \tag{4}$$

where $x_{vi} = 1$ if $(v, i) \in X$ and $x_{vi} = 0$ otherwise $(\forall v \in V,\ i \in C)$. This function has the following properties .

Lemma IV.1: The function $\widetilde{F}_{\text{RNR}}(X)$ is monotone increasing and submodular in X.

Under the set function representation, the maximization of $F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ subject to (2b)–(2d) is equivalent to

$$\max_{X \subseteq V \times C} \quad \widetilde{F}_{\text{RNR}}(X) \tag{5a}$$

$$\text{s.t.} \quad |\{i \in C : (v, i) \in X\}| \le c_v, \qquad \forall v \in V, \tag{5b}$$

where the optimization of $\boldsymbol{r}$ has been incorporated into $\widetilde{F}_{\text{RNR}}(X)$. It is easy to see that the cache capacity constraint (5b) is a matroid constraint [28].

There are generic polynomial-time approximation algorithms for maximizing a monotone submodular function under matroid constraints. Specifically, the greedy algorithm of iteratively expanding the set X by adding an element (v, i) that maximally increases the objective value achieves a $1/2$-approximation [29]. A better approximation ratio of $(1 - 1/e)$ is achieved by the randomized algorithm in [28], which cannot be further improved under the value oracle model [30]. However, this randomized algorithm has a complexity of $O(n^8)$ where n is the rank of the matroid [28]. In our case, $n = \sum_{v \in V} c_v$ (total cache capacity), which can be large, making this generic algorithm computationally expensive.

Remark: Contrary to the claim in [3] that jointly optimizing caching and routing decisions *is not* a submodular maximization problem subject to matroid constraints, we have proved that under proper formulation (i.e., (5)), the problem *is* a submodular maximization problem under matroid constraints. Note that our problem is equivalent to the (offline) joint caching and routing problem in [3] in that the optimal content placement according to (5) together with RNR solves the joint caching and routing problem in [3] optimally.

3) Approximation Algorithm: Below, we will develop a tailor-made algorithm for maximizing $F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ subject to (2b)–(2d) that achieves the same approximation ratio as the generic algorithm in [28] at a much lower complexity. The idea is to use pipage rounding [31]. Generally, to apply pipage rounding, we need to answer two questions: (i) how to efficiently compute a fractional solution that achieves a guaranteed approximation to the optimal, and (ii) how to round the fractional solution to an integral solution without degrading the objective value. We now answer these questions in detail.

Auxiliary LP: We compute a fractional approximate solution by replacing the non-concave objective function $F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ by a concave function that is easier to maximize.

Lemma IV.2: For any $\boldsymbol{x}$ and $\boldsymbol{r}$ satisfying $x_{vi}, r_v^{(i,s)} \in [0,\ 1]$ $(\forall v \in V, i \in C, (i, s) \in R)$, $(1 - 1/e) L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \le F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \le L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$, where

$$L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) := \sum_{(i,s) \in R} \lambda_{(i,s)} \sum_{v \in V} w_{\max} \cdot \min\left(1, 1 - r_v^{(i,s)} + \frac{x_{vi}(w_{\max} - w_{v \to s})}{w_{\max}}\right). \tag{6}$$

The new objective function $L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ is concave and piecewise linear. By introducing an auxiliary variable $z_v^{(i,s)}$, we can formulate the maximization of $L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ subject to (2b), (2c), and the relaxation of (2d) as an LP:

$$\max_{\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z}} \quad \sum_{(i,s) \in R} \lambda_{(i,s)} \sum_{v \in V} w_{\max} z_v^{(i,s)} \tag{7a}$$

$$\text{s.t.} \quad z_v^{(i,s)} \le 1, \qquad \forall (i, s) \in R, v \in V, \tag{7b}$$

$$z_v^{(i,s)} \le 1 - r_v^{(i,s)} + \frac{x_{vi}(w_{\max} - w_{v \to s})}{w_{\max}}, \quad \forall (i, s) \in R, v \in V, \tag{7c}$$

$$(2b), (2c), \tag{7d}$$

$$x_{vi},\ r_v^{(i,s)} \in [0,\ 1], \qquad \forall v \in V, (i, s) \in R. \tag{7e}$$

Due to the maximization and the constraints (7b)–(7c), $z_v^{(i,s)}$ must equal $\min\left(1, 1 - r_v^{(i,s)} + \frac{x_{vi}(w_{\max} - w_{v \to s})}{w_{\max}}\right)$ under the optimal solution, making the objective function (7a) equal to $L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$. As an LP, (7) is polynomial-time solvable. Solving (7) gives a fractional solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$ that maximizes L_{RNR} and hence achieves a $(1 - 1/e)$-approximation in maximizing F_{RNR} by Lemma IV.2.

Pipage rounding: Given the fractional solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$, we round it to an integral solution while preserving F_{RNR} by repeating the following step: As long as $\exists \tilde{x}_{vi}, \tilde{x}_{vj} \in (0, 1)$, we will update their values by

$$x_{vi} = \min(1, \tilde{x}_{vi} + \tilde{x}_{vj}), \quad x_{vj} = \tilde{x}_{vi} + \tilde{x}_{vj} - x_{vi} \tag{8}$$

if $\sum_{s:(i,s) \in R} \lambda_{(i,s)} \tilde{r}_v^{(i,s)} (w_{\max} - w_{v \to s}) \ge \sum_{s:(j,s) \in R} \lambda_{(j,s)} \tilde{r}_v^{(j,s)} (w_{\max} - w_{v \to s})$, and

$$x_{vj} = \min(1, \tilde{x}_{vi} + \tilde{x}_{vj}), \quad x_{vi} = \tilde{x}_{vi} + \tilde{x}_{vj} - x_{vj} \tag{9}$$

otherwise. This rounding scheme has the following property.

Lemma IV.3: Given a possibly fractional solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$ satisfying (2b), (2c), and (7e), repeatedly applying (8)–(9) will construct an integral solution $\boldsymbol{x}$ in $O(|V|^2|C|)$ time that satisfies (2c), (2d), and $F_{\text{RNR}}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \ge F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$.

Proposed algorithm: The entire algorithm is summarized in Algorithm 1, where line 1 prepares parameters for the auxiliary LP (7), line 2 solves the LP for a fractional solution, line 3 applies pipage rounding, and line 4 computes the corresponding source selection by serving each request from the nearest node storing the requested content, i.e., RNR. The performance of Algorithm 1 is guaranteed as follows.

Theorem IV.4: Algorithm 1 has a complexity of $O(|V||E| + |R|^{2.5}|V|^{2.5})$ and produces a feasible solution $(\boldsymbol{x}, \boldsymbol{r})$ such that

Algorithm 1: Integral Caching and Source Selection under RNR.

input: Network topology $G=(V,E)$, link costs $(w_{uv})_{(u,v)\in E}$, cache capacities $(c_v)_{v\in V}$, content catalog C, request rates $(\lambda_{(i,s)})_{(i,s)\in R}$
output: Integral caching decision $\boldsymbol{x}$ and source selection $\boldsymbol{r}$
compute pairwise least costs $(w_{v\to s})_{v,s\in V}$ and the maximum pairwise cost $w_{\max}$;
solve the LP (7) for a fractional solution $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{r}})$;
round $\tilde{\boldsymbol{x}}$ to an integral solution $\boldsymbol{x}$ by (8)–(9);
compute an integral $\boldsymbol{r}$ based on $\boldsymbol{x}$ using RNR;

$F_{\text{RNR}}(\boldsymbol{x},\boldsymbol{r})\geq(1-1/e)F_{\text{RNR}}(\boldsymbol{x}^*,\boldsymbol{r}^*)$, where $(\boldsymbol{x}^*,\boldsymbol{r}^*)$ is the optimal solution to (2).

Remark: Although a similar approach was taken in [3], the solution therein enumerates candidate paths and thus can only consider a subset of all possible paths to achieve a polynomial complexity. In contrast, our algorithm effectively optimizes over all possible paths (while maintaining a polynomial complexity), and can thus significantly outperform [3] (see Fig. 5).

4) A Special Case: Consider now the special case where a subset U of nodes are pure requesters (not caching anything), and another subset H of nodes are pure caches (not requesting anything). In this case, it suffices to model the network as a bipartite graph $\tilde{G}=(H,U,\tilde{E})$, where the logical link $(h,u)\in\tilde{E}$ represents the least-cost path from h to u, with cost $w_{h\to u}$. We can ignore how these least-cost paths traverse the underlying network as the links have unlimited capacities.

This reduces our problem to the *FemtoCaching* problem in wireless networks [32], where nodes generating requests are one-hop away from caches deployed at the network edge. In the further special case where except for one node $h_0\in H$ (that denotes the origin server), all the cache $\to$ requester paths have equal cost w_1 with $w_1<\min_{u\in U} w_{h_0\to u}$, [32] developed a pipage-rounding-based algorithm with an approximation ratio of[2] $(1-1/e)$, and a complexity similar to solving an LP with $(|U|+|H|)|C|$ variables and constraints. In this sense, we have shown that the same performance guarantee can be achieved for a general cache network with arbitrary routing costs, as long as the links are uncapacitated. The cost we pay for such generality is complexity: instead of solving an LP with $(|U|+|H|)|C|=O(|V||C|)$ variables and constraints as in [32], Algorithm 1 needs to solve an LP with $O(|V||R|)=O(|V|^2|C|)$ variables and constraints.

B. Bicriteria Approximation Under Binary Cache Capacities

We see from Section IV-A that the routing decision becomes trivial (i.e., RNR) when the link capacity constraints are removed. We now consider another special case where the caching decision becomes trivial. Specifically, suppose that $c_v=|C|$ for $v\in V_s\subset V$, and $c_v=0$ for the rest. Then each node $v\in V_s$ will store the entire catalog and each $v\in V\setminus V_s$ will store nothing.

[2]The precise approximation ratio is $1-(1-\frac{1}{d})^d$, where $d:=\max_{u\in U}\deg(u)-1$ [32], which converges to $1-1/e$ as d gets large.

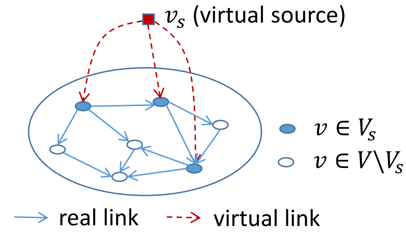

Fig. 2. Auxiliary graph G' that augments G by adding a virtual source v_s connected to all real sources in V_s.

This models scenarios with predetermined, geographically distributed backup servers (i.e., in CDNs).

1) Equivalent Formulation: We will show that in this case, the joint optimization of source selection and routing is equivalent to a single-source routing problem in an auxiliary graph. Consider the auxiliary graph G' that is constructed by adding to G a new node v_s and a new link $(v_s,\ v)$ for every $v\in V_s$, as illustrated in Fig. 2. We will refer to v_s as the *virtual source* and $(v_s,\ v)$ as a *virtual link*. Let $E':=E\cup\{(v_s,v):\ v\in V_s\}$ denote the link set for G'. Assign to each virtual link a zero cost and an unlimited capacity. Then our problem is equivalent to a single-source routing problem in G' as stated below.

Lemma IV.5: Under the content placement $x_{vi}=1$ for all $v\in V_s, i\in C$ and $x_{vi}=0$ otherwise, minimizing the cost in serving all the requests in G by optimizing source selection $\boldsymbol{r}$ and routing $\boldsymbol{f}$ is equivalent to minimizing the cost in serving the same requests in G' by optimizing the routing $\boldsymbol{f}'$ from v_s to content requesters.

2) Bicriteria Approximation Algorithms: Under fractional routing (which implies fractional source selection) in G, the corresponding single-source routing problem in G' is easily solvable by an LP (e.g., the LP relaxation of (10)). Hence, we focus on the case of integral routing (and integral source selection), in which case the corresponding single-source routing problem in G' is:

$$\min_{\boldsymbol{f}'} \sum_{(i,s)\in R}\lambda_{(i,s)}\sum_{(u,v)\in E} w_{uv}{f'}^{(i,s)}_{uv} \tag{10a}$$

$$\text{s.t.} \sum_{(i,s)\in R}\lambda_{(i,s)}{f'}^{(i,s)}_{uv}\leq c_{uv},\quad \forall (u,v)\in E, \tag{10b}$$

$$\sum_{w:(u,w)\in E}{f'}^{(i,s)}_{uw}-\sum_{w:(w,u)\in E}{f'}^{(i,s)}_{wu}=\mathbb{1}_{u\in V_s}{f'}^{(i,s)}_{v_s u}-\mathbb{1}_{u=s},\quad \forall (i,s)\in R, u\in V, \tag{10c}$$

$$\sum_{v\in V_s}{f'}^{(i,s)}_{v_s v}=1,\qquad \forall (i,s)\in R, \tag{10d}$$

$${f'}^{(i,s)}_{uv}\in\{0,\ 1\},\quad \forall (i,s)\in R, (u,v)\in E', \tag{10e}$$

known as the *minimum-cost single-source unsplittable flow problem (MSUFP)* [33]. Under the conversion of $f^{(i,s)}_{uv}={f'}^{(i,s)}_{uv}$ for all $(i,s)\in R$ and $(u,v)\in E$, and $r^{(i,s)}_v={f'}^{(i,s)}_{v_s v}$ for all $(i,s)\in R$ and $v\in V_s$, it is easy to see that (10) is a special case of (1) under integral routing, when $c_v = |C|$ for all $v \in V_s$ and $c_v = 0$ for all $v \in V \setminus V_s$.

For ease of presentation, we define MSUFP using simpler notations as follows.

Definition 1: Given a graph $G = (V, E)$ with capacity c_e and cost w_e associated with each link $e \in E$, and commodities $i = 1, \ldots, n$, each with source s, destination d_i, and demand λ_i, MSUFP aims at finding an unsplittable flow satisfying all the demands within the link capacities at the minimum cost, i.e., a set of paths $\{p_i\}_{i=1}^n$ such that routing commodity i on p_i satisfies the demands while satisfying $\sum_{i:e\in p_i} \lambda_i \le c_e$ ($\forall e \in E$), and achieves the minimum cost measured by $\sum_{i=1}^n \lambda_i \sum_{e\in p_i} w_e$ among all the feasible solutions.

MSUFP is NP-hard [25]. Notable efforts have been devoted to designing approximation algorithms, which generally start from an initial splittable flow $\tilde{\boldsymbol{f}}$ (i.e., fractional routing) and then round it into an unsplittable flow $\boldsymbol{f}$. It has been shown in [34] that in the worst case, rounding a splittable flow that satisfies the link capacity constraints into an unsplittable flow will violate the capacity of some link by an amount arbitrarily close to the maximum demand. Therefore, existing algorithms focus on obtaining *bicriteria approximation* defined as follows.

Definition 2: A solution $\boldsymbol{f}$ to MSUFP is a bicriteria (α, β)-approximation if: (i) the total load on each link imposed by $\boldsymbol{f}$ is within α times its capacity, and (ii) the total cost incurred by $\boldsymbol{f}$ is within β times the optimal cost.

Despite extensive studies, existing results on MSUFP are far from satisfactory. Under arbitrary demand, the best known bicriteria approximation ratio is $(3 + 2\sqrt{2},\ 1)$ [33]. If the maximum demand is within the minimum link capacity, the best known ratio is $(3,\ 1)$ [33]; under the same condition, [35] proved that for any $\epsilon > 0$, there is no bicriteria $(2 - \epsilon,\ 1)$-approximation algorithm for MSUFP unless $P = NP$. These results imply that if we use the algorithms therein to solve (10), some link may carry a load that is three times its capacity, which will cause significant congestion.

To address this issue, we will show a better approximation algorithm in the scenario where the maximum demand is much smaller than the minimum link capacity, i.e., $\max_{i\in\{1,\ldots,n\}} \lambda_i =: \lambda_{\max} \ll c_{\min} := \min_{e\in E} c_e$. This scenario models cases where the network serves a large number of users with a large catalog, but each user only has a small demand for each item in the catalog. In this case, we will provide a polynomial-time algorithm that achieves no more than the optimal cost, while causing no more than ϵ congestion on each link for an arbitrarily small $\epsilon > 0$. We will present the main results here and defer further explanations to Appendix B of the supplementary file, available online.

Subroutine: The basis of our solution is an algorithm developed in [33], which converts a splittable flow to an unsplittable flow with the following properties.

Lemma IV.6 ([33]): Given MSUFP with demands $\lambda_i = \lambda_{\min} 2^{q_i}$ $(i = 1, \ldots, n)$ for $q_i \in \mathbb{N}$ (natural numbers including zero) and $0 = q_1 \le q_2 \le \cdots \le q_n$, and a splittable flow $\boldsymbol{f}$ satisfying all the demands, [33, Algorithm 2] outputs an unsplittable flow that routes each commodity i on a single path p_i in $O(n|V| + |E|q_n + |V||E|)$ time, such that (i) $\sum_{i=1}^n \lambda_i \sum_{e\in p_i} w_e$ is no more than the cost of $\boldsymbol{f}$, and (ii) $\forall e \in E$, if $i_e := \arg\max_{i:e\in p_i} \lambda_i$, then $\sum_{i\ne i_e: e\in p_i} \lambda_i < f(e)$, the total flow on link e under $\boldsymbol{f}$.

Algorithm 2: Bicriteria Approximation for MSUFP.

input : Network topology $G = (V, E)$, link costs $(w_e)_{e\in E}$, link capacities $(c_e)_{e\in E}$, commodities $i \in \{1, \ldots, n\}$ with source s, destination d_i, and demand λ_i, and design parameter $K \in \mathbb{N}$

output: Paths $(p_i)_{i=1}^n$, each for routing one commodity

1 compute a feasible splittable flow $\boldsymbol{f} := ((f_e^{(i)})_{e\in E})_{i=1}^n$ that satisfies all the demands $(\lambda_i)_{i=1}^n$ at the minimum cost;

2 convert the link-level flow $\boldsymbol{f}$ to a path-level flow $((f_p^{(i)})_{p\in P_i})_{i=1}^n$ by the *Edmonds-Karp algorithm* as in [36], where P_i is the set of paths carrying commodity i and $f_p^{(i)}$ the amount of commodity i on path $p \in P_i$;

3 **foreach** $i = 1, \ldots, n$ **do**

4   reduce $(f_p^{(i)})_{p\in P_i}$ in descending order of $\sum_{e\in p} w_e$ until the reduced flow satisfies $\sum_{p\in P_i} \bar{f}_p^{(i)} = \bar{\lambda}_i$ as in (11);

5 split the reduced flow $\bar{\boldsymbol{f}}$ into $\bar{\boldsymbol{f}}_j := ((\bar{f}_p^{(i)})_{p\in P_i})_{i\in S_j}$ $(j = 0, \ldots, K-1)$ for S_j defined in (12);

6 **foreach** $j = 0, \ldots, K-1$ **do**

7   convert $\bar{\boldsymbol{f}}_j$ into an unsplittable flow by [33, Algorithm 2], specified by paths $(p_i)_{i\in S_j}$;

8 return paths $(p_i)_{i=1}^n$, with p_i serving demand λ_i;

Proposed algorithm: Using [33, Algorithm 2] as a subroutine, we develop a three-step algorithm for arbitrary demands $(\lambda_i)_{i=1}^n$ in Algorithm 2. First, we compute by LP an optimal splittable flow $\boldsymbol{f}$ that satisfies these demands within the link capacities with the minimum cost (line 1). Second, using a given parameter $K \in \mathbb{N}$, we round each demand λ_i to[3]

$$\bar{\lambda}_i := \begin{cases} \lambda_{\max} 2^{\lfloor K \log(\lambda_i/\lambda_{\max})\rfloor/K} & \text{if } \lambda_i < \lambda_{\max}, \\ \lambda_{\max}/2^{1/K} & \text{if } \lambda_i = \lambda_{\max}. \end{cases} \tag{11}$$

The rounded demand satisfies $\lambda_i 2^{-1/K} \le \bar{\lambda}_i \le \lambda_i$. We then reduce the flow $\boldsymbol{f}$ along the most expensive paths to a new splittable flow $\bar{\boldsymbol{f}}$ satisfying demands $(\bar{\lambda}_i)_{i=1}^n$ (lines 2–4). Third, we partition the commodities $\{1, \ldots, n\}$ into K subsets:

$$S_j :\, = \left\{i \in \{1, \ldots, n\} :\, -\frac{\lfloor K \log(\lambda_i/\lambda_{\max})\rfloor}{K} + \frac{j}{K} \in \mathbb{N}\right\},\quad j = 0, \ldots, K-1. \tag{12}$$

We then split $\bar{\boldsymbol{f}}$ into flows $\bar{\boldsymbol{f}}_j$ $(j = 0, \ldots, K-1)$ such that $\bar{\boldsymbol{f}}_j$ satisfies demands $(\bar{\lambda}_i)_{i\in S_j}$ (line 5), and convert $\bar{\boldsymbol{f}}_j$ into an unsplittable flow by [33, Algorithm 2], which routes each commodity i $(i \in S_j)$ on a path p_i (lines 6–7). The final solution is to route the original demand λ_i on path p_i for each $i = 1, \ldots, n$ (line 8).

The performance of Algorithm 2 is guaranteed as follows.

Theorem IV.7: Given MSUFP with demands $\lambda_{\min} := \lambda_1 \le \cdots \le \lambda_n =: \lambda_{\max}$, Algorithm 2 computes an unsplittable

[3] The log here denotes base-2 logarithm.

flow that serves demand λ_i by path p_i $(i = 1, \ldots, n)$ in $O(n^{2.5}(|V| + |E|)^{2.5} + K|E|(\log(\frac{\lambda_{\max}}{\lambda_{\min}}) + |V|))$ time, such that (i) $\sum_{i=1}^{n} \lambda_i \sum_{e\in p_i} w_e$ is no more than the minimum cost, and (ii) $\sum_{i:e\in p_i} \lambda_i < \frac{2^{1/K}}{2(2^{1/K}-1)}\lambda_{\max} + 2^{1/K} c_e, \forall e \in E$.

Remark: Algorithm 2 extends the solution in [33] (called *variant of Algorithm 3*), which addressed a special case of $K = 2$. When $\lambda_{\max} \ll c_{\min}$, choosing $K = \lceil 1/\log(1+\epsilon)\rceil$ for a small $\epsilon > 0$ implies that the solution given by Algorithm 2 will achieve the optimal cost while incurring a load on each link that is within $(1 + \epsilon)$ times its capacity, i.e., giving a bicriteria $(1 + \epsilon,\ 1)$-approximation. Applying this algorithm to (10) will then give an integral source selection and routing solution to (1) when the catalog is replicated over a given subset of nodes V_s, which incurs no more than the optimal cost and exceeds the capacity of any link by at most a factor of ϵ.

While the case of $\lambda_{\max} \ll c_{\min}$ was considered in [37], which proposed a different algorithm, the performance of that algorithm was not analyzed rigorously. To our knowledge, Algorithm 2 is the *first* algorithm achieving $(1 + \epsilon,\ 1)$-approximation for MSUFP.

C. Heuristics Under General Link/Cache Capacities

Given our experiences in solving the special cases, we propose to solve the general case with arbitrary link/cache capacities by alternatingly optimizing content placement and routing (including source selection).

1) Approximation Algorithm for Content Placement: Consider the problem of integral content placement under a given solution $(\boldsymbol{r},\ \boldsymbol{f})$ to source selection and routing. In the case of integral routing, this problem has been studied in [38], for which a $(1 - 1/e)$-approximation algorithm based on pipage rounding was proposed. Below we show how to achieve the same approximation ratio in the case of fractional routing. Under source selection $\boldsymbol{r}$ and routing $\boldsymbol{f}$, let $P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}$ denote the set of cycle-free paths used to serve requests of type (i, s) and $\lambda_p^{(i,s)}$ $(\forall p \in P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}})$ the rate of requests served by path p. Specifically, given a possibly fractional link-level routing decision $\boldsymbol{f} := (f_{uv}^{(i,s)})_{(i,s)\in R,(u,v)\in E}$, the corresponding path-level routing decision $((f_p^{(i,s)})_{p\in P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}})_{(i,s)\in R}$ can be computed as in [36] in $O(|R||V||E|)$ time ($f_p^{(i,s)}$: the fraction of type-(i, s) requests served by path p), and then $\lambda_p^{(i,s)} = \lambda_{(i,s)} f_p^{(i,s)}$. This conversion also guarantees that $|P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}| \le |E|$ $(\forall (i, s) \in R)$ (see the proof of Theorem IV.7 for explanation). Let $|p|$ denote the number of nodes on path p and p_i $(i = 1, \ldots, |p|)$ the i-th node from the source. Then the cost of serving requests using the paths and rate allocation specified by $(\boldsymbol{r}, \boldsymbol{f})$ and an integral content placement $\boldsymbol{x}$ is

$$C_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) := \sum_{(i,s)\in R} \sum_{p\in P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}} \lambda_p^{(i,s)} \sum_{k=1}^{|p|-1} w_{p_{|p|-k}p_{|p|-k+1}} \cdot \prod_{k'=0}^{k-1}(1 - x_{p_{|p|-k'}i}), \quad (13)$$

because the response to request (i, s) along path p needs to traverse link $(p_{|p|-k},\ p_{|p|-k+1})$ if and only if no node closer to the requester (at node $p_{|p|}$) than node $p_{|p|-k}$ has content i, i.e., $\prod_{k'=0}^{k-1}(1 - x_{p_{|p|-k'}i}) = 1$. This is a generalization of the formulation in [38], which only considers the special case of $|P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}| = 1$ (i.e., integral routing). Note that to be consistent with previous sections, we consider each $p \in P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}$ to be a response path, instead of a request path as in [38].

The solution is based on similar ideas as in Algorithm 1. *First,* the minimization of cost (13) is converted into an equivalent maximization of cost saving, defined as

$$F_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) := C_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{0}) - C_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) = \sum_{(i,s)\in R} \sum_{p\in P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}} \lambda_p^{(i,s)} \sum_{k=1}^{|p|-1} w_{p_{|p|-k}p_{|p|-k+1}} \cdot \left(1 - \prod_{k'=0}^{k-1}(1 - x_{p_{|p|-k'}i})\right). \quad (14)$$

Second, the nonconcave objective function (14) is replaced by a piecewise-linear concave objective function:

$$L_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) := \sum_{(i,s)\in R} \sum_{p\in P^{(i,s)}_{\boldsymbol{r},\boldsymbol{f}}} \lambda_p^{(i,s)} \sum_{k=1}^{|p|-1} w_{p_{|p|-k}p_{|p|-k+1}} \cdot \min\left(1,\ \sum_{k'=0}^{k-1} x_{p_{|p|-k'}i}\right), \quad (15)$$

which can be shown to satisfy $(1 - 1/e)L_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) \le F_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) \le L_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x})$ by applying the Goemans-Williamson inequality [38], [39] as in Lemma IV.2. Using auxiliary variables to represent $\min(1,\ \sum_{k'=0}^{k-1} x_{p_{|p|-k'}i})$ as in (7), the maximization of (15) under cache capacity constraints and $x_{vi} \in [0, 1]$ $(\forall v \in V, i \in C)$ can be written as an LP and solved efficiently. *Finally,* if the solution $\tilde{\boldsymbol{x}}$ is fractional, then a pipage rounding scheme similar to (8)–(9) can be used to round it into an integral solution $\boldsymbol{x}$ such that $F_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) \ge F_{\boldsymbol{r},\boldsymbol{f}}(\tilde{\boldsymbol{x}})$.

Together, these steps produce an integral content placement $\boldsymbol{x}$ that achieves $(1 - 1/e)$-approximation in terms of maximizing $F_{\boldsymbol{r},\boldsymbol{f}}$. That is, compared to the content placement $\boldsymbol{x}^*_{\boldsymbol{r},\boldsymbol{f}}$ that maximizes (14), $\boldsymbol{x}$ satisfies $F_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}) \ge (1 - 1/e)F_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x}^*_{\boldsymbol{r},\boldsymbol{f}})$.

2) Algorithms for Source Selection and Routing: Given an integral content placement $\boldsymbol{x}$, we can reduce the joint optimization of source selection $\boldsymbol{r}$ and routing $\boldsymbol{f}$ to a pure routing problem by a construction similar to Lemma IV.5. Specifically, let $V_i^{\boldsymbol{x}} := \{v \in V :\ x_{vi} = 1\}$ be the set of nodes storing content i under placement $\boldsymbol{x}$ $(\forall i \in C)$. We can construct an auxiliary graph $G^{\boldsymbol{x}} := (V \cup \{v_i\}_{i\in C},\ E \cup \bigcup_{i\in C}\{(v_i, v) :\ v \in V_i^{\boldsymbol{x}}\})$, where v_i is the virtual source for content i that is connected to each of the real sources for content i via a virtual link that has a zero cost and an unlimited capacity. Then by the same arguments as in Lemma IV.5, we see that minimizing the total routing cost in G by a joint optimization of source selection and routing under content placement $\boldsymbol{x}$ is equivalent to minimizing the total routing cost in $G^{\boldsymbol{x}}$ by optimizing the routing from the virtual source v_i of each content to its requesters.

The resulting routing problem in $G^{\boldsymbol{x}}$ is known as the *minimum-cost multiple-source splittable/unsplittable flow problem (MMSFP/MMUFP)* depending on whether fractional routing is allowed. Under fractional routing, the corresponding problem (MMSFP) can be solved via LP. If routing must be integral, then the corresponding problem (MMUFP) is NP-hard [26]. A number of heuristics for MMUFP, e.g., greedy and LP relaxation with randomized rounding, have been proposed [26]. The optimal solution can also be computed by the branch-and-price-and-cut algorithm [40], although with an exponential complexity. *Remark:* In contrast to the bicriteria approximations for MSUFP (see Section IV-B2), approximating MMUFP is much harder. This is because in the single-source case, if all demands and link capacities are integer multiples of α for any $\alpha > 0$, then we can compute in polynomial time a minimum-cost flow whose value on each link is an integer multiple of α [33], but in the multiple-source case, computing such a flow is NP-hard [41]. To our knowledge, how to solve MMUFP efficiently with approximation guarantee remains an open problem.

3) Overall Algorithm: Based on the solutions for the subproblems in Sections IV-C1–IV-C2, we propose an algorithm that alternatingly optimizes $\boldsymbol{x}$ and $(\boldsymbol{r}, \boldsymbol{f})$ as follows. Starting from an arbitrary feasible solution $(\boldsymbol{x}^{(0)}, \boldsymbol{r}^{(0)}, \boldsymbol{f}^{(0)})$, repeat the following steps for $t = 1, 2, \ldots$ until there is no more improvement in cost or congestion:

1) compute $\boldsymbol{x}^{(t)}$ by maximizing $F_{\boldsymbol{r}^{(t-1)}, \boldsymbol{f}^{(t-1)}}(\boldsymbol{x})$ subject to cache capacity constraints;
2) compute $(\boldsymbol{r}^{(t)}, \boldsymbol{f}^{(t)})$ by solving MMSFP (under fractional routing) or MMUFP (under integral routing) in $G^{\boldsymbol{x}^{(t)}}$.

After each iteration, we only retain the new solution if it has a lower cost than the solution from the previous iteration.

4) Limitation: The above approach effectively treats the joint caching and routing problem (1) as a two-player cooperative game: one player optimizes the content placement $\boldsymbol{x}$ and the other player optimizes the source selection and routing $(\boldsymbol{r}, \boldsymbol{f})$. The alternating optimization algorithm is designed to find a Nash Equilibrium (NE), where neither player can improve the performance by unilaterally changing its decision. However, this game can have many NEs, some of which can be arbitrarily worse than the optimal solution, as shown below.

Proposition IV.8: The algorithm in Section IV-C3 has an unbounded approximation ratio, even if each of the steps (i.e., optimizing $\boldsymbol{x}^{(t)}$ based on $(\boldsymbol{r}^{(t-1)}, \boldsymbol{f}^{(t-1)})$ and optimizing $(\boldsymbol{r}^{(t)}, \boldsymbol{f}^{(t)})$ based on $\boldsymbol{x}^{(t)}$) is solved optimally.

Remark: Proposition IV.8 indicates that sometimes a locally suboptimal caching/routing decision is needed to converge towards the optimal solution for joint caching and routing. It remains open how to make such suboptimal decisions such that the overall solution achieves a guaranteed approximation, which is left to future work. Meanwhile, despite this negative result on the worst-case performance, the alternating optimization algorithm has shown very good performance in comparison with the state of the art and quick convergence in our evaluations based on real topology and request traces (see Fig. 7–11).

V. Extension to Heterogeneous Content Sizes

Although caching equal-sized chunks as we consider so far is a common assumption in the literature (e.g., [3], [12], [21] and references therein), it implies additional processing at application layer to assemble the equal-sized chunks into the requested files, which generally have heterogeneous sizes. A question of interest is thus how to solve the joint caching and routing problem if we directly cache the files.

A. Extending Problem Formulation

To model heterogeneous file sizes, we allow each content item $i \in C$ to have an arbitrary size of b_i bits. Accordingly, we measure the cache size c_v at node $v \in V$ in bits, the capacity c_{uv} of link $(u, v) \in E$ in bits per unit time, and the demand $\lambda_{(i,s)}$ of type $(i, s) \in R$ in bits per unit time. We also interpret the cost w_{uv} of link $(u, v) \in E$ as the cost of moving one bit over the link.

Under the above model, the problem of minimizing the routing cost under link and cache capacity constraints can still be formulated as in (1), except that the cache capacity constraint (1f) is changed into:

$$\sum_{i \in C} x_{vi} b_i \leq c_v, \quad \forall v \in V. \tag{16}$$

The generalized problem is no easier than the original problem that assumes $b_i \equiv 1$ ($\forall i \in C$), and remains an LP in the case of FC-FR. Hence, the complexity analysis in Section III remains applicable.

B. Revisiting Algorithm Design

1) Case of Binary Cache Capacities: In the special case that a subset of nodes can store the entire catalog and the rest store none as assumed in Section IV-B, Algorithm 2 remains applicable with the same performance guarantee.

2) Case of Unlimited Link Capacities: Under unlimited link capacities as assumed in Section IV-A, Algorithm 1 is no longer applicable. The reason is that the rounding scheme in Lemma IV.3 hinges on the ability to swap equal fractions of different items at a node without exceeding the cache capacity, which is generally infeasible if the items have heterogeneous sizes. Nevertheless, using the objective function $\widetilde{F}_{\text{RNR}}(X)$ defined in (4), we can reformulate the problem as

$$\max_{X \subseteq V \times C} \quad \widetilde{F}_{\text{RNR}}(X) \tag{17a}$$

$$\text{s.t.} \quad \sum_{i \in C: (v,i) \in X} b_i \leq c_v, \qquad \forall v \in V. \tag{17b}$$

We know from Lemma IV.1 that $\widetilde{F}_{\text{RNR}}(\cdot)$ is monotone and submodular. Moreover, the solution space of (17) also has a desirable property as shown below.

Definition 3 ([42]): Let A be a universe of elements and $\mathcal{I} \subseteq 2^A$ a collection of subsets of A.

1) The pair $(A, \mathcal{I})$ is called an *independence system* if: (i) $\emptyset \in \mathcal{I}$, and (ii) if $S_1 \in \mathcal{I}$ and $S_2 \subseteq S_1$, then $S_2 \in \mathcal{I}$.
2) Given an independence system $(A, \mathcal{I})$ and a set $S \subseteq A$, a maximal subset of S that is in $\mathcal{I}$ is called a basis of S. The

rank $r(S)$ is the cardinality of the largest basis of S, and the lower rank $\rho(S)$ is the cardinality of the smallest basis of S. The independence system is called a *p-independence system* if $\max_{S\subseteq A} \frac{r(S)}{\rho(S)} \leq p$.

Lemma V.1: For $\mathcal{I} := \{X \subseteq V \times C : X \text{ satisfies (17b)}\}$, $(V \times C, \mathcal{I})$ is a p-independence system for $p = \lceil b_{\max}/b_{\min} \rceil$, where $b_{\max}/b_{\min}$ is the maximum/minimum item size.

Combining Lemmas IV.1 and V.1 yields the following result.

Theorem V.2: Greedy content placement (i.e., iteratively expanding X by adding an element (v, i) that achieves the maximum $\widetilde{F}_{\text{RNR}}(X \cup \{(v, i)\})$) achieves $1/(1 + p)$-approximation for (17), where $p = \lceil b_{\max}/b_{\min} \rceil$.

Remark: In contrast to the $(1 - 1/e)$-approximation in the case of equal-sized items, caching items of arbitrary sizes has a worse approximation ratio. This is essentially the cost of storing arbitrary-sized files instead of equal-sized chunks.

3) General Case: In the general case with arbitrary link/cache capacities, the heterogeneity in item sizes only affects the content placement subproblem. For reasons similar to Section V-B2, the $(1 - 1/e)$-approximation algorithm in [38] no longer applies. However, we can show that the equivalent objective function $\widetilde{F}_{\boldsymbol{r},\boldsymbol{f}}(X)$ is still monotone and submodular.

Lemma V.3: The function $\widetilde{F}_{\boldsymbol{r},\boldsymbol{f}}(X) := F_{\boldsymbol{r},\boldsymbol{f}}(\boldsymbol{x})$ in (14), where $x_{vi} = 1$ if and only if $(v, i) \in X$, is monotone increasing and submodular in X.

This result together with Lemma V.1 implies that when formulated as a cost saving maximization problem $\max \widetilde{F}_{\boldsymbol{r},\boldsymbol{f}}(X)$ s.t. (17b), the content placement subproblem is again a submodular maximization subject to a p-independence constraint, for which the greedy algorithm achieves $1/(1 + p)$-approximation as shown in the proof of Theorem V.2. Given a content placement, the source selection and routing subproblem can still be solved as MMSFP/MMUFP as in Section IV-C2. Thus, we can still apply the alternating optimization algorithm in Section IV-C3.

VI. Performance Evaluation

We evaluate our solutions against benchmarks in the scenario of *edge caching*, where content items are cached at locations within/near users' access networks. Edge caching has been widely used by large content providers like Google [43] and distributors like Akamai [44], and has been shown to achieve most of the benefits of ICN [45].

Simulation setting: To simulate edge caching, we use an Internet Service Provider (ISP) topology called Abovenet from [46] to model the network, where a degree-1 node is designated as (the gateway to) the origin server permanently storing all the items, and a set V_e of low-degree nodes (with degree ≤ 3) are designated as edge nodes, which receive requests from users and host caches. We assume that each cache can store ζ items. The other nodes are internal routers that only forward requests/responses. See Fig. 3 for the topology.

As the origin server is usually much farther away from users than edge caches, we select the cost for the outgoing link of the origin server randomly from $[100, 200]$, and the costs for the other links randomly from [1,20]. These costs can represent any additive metric (see examples in Section II-A); the choice of cost measure is not our focus. In [1], we have conducted extensive synthetic simulations based on requests generated according to the Zipf distribution as in [3]. Here, we will simulate more realistic content demands based on traces. To this end, we collected #views per hour of the top 12 YouTube videos over 100 consecutive hours between 11/14/2021 and 11/18/2021; see Table I for the statistics. Additional 550 hours of #views of the top 300 videos were collected for training a demand predictor. We use the collected data to perform simulations at two different levels:

- *Chunk level (with homogeneous-sized items):* Each video is divided into 100-MB chunks to be stored in the cache nodes, and the application layer will assemble the equal-sized chunks into the video files that are requested.
- *File level (with heterogeneous-sized items):* Each video is treated as a single item with a heterogeneous size, and stored in its entirety in the cache nodes.

We use the views of these popular videos to represent content requests and randomly distribute the requests for each video among the edge nodes. Following the setting in [3] for topology with a similar size as Abovenet, we set $|C| = 10$ and $\zeta = 2$ by default for file-level simulation. Correspondingly, we set $|C| = 54$ and $\zeta = 12$ for chunk-level simulation to represent the same set of videos and the same cache size (on the average). Both parameters will be varied later. We give each link a default capacity of κ, which is set to 0.7% of the total request rate. In our traces, the top-10 videos have a total request rate of

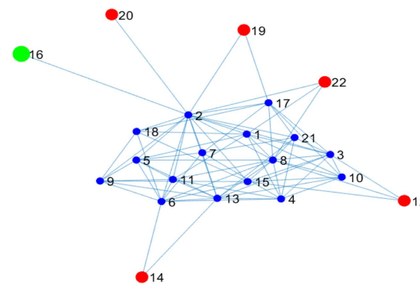

Fig. 3. Abovenet topology; ●: origin server, ●: edge nodes, ●: internal nodes.

TABLE I
Statistics of YouTube Videos in Evaluation

video_id	size (MB)	#100-MB chunks[4]	total #views[5]
dNCWe_6HAM8	450.8789	5	14144021
f5_wn8mexmM	611.7188	7	6046921
3YqPKLZF_WU	746.1914	8	3516996
2dTMIH5gCHg	387.5977	4	2724433
CULF9lXH87w	851.6602	9	1935258
QDYDRA5JPLE	427.1484	5	1606676
LWAI7HkQMyc	158.2031	2	2701699
Zpi7CTDvi1A	709.2773	8	1286994
vH7n1vj-cwQ	155.5664	2	128860
JNCkUEeUFy0	308.4961	4	369157
CaimKeDcudo	337.5	4	613737
gXH7_XaGuPc	680.2734	7	368432

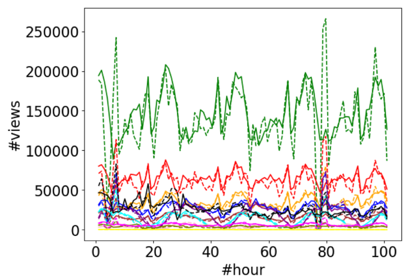

Fig. 4. Numbers of views per hour for top-12 YouTube videos; solid: ground truth, dashed: prediction.

1949666.52 chunks/hour, which equals 381.1902 Gbps. This leads to a default link capacity of 13715.796 chunks/hour, or roughly $\kappa = 3$ Gbps. We will vary the default link capacity later. To ensure feasibility, we augment link capacities along a cycle-free path from the origin server to each edge node so that all the requests can be served by the origin server as a last resort. Our evaluation focuses on the most challenging case of IC-IR. Results are averaged over 100 Monte Carlo runs.

The above setting simulates a real-world scenario, where the network provider adjusts caching and routing decisions on an hourly basis based on the predicted demand. To enable this, we apply Guassian process regression (GPR) from the scikit-learn library [47], with white noise, periodic, and radial-basis function kernels and maximum marginal likelihood fitting, to predict the request rates for the next hour based on a history of at least 550 hours.[6] See the results in Fig. 4. We note that this prediction method is only used to evaluate the proposed caching/routing algorithms under realistic demand prediction; demand prediction is not the focus of this work, and other prediction methods can be applied.

Simulation results: First, in the special case of unlimited link capacities, we compare our proposed algorithm (Algorithm 1 for chunk-level simulation and greedy algorithm for file-level simulation) with the solution in [3] ('k shortest paths') and the content placement algorithm in [38] based on shortest path routing ('shortest path'). We configure the solution in [3] according to its recommendation, by constructing k shortest paths from the server to each edge node as the candidate paths with $k = 10$ by default. We have the following observations based on the results in Fig. 5. In the case of homogeneous item sizes (i.e., chunk-level simulation), (i) our algorithm achieves a substantially lower routing cost than the state-of-the-art solutions in [3], [38], and (ii) the advantage remains as the number of candidate paths for [3] increases. This is because [3], [38] both predetermine the candidate paths based on the server's location, hence not fully utilizing the caches. In the case of heterogeneous item sizes (i.e., file-level simulation), (i) the benchmarks from [3], [38] appear to achieve a lower routing cost than our algorithm, but (ii) their content placement solutions are actually infeasible as shown by the plots of the maximum cache occupancy. This is because the pipage rounding scheme used in [3], [38] swaps equal fractions

[6]To accommodate the training time, we perform prediction for five hours at a time, and then retrain the model using the cumulative history.

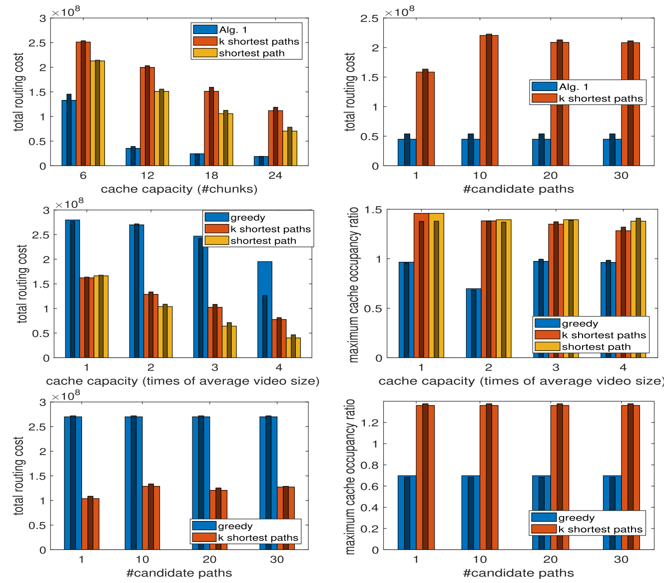

Fig. 5. Case of unlimited link capacities: first row – chunk-level simulation, second row – file-level simulation under varying cache capacity, third row – file-level simulation under varying #candidate paths (light: true demand; dark: predicted demand).

of different items to minimize cost, which can exceed the cache capacity when items have different sizes. The above observations hold regardless of whether the algorithms run on the predicted demand or the true demand (all the performances are evaluated based on the true demand).

Next, we consider the special case of binary cache capacities, where one of the edge nodes (in addition to the server) stores all the items and the rest store none. As our problem reduces to MSUFP in this case, we compare our Algorithm 2, with parameter K tuned to minimize congestion under the default link capacity of 15 Gbps (shown by the varying-K plots in Fig. 6), with the state-of-the-art MSUFP algorithm in [33], which is a special case of our algorithm with $K = 2$. As benchmarks, we also compare with the splittable flow and the solution by [3], which routes each request to the nearest replica ('RNR'). As some of the algorithms may exceed the link capacities, we evaluate congestion in addition to routing cost, measured by the maximum load-to-capacity ratio over all the links. The results in Fig. 6 show that: (i) RNR can cause severe congestion (it exceeds link capacities by up to 51 times; the congestion plots have been truncated for better visibility of other results), (ii) compared to the state-of-the-art algorithm in [33] ('K = 2'), Algorithm 2 with a larger K can substantially reduce the congestion while achieving/beating the minimum routing cost achievable without congestion (which is lower-bounded by the cost for 'splittable flow'), and (iii) compared to serving each video as a whole, serving it in chunks can substantially reduce the routing cost (by 5–6 times) without worsening the congestion. The second observation is because a larger K leads to smaller errors when rounding the demands and thus less congestion when serving the actual demands over paths selected based on the rounded demands. The third observation is because chunking the videos

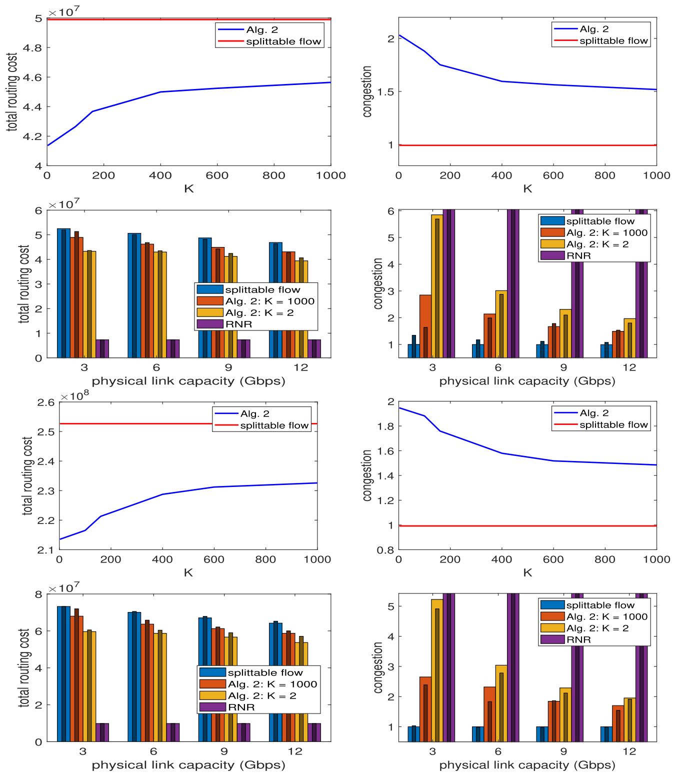

Fig. 6. Case of binary cache capacities: top two rows – chunk-level simulation, bottom two rows – file-level simulation (light: true demand; dark: predicted demand).

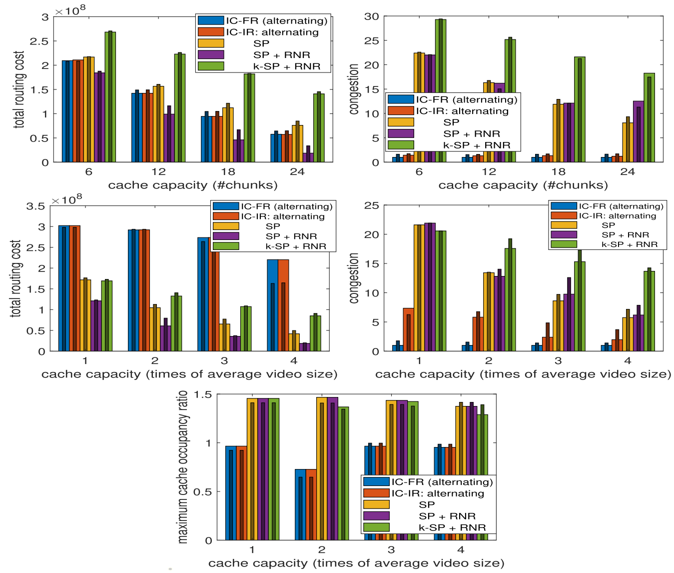

Fig. 7. General case under varying cache capacity: first row – chunk-level simulation, second and third row – file-level simulation (light: true demand; dark: predicted demand).

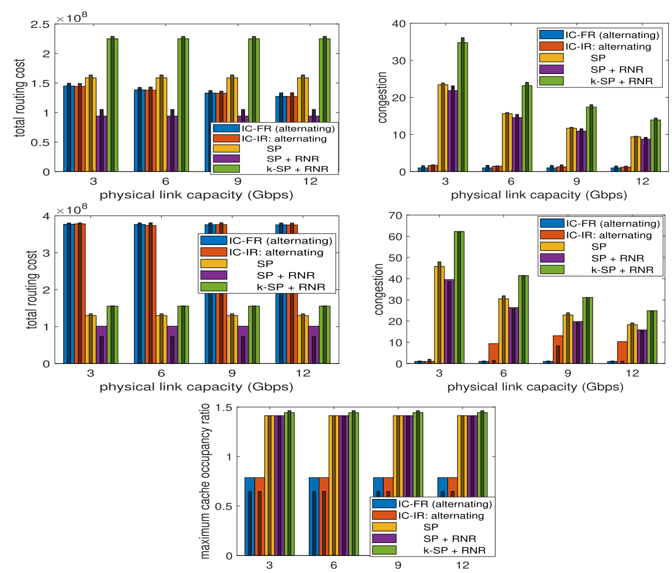

Fig. 8. General case under varying link capacity: first row – chunk-level simulation, second and third row – file-level simulation (light: true demand; dark: predicted demand).

effectively allows each video to be served to each requester via multiple paths (one per chunk), which provides more flexibility than serving each video in its entirety via a single path.

Finally, we consider the general case with limited cache and link capacities. We implement versions of the alternating optimization algorithm proposed in Section IV-C3 ('alternating') that solve content placement by pipage rounding (chunk-level simulation) or greedy algorithm (file-level simulation), and MMUFP by LP relaxation with randomized rounding. We compare them with the solution in [38] based on shortest path routing ('SP'), a variation of [3] with the shortest path as the only candidate path ('SP + RNR'), and the solution in [3] with its recommended way of constructing candidate paths as the $k = 10$ shortest paths ('k-SP + RNR'). The results in Figs. 7 and 8 show the following. For chunk-level caching and routing, (i) our algorithm significantly outperforms [3], [38] in both cost and congestion, and (ii) while 'SP + RNR' achieves a lower cost, it causes severe congestion. For file-level caching and routing, (i) none of the benchmarks is feasible as their content placements substantially exceed the capacity of at least one cache, and (ii) although our algorithm maintains feasibility with respect to cache capacities, it incurs a notably higher level of congestion and a higher routing cost than what is achieved in chunk-level simulation. Our algorithm has also exhibited quick convergence (within 10 iterations) in all the evaluated cases.

In addition to the quality of the solutions, We have also evaluated the computation efficiency of the algorithms as measured by their average execution times under the default parameter setting in the most computationally challenging case of IC-IR. The results, shown in Appendix C of the supplementary file, available online, indicate that the proposed algorithms are sufficiently fast to be applied to adjust caching and routing decisions on a regular basis.

Summary of results: To facilitate comparison, we summarize the qualitative observations from the chunk-level simulation under IC-IR in Table II, which compares our proposed solutions (in bold) with the benchmarks. The summary clearly highlights

TABLE II
SUMMARY OF PERFORMANCE EVALUATION RESULTS

scenario	algorithm	routing cost	congestion
$c_{uv} = \infty$	**Alg. 1**	**lowest**	—
	[3] ('k shortest paths')	highest	—
	[38] ('shortest path')	2nd highest	—
$c_v = 0/\|C\|$	**Alg. 2** ($K = 1000$)	$<$ **optimal**	**low**
	[33] ($K = 2$)	$<$ optimal	moderate
	[3] ('RNR')	$\ll$ optimal	severe
general	**alternating**	$\approx$ **IC-FR**	**low**
	[38] ('SP')	$>$ IC-FR	severe
	[3] ('SP + RNR')	$<$ IC-FR	severe
	[3] ('k-SP + RNR')	$\gg$ IC-FR	severe

the advantage of our solutions in terms of lower cost and/or lower congestion. While these conclusions are obtained under a fixed setting, we have validated them under other settings as shown in Appendix D of the supplementary file, available online.

VII. CONCLUSION

We studied the fundamental problem of joint caching and routing in a cache network with arbitrary topology, with the objective of minimizing routing cost under link/cache capacity constraints. After characterizing the complexity of this problem in all the cases, we developed polynomial-time algorithms that achieved guaranteed approximations in important special cases and superior empirical performance in the general case. While our focus was on one-shot optimization for a given set of demands, our solution was shown to work well in an online setting when combined with reasonable demand prediction. Meanwhile, our negative result also indicates further room of improvement for the worst-case performance in the general case.

ACKNOWLEDGMENT

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation here on.

REFERENCES

[1] T. Xie, S. Thakkar, T. He, P. McDaniel, and Q. Burke, "Joint caching and routing in cache networks with arbitrary topology," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst.*, 2022, pp. 56–66.

[2] S. Sardellitti, F. Costanzo, and M. Merluzzi, "Joint optimization of caching and transport in proactive edge cloud," in *Proc. 26th Eur. Signal Process. Conf.*, 2018, pp. 797–801.

[3] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1258–1275, Jun. 2018.

[4] M. Mahdian and E. Yeh, "MinDelay: Low-latency joint caching and forwarding for multi-hop networks," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–7.

[5] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug. 2016.

[6] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.

[7] M. Dehghan et al., "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1635–1648, Jun. 2017.

[8] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.

[9] X. Li, X. Wang, K. Li, Z. Han, and V. C. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926–6939, Oct. 2017.

[10] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug. 2016.

[11] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *Proc. IEEE INFOCOM*, 2012, pp. 2444–2452.

[12] B. Liu, K. Poularakis, L. Tassiulas, and T. Jiang, "Joint caching and routing in congestible networks of arbitrary topology," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 105–10 118, Dec. 2019.

[13] J. Liu, Q. Yang, and G. Simon, "Congestion avoidance and load balancing in content placement and request redirection for mobile CDN," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 851–863, Apr. 2018.

[14] K. Lim, Y. Bang, J. Sung, and J.-K. K. Rhee, "Joint optimization of cache server deployment and request routing with cooperative content replication," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 1790–1795.

[15] J. Perelló, K. Walkowiak, M. Klinkowski, S. Spadaro, and D. Careglio, "Joint content placement and lightpath routing and spectrum assignment in CDNs over elastic optical network scenarios," *Comput. Commun.*, vol. 77, pp. 72–84, 2016.

[16] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact algorithms for the joint object placement and request routing problem in content distribution networks," *Comput. Operations Res.*, vol. 35, no. 12, pp. 3860–3884, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054807000482

[17] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren, "Quantifying the benefits of joint content and network routing," in *Proc. ACM SIGMETRICS/Int. Conf. Meas. Model. Comput. Syst.*, 2013, pp. 243–254.

[18] D. Ren, S.-H. G. Chan, G. Shi, and H. Zhang, "Distributed joint optimization for large-scale video-on-demand," *Comput. Netw.*, vol. 75, pp. 86–98, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128614003478

[19] R. Moosavi, S. Parsaeefard, M. A. Maddah-Ali, V. Shah-Mansouri, B. H. Khalaj, and M. Bennis, "Energy efficiency through joint routing and function placement in different modes of SDN/NFV networks," 2020, *arXiv: 2007.13230*.

[20] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 374–388, Mar. 2019.

[21] K. Kamran, A. Moharrer, S. Ioannidis, and E. Yeh, "Rate allocation and content placement in cache networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[22] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug. 2018.

[23] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Proactive retention-aware caching with multi-path routing for wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1286–1299, Jun. 2018.

[24] T. Ho et al., "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[25] J. M. Kleinberg, "Single-source unsplittable flow," in *Proc. IEEE 37th Annu. Symp. Foundations Comput. Sci.*, 1996, Art. no. 68.

[26] Y. Asano, "Experimental evaluation of approximation algorithms for the minimum cost multiple-source unsplittable flow problem," in *ICALP Satell. Workshops*, 2000, pp. 111–122.

[27] G. Strang, "Karmarkar's algorithm and its place in applied mathematics," *Math. Intelligencer*, 1987.

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

[29] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions–I,” *Math. Program.*, vol. 14, no. 1, pp. 265–294, Dec. 1978.

[30] G. L. Nemhauser and L. A. Wolsey, “Best algorithms for approximating the maximum of a submodular set function,” *Math. Operations Res.*, vol. 3, no. 3, pp. 177–188, Aug. 1978.

[31] A. Ageev and M. Sviridenko, “Pipage rounding: A new method of constructing algorithms with proven performance guarantee,” *J. Combinatorial Optim.*, vol. 8, pp. 307–328, 2004.

[32] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, “FemtoCaching: Wireless content delivery through distributed caching helpers,” *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[33] M. Skutella, “Approximating the singel source unsplittable min-cost flow problem,” *Math. Program.*, vol. 91, pp. 493–514, 2002.

[34] Y. Dinitz, N. Garg, and M. X. Goemans, “On the single-source unsplittable flow problem,” *Combinatorica*, vol. 19, no. 1, pp. 17–41, 1999.

[35] T. Erlebach and A. Hall, “NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow,” in *Proc. 13th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2002, pp. 194–202.

[36] S. Achleitner, N. Bartolini, T. He, T. La Porta, and D. Zad Tootaghaj, “Fast network configuration in software defined networking,” *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1249–1263, Dec. 2018.

[37] C. Peng, Y. Tan, and L. T. Yang, “New algorithms for the minimum-cost single-source unsplittable flow problem,” in *Proc. IEEE Adv. Inf. Netw. Appl. Workshops*, 2007, pp. 136–141.

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

[39] M. X. Goemans and D. P. Williamson, “New 3/4-approximation algorithms for the maximum satisfiability problem,” *SIAM J. Discrete Math.*, vol. 7, no. 4, pp. 656–666, 1994.

[40] C. Barnhart, C. A. Hane, and P. H. Vance, “Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems,” *Operations Res.*, vol. 48, no. 2, pp. 318–326, 2000.

[41] S. Even, A. Itai, and A. Shamir, “On the complexity of timetable and multicommodity flow problems,” *SIAM J. Comput.*, vol. 5, no. 4, pp. 691–703, Dec. 1976.

[42] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, “Constrained non-monotone submodular maximization: Offline and secretary algorithms,” in *Proc. Int. Workshop Internet Netw. Econ.*, 2010, pp. 246–257.

[43] “What’s in a name? Understanding the Google Cloud network ‘edge’.” [Online]. Available: https://cloud.google.com/blog/products/networking/understanding-google-cloud-network-edge-points

[44] “AkamaiAPI gateway user guide: Caching.” [Online]. Available: https://learn.akamai.com/en-us/webhelp/api-gateway/api-gateway-user-guide/GUID-B717E657--4C07-4B76-934A-36F1C40F91AE.html

[45] S. K. Fayazbakhsh et al., “Less pain, most of the gain: Incrementally deployable ICN,” in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 147–158.

[46] N. Spring, R. Mahajan, and D. Wetherall, “Measuring ISP topologies with rocketfuel,” in *Proc. Conf. Appl. Technol. Architectures Protoc. Comput. Commun.*, 2002, pp. 133–145.

[47] “scikit-learn: Gaussian processes.” [Online]. Available: https://scikit-learn.org/stable/modules/gaussian_process.html

[48] P. M. Vaidya, “Speeding-up linear programming using fast matrix multiplication,” in *Proc. 30th Annu. Symp. Foundations Comput. Sci.*, 1989, pp. 332–337.

[49] M. Fisher, G. Nemhauser, and L. Wolsey, “An analysis of approximations for maximizing submodular set functions–II,” *Math. Prog. Study*, vol. 8, pp. 73–87, 1978.

[50] “REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms.” 2019. [Online]. Available: https://github.com/svissicchio/Repetita

Tian Xie (Student Member, IEEE) received the BEng degree in software engineering from Wuhan University, China, in 2019, awarded as the ‘Top Ten Outstanding Luojia Scholars of the Year 2018’. She is currently working toward the PhD degree in computer science and engineering with Pennsylvania State University, University Park, PA, in 2019. Advised by Prof. Ting He, her research interests include the area of network security and network modeling and joint optimization problems over cache network.

Sanchal Thakkar received the BE degree in electronics and communication engineering from the Vellore Institute of Technology, India, in 2017, and the MS degree in computer science and engineering from the Pennsylvania State University, in 2022. He is currently working as a software engineer for the SD-WAN team with Hewlett Packard Enterprise, San Jose, CA. His interests include network security, computer networks, algorithm design and distributed systems.

Ting He (Senior Member, IEEE) received the PhD degree in electrical and computer engineering from Cornell University, in 2007. She is an associate professor with the School of EECS, Penn State University. Her interests include computer networking, performance evaluation, statistical inference, and machine learning. She has served as associate editor for *IEEE Transactions on Communications* (2017-2020) and *IEEE/ACM Transactions on Networking* (2017-2023), General co-chair of IEEE RTCSA (2023), TPC co-chair of IEEE ICCCN (2022), and Area TPC chair of IEEE INFOCOM (2021). Her work received multiple Outstanding Contributor Awards from IBM, multiple awards from DAIS-ITA and NIS-ITA, and multiple paper awards such as the 2021 IEEE Communications Society Leonard G. Abraham Prize and the Best Paper Award at the 2013 ICDCS.

Patrick McDaniel (Fellow, IEEE) is the Tsun-Ming Shih professor of computer sciences with the School of Computer, Data & Information Sciences, University of Wisconsin-Madison. He is also a fellow of the ACM and AAAS and the director of the NSF Frontier Center for Trustworthy Machine Learning. He also served as the program Manager and lead scientist for the Army Research Laboratory’s Cyber-Security Collaborative Research Alliance from 2013 to 2018. His research focuses on a wide range of topics in computer and network security and technical public policy. Prior to joining UW-Madison in 2022, he was previously with Penn State since 2004, and a senior research staff member with AT&T Labs-Research prior.

Quinn Burke received the BS and MS degrees in computer science from the Pennsylvania State University with a focus on computer security. He is currently working toward the PhD degree in computer science with the University of Wisconsin-Madison. His research interests include network and systems security, software-defined networking, and virtualization technologies.