# Joint Caching and Routing in Cache Networks with Arbitrary Topology

Tian Xie, Sanchal Thakkar, Ting He, Patrick McDaniel, and Quinn Burke
Pennsylvania State University, University Park, PA, USA. Email: {tbx5027,sjt5721,tzh58,pdm12,qkb5007}@psu.edu

***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 an 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 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*—cache network, joint caching and routing, unsplittable flow problem, approximation algorithm.**

## 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 knobs 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 [1], [2], [3], and next generation cellular network proposes to offer content caching at various types of base stations [4], [5], [6], [7], [8], [9]. 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 [10] or a heterogeneous cellular network with small-cell and macrocell base stations [4], [5]. 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, existing solutions either relied on heuristics or resorted to the generic branch-and-bound method with exponential worst-case complexity [11]. Polynomial-time algorithms with approximation guarantees were not available until recently, when [2] proposed an approximation algorithm for minimizing the routing cost in the underloaded regime and [11] 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 [2]. However, our work differs from [2] in that: (i) while [2] ignored link capacity constraints, we consider both limited and unlimited link capacities, where the limited link capacities significantly complicate the routing problem; (ii) while [2] only optimized routing among a limited set of candidate paths (e.g., $k$-shortest paths to origin servers), we (effectively) optimize routing among all possible paths while maintaining a polynomial complexity. As shown later (Section V), 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 [1], [2], [3], Content Delivery Network (CDN) [12], [13], [14], content provider networks [10], [15], cellular networks [4], [5], [6], [7], [8], [9], and IoT networks [11]. Majority of existing works focused on specific topologies, e.g., a 3-tier hierarchical topology [10], [8], [7], or a 2-tier hierarchical topology [4], [5], [6]. These special topologies have very limited routing options, thus simplifying the problem.

Among works considering general network topology, only a few provided performance guarantees [2], [11], [9]. However, [9], [2] 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 [11] 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 [2], which represents the normal operation state of most networked systems, but we tackle a much more general problem than [2] 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,

This work was supported by the National Science Foundation under award CNS-1946022.

e.g., joint cache deployment, request routing, and content placement [13]. The content placement problem is also similar to the placement of virtual network functions, which is usually jointly optimized with routing [16], [17]. However, due to the high complexity of these problems, existing solutions are mostly based on heuristics. The few existing solutions that provide performance guarantees [18], [19], [20] address optimization problems that are very different from ours, and are thus not comparable with our work. Specifically, [18] optimized request rates and content placement, but assumed predetermined routes; [19] optimized VM allocation, content placement, and request routing, but ignored link capacities; [20] optimized request routing and content retention time, but only provided performance bounds in the case of uncapacitated caches.

### B. Summary of Contributions

Our contributions are summarized as follows:

1) We formulate the problem of joint caching and routing as a comprehensive optimization that minimizes the total routing cost under linear constraints, which 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*).

2) We analyze the complexity of the optimization in all the cases through connections to known NP-hard problems.

3) We develop efficient algorithms, with focus on the hardest case of integral caching and integral routing. In the special case of unlimited link capacities as considered in [2], we develop a polynomial-time algorithm that achieves a constant approximation in maximizing the cost saving due to caching. 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 for MSUFP when individual demands are 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.

4) We evaluate our solutions against state-of-the-art benchmarks in the common 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, and (ii) the advantage is preserved when our algorithms are based on predicted demand produced by a realistic prediction method.

**Roadmap.** Section II formulates our optimization problem, Section III analyzes its complexity, Section IV presents our algorithms and approximation analysis, Section V provides evaluation results, and finally Section VI concludes the paper. *All the proofs can be found in Appendix.*

## 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., data chunks). As common in the literature [2], [11], [18], we assume the content items to be of equal size, which represent fixed-size chunks constituting data files. 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 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 underloaded regime, i.e., all the requests can be satisfied. This regime represents the normal operation scenario (assuming suitable capacity planning). In contrast to the overloaded regime where the focus is on serving as many requests as possible [11], there are generally multiple ways to place and route content items to satisfy all the requests in the underloaded regime, so the focus is typically on finding a feasible solution that minimizes the total routing cost [2]. 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 performance metric. For example, if $w_{uv}$ denotes delay, then minimizing the total routing cost minimizes the average content access delay; 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 sub-chunks and store linear combinations of these sub-chunks at caches such that the original item can be recovered with high probability as long as sufficiently many coded sub-chunks are retrieved [21], where $x_{vi}$ denotes the fraction of coded sub-chunks for content item $i$ that are stored at node $v$. Other coding schemes such as MDS rateless codes [22] can also be used.

### 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.

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)*.


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

Note that integral routing implies integral source selection, as modeled by (1c). 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 (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 [23], [24]. 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 [2], 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 [25]).

*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 ($\sum_{(i,s)\in R}\lambda_{(i,s)} \le \min_{(u,v)\in E} c_{uv}$), 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 [2], 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 [2], 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:

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

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

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

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

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 $F_{\text{RNR}}$ can be written as a *monotone submodular function* [26] of content placement. To this end, we rewrite $F_{\text{RNR}}$ as a set function: for any $X\subseteq V\times C$,

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

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 [26].

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 [27]. A better approximation ratio of $(1-1/e)$ is achieved by the randomized algorithm in [26], which cannot be further improved under the value oracle model [28]. However, this randomized algorithm has a complexity of $O(n^8)$ where $n$ is the rank of the matroid [26]. 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 [2] 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

[1]The issue was addressed in [2] 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.

caching and routing problem in [2] in that the optimal content placement according to (5) together with RNR solves the joint caching and routing problem in [2] 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 [26] at a much lower complexity. The idea is to use pipage rounding [29]. 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 $x_{vi}, r_v^{(i,s)} \in [0,\ 1]$ $(\forall v \in V, i \in C, (i,s) \in R)$, $(1 - 1/e)L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \leq F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \leq L_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$, where

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

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:

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

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

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

$$(2b), (2c), \quad (7d)$$

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

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 $L_{\text{RNR}}$ and hence achieves a $(1-1/e)$-approximation in maximizing $F_{\text{RNR}}$ 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 $F_{\text{RNR}}$ by repeating the following step: As long as $\exists \tilde{x}_{vi}, \tilde{x}_{vj} \in (0, 1)$, we will update their values by

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

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

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

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 $F_{\text{RNR}}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \geq F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$.

*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.

**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 $\boldsymbol{r}$
1 compute pairwise least costs $(w_{v\to s})_{v,s\in V}$ and the maximum pairwise cost $w_{\max}$;
2 solve the LP (7) for a fractional solution $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$;
3 round $\tilde{\boldsymbol{x}}$ to an integral solution $\boldsymbol{x}$ by (8)–(9);
4 compute an integral $\boldsymbol{r}$ based on $\boldsymbol{x}$ using RNR;

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

*Remark:* Although a similar approach was taken in [2], 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 [2] (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 [22], 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→requester paths have equal cost $w_1$ with $w_1 < \min_{u\in U} w_{h_0\to u}$, [22] 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

[2]The precise approximation ratio is $1 - (1 - \frac{1}{d})^d$, where $d := \max_{u\in U} \deg(u) - 1$ [22], which converges to $1 - 1/e$ as $d$ gets large.

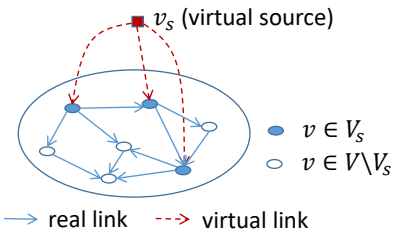



Fig. 2. Auxiliary graph $G'$ that augments $G$ by adding a virtual source $v_s$ connected to all real sources in $V_s$.

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 [22], 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. 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:

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

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

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

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

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

known as the *minimum-cost single-source unsplittable flow problem (MSUFP)* [30]. 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 $\lambda_i$, 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 [23]. 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 [31] 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 $(\alpha, \beta)$-approximation if: (i) the total load on each link imposed by $\boldsymbol{f}$ is within $\alpha$ times its capacity, and (ii) the total cost incurred by $\boldsymbol{f}$ is within $\beta$ 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 $(3 + 2\sqrt{2}, 1)$ [30]. If the maximum demand is within the minimum link capacity, the best known ratio is $(3, 1)$ [30]; under the same condition, [32] 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 $\epsilon$ congestion on each link for an arbitrarily small $\epsilon > 0$.

*Subroutine:* The basis of our solution is an algorithm developed in [30], which converts a splittable flow to an unsplittable flow with the following properties.

**Lemma IV.6** ([30])**.** 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 $0 = q_1 \leq q_2 \leq \cdots \leq q_n$, and a splittable flow $\boldsymbol{f}$ satisfying all the demands, [30, 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 $\sum_{i \neq i_e : e \in p_i} \lambda_i < f(e)$, the total flow on link $e$ under $\boldsymbol{f}$.

*Proposed algorithm:* Using [30, 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 $\lambda_i$ to[3]

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

The rounded demand satisfies $\lambda_i 2^{-1/K} \leq \bar{\lambda}_i \leq \lambda_i$. 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:

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

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 [30, 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 $\lambda_i$ 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 $\lambda_{\min} := \lambda_1 \leq \cdots \leq \lambda_n =: \lambda_{\max}$, Algorithm 2 computes an unsplittable flow that serves demand $\lambda_i$ by path $p_i$ $(i = 1, \ldots, n)$ in $O\left(n^{2.5}(|V| + |E|)^{2.5} + K|E|\left(\log(\frac{\lambda_{\max}}{\lambda_{\min}}) + |V|\right)\right)$ 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 [30] (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 $\epsilon$.

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

**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 $\lambda_i$, 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 [33], 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 [30, Algorithm 2], specified by paths $(p_i)_{i \in S_j}$;

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

While the case of $\lambda_{\max} \ll c_{\min}$ was considered in [34], 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

In the general case with arbitrary link/cache capacities, we alternatingly optimize content placement and routing (including source selection), based on experiences gained in studying the special cases.

*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 [35], 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 $P_{\boldsymbol{r},\boldsymbol{f}}^{(i,s)}$ denote the set of cycle-free paths used to serve requests of type $(i, s)$ and $\lambda_p^{(i,s)}$ $(\forall p \in P_{\boldsymbol{r},\boldsymbol{f}}^{(i,s)})$ 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 $\left((f_p^{(i,s)})_{p \in P_{\boldsymbol{r},\boldsymbol{f}}^{(i,s)}}\right)_{(i,s) \in R}$ can be computed as in [33] 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 $|P_{\boldsymbol{r},\boldsymbol{f}}^{(i,s)}| \leq |E|$ $(\forall (i, s) \in R)$ (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

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

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 [35], 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 [35].

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

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

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

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

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 [36], [35] as in Lemma IV.2. Using auxiliary variables to represent $\min\left(1,\, \sum_{k'=0}^{k-1} x_{p_{|p|-k'}i}\right)$ 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 $V^{\boldsymbol{x}}_i := \{v \in V :\, x_{vi} = 1\}$ be the set of nodes storing content $i$ under placement $\boldsymbol{x}$ $(\forall i \in C)$. We can construct an auxiliary graph $G^{\boldsymbol{x}} := (V \cup \{v_i\}_{i\in C},\, E \cup \bigcup_{i\in C}\{(v_i, v) :\, v \in V^{\boldsymbol{x}}_i\})$, 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 [24]. A number of heuristics for MMUFP, e.g., greedy and LP relaxation with randomized rounding, have been proposed [24]. The optimal solution can also be computed by the branch-and-price-and-cut algorithm [37], 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 $\alpha$ 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 $\alpha$ [30], but in the multiple-source case, computing such a flow is NP-hard [38]. As we need to solve MMUFP to obtain source selection and routing even if the optimal content placement is known, a fundamental challenge in solving IC-IR with approximation guarantee is to develop a good approximation algorithm for MMUFP, which is still 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 $(\boldsymbol{r}^{(0)},\, \boldsymbol{f}^{(0)})$ to source selection and routing, 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.

*Remark:* While this algorithm remains a heuristic for now, it has exhibited good performance (see Fig. 7–12) and quick convergence (within 10 iterations) in all our evaluations.

## V. 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 [39] and distributors like Akamai [40], and has been shown to achieve most of the benefits of ICN [41].

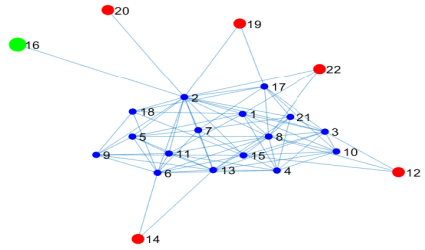



Fig. 3. Abovenet topology; ●: origin server, ●: edge nodes, ●: internal nodes.

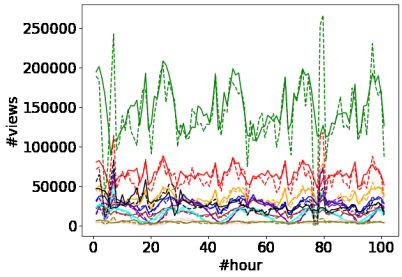



Fig. 4. Numbers of views per hour for top-10 YouTube videos; solid: ground truth, dashed: prediction.

*Simulation setting:* To simulate edge caching, we use an Internet Service Provider (ISP) topology called Abovenet from [42] 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 $\leq 3$) are designated as edge nodes, which receive requests from users and host caches. The others are internal nodes that only forward requests/responses. See Fig. 3 for the topology.

We assume that each edge node can cache $\zeta$ items. We measure routing costs by (propagation) delays. As the origin server is usually much farther away from users than caches, we select the delay for the outgoing link of the server randomly from $[100, 200]$ms, and the delays for the other links randomly from $[1, 20]$ms. We generate requests in two ways:

- *Synthetic:* Requests are generated for $|C|$ items according to [2], which used the Zipf distribution with skewness 1.2 and a total rate of 1 request/ms per edge node.
- *Trace-driven:* Requests are generated according to #views per hour of the top $|C|$ YouTube videos we collected over 100 consecutive hours between 11/14/2021 and 11/18/2021 (additional 550 hours of #views have been collected for training purposes). We randomly distribute the requests for each video among the edge nodes.

Following the setting in [2] for a topology with similar size as Abovenet, we set $|C| = 10$ and $\zeta = 2$ by default, which will be varied later.

We give each link a default capacity of $\kappa$, which is set to $4\%$ of the total request rate. In synthetic simulation, this amounts to a bandwidth of 0.2 items/ms, which equals 15 Gbps for an item size of 9.375 MB; in trace-driven simulation, this amounts to a bandwidth of $13,715.796$ videos/hour, which equals 14.7 Gbps for the average size of 480.67 MB

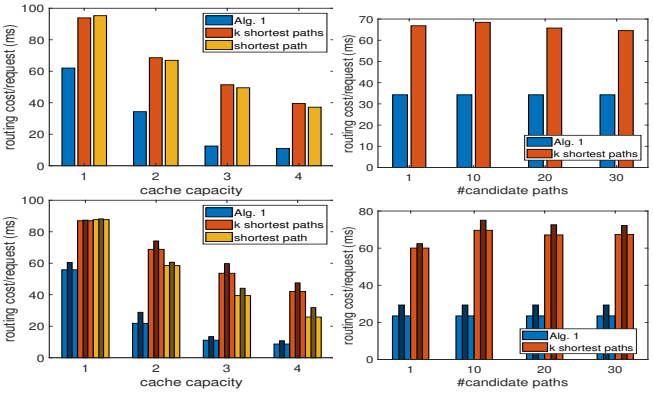



Fig. 5. Unlimited link capacities: first row – synthetic simulation, second row – trace-driven simulation (light: true demand; dark: predicted demand).

for the top-10 videos[4]. 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 server as a last resort. Our evaluation focuses on the case of IC-IR. All results are averaged over 100 Monte Carlo runs.

Our trace-driven 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 [43], 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[5]. 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.

*Results:* First, in the special case of unlimited link capacities, we compare our Algorithm 1 against the solution in [2] ('k shortest paths') and content placement as in [35] based on shortest path routing ('shortest path'). We configure the solution in [2] 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. The results in Fig. 5 show that: (i) our algorithm substantially outperforms these state-of-the-art solutions, (ii) the advantage remains as we increase the number of candidate paths for [2], and (iii) the same holds even if our algorithm runs on the predicted demand and the benchmarks run on the true demand (*all the performances are evaluated based on the true demand*). This is because [2], [35] both predetermine the candidate paths based on the server's location, hence not fully utilizing the caches.

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

[4]While the videos have variable sizes, here we only use the #views of these popular videos to represent realistic content request patterns. Our assumption of equal item size can be satisfied by partitioning each video into equal-sized chunks. We leave the chunk-level simulation to future work.

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

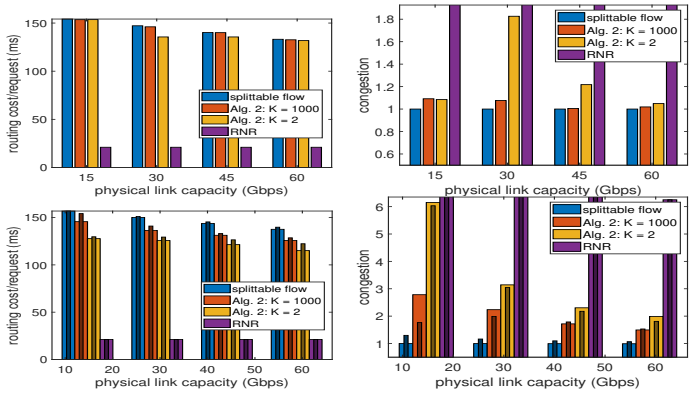



Fig. 6. Binary cache capacities: first row – synthetic simulation, second row – trace-driven simulation (light: true demand; dark: predicted demand).

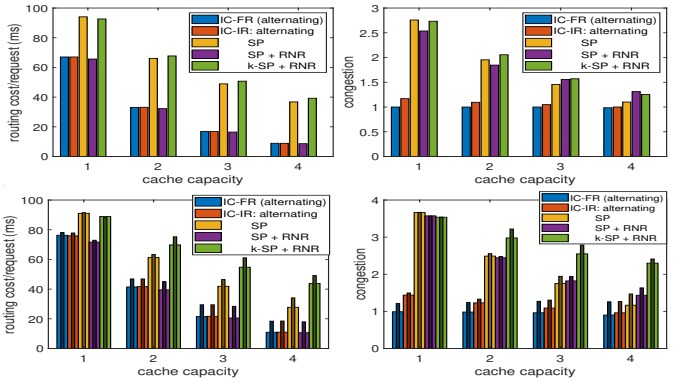



Fig. 7. General case under varying cache capacity: first row – synthetic simulation, second row – trace-driven simulation (light: true demand; dark: predicted demand).

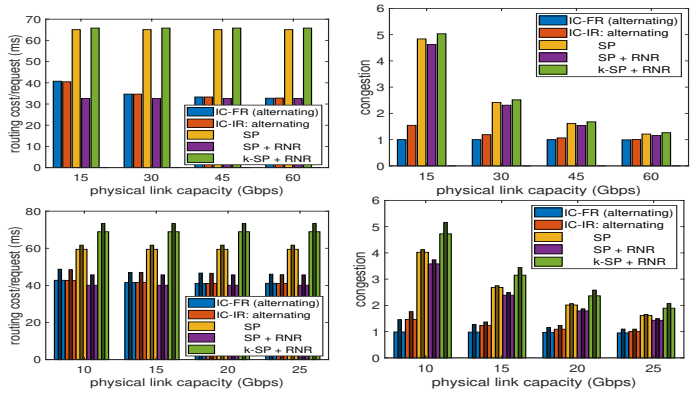



Fig. 8. General case under varying link capacity: first row – synthetic simulation, second row – trace-driven simulation (light: true demand; dark: predicted demand).

(with parameter $K$ tuned to minimize congestion under the default link capacity) with the state-of-the-art MSUFP algorithm proposed in [30], which is a special case of our algorithm with $K = 2$. As benchmarks, we also compare with the splittable flow and the solution by [2], which routes each request to the nearest replica ('RNR'). In addition to routing cost, we also evaluate congestion, 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 25 times; the congestion plots have been truncated for better visibility of other results), and (ii) compared to the state of the art [30] ('K = 2'), Algorithm 2 with a larger $K$ can substantially reduce the congestion while achieving/beating the minimum routing cost achievable without congestion (given by 'splittable flow'). This 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.

Finally, we consider the general case. We compare our algorithm proposed in Section IV-C ('alternating'), which solves MMUFP by LP relaxation with randomized rounding, with the solution in [35] based on shortest path routing ('SP'), a variation of [2] with the shortest path as the only candidate path ('SP + RNR'), and [2] with its recommended way of constructing candidate paths as the $k$ ($k = 10$) shortest paths ('k-SP + RNR'). The results in Fig. 7–8 show that: (i) our algorithm significantly outperforms [35], [2] in both cost and congestion, and (ii) while 'SP + RNR' also achieves a low cost, it causes much more congestion. Our algorithm also converges quickly (within 10 iterations) in all the cases.

While the above observations are obtained under a relatively small catalog size (set according to the evaluations in [2]), we have verified that they remain valid (and even become more prominent) as the catalog size increases, as shown in Fig. 12. Evaluations under larger catalog sizes are left to future work.

## VI. 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 given demands, our solution was shown to work well in an online setting when combined with reasonable demand prediction.

## References

[1] S. Sardellitti, F. Costanzo, and M. Merluzzi, "Joint optimization of caching and transport in proactive edge cloud," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 797–801.

[2] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1258–1275, 2018.

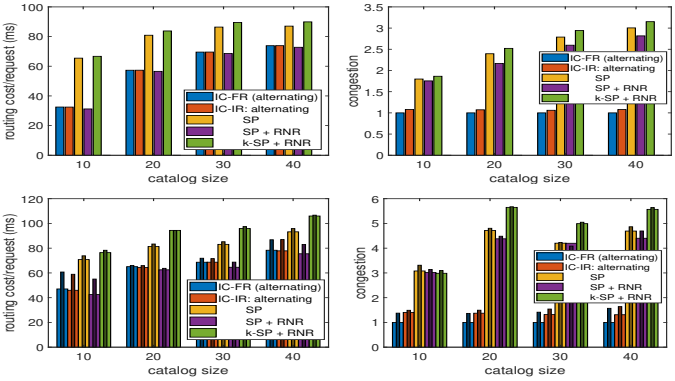



Fig. 9. General case under varying catalog size: first row – synthetic simulation, second row – trace-driven simulation (light: true demand; dark: predicted demand).

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

[4] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2275–2284, 2016.

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

[6] 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, June 2017.

[7] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1382–1393, 2016.

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

[9] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1863–1876, 2015.

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

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

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

[13] 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 *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 1790–1795.

[14] 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," *Computer Communications*, vol. 77, pp. 72–84, 2016.

[15] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren, "Quantifying the benefits of joint content and network routing," in *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, 2013, pp. 243–254.

[16] 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," *arXiv preprint arXiv:2007.13230*, 2020.

[17] 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 Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[18] K. Kamran, A. Moharrer, S. Ioannidis, and E. Yeh, "Rate allocation and content placement in cache networks," in *IEEE INFOCOM*, May 2021.

[19] 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 Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1751–1767, 2018.

[20] 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 Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1286–1299, 2018.

[21] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[22] 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.

[23] J. M. Kleinberg, "Single-source unsplittable flow," in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, 1996, p. 68.

[24] Y. Asano, "Experimental evaluation of approximation algorithms for the minimum cost multiple-source unsplittable flow problem," in *ICALP Satellite Workshops*, 2000.

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

[26] G. Calinescu, C. Chekuri, M. Pál, 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.

[27] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions–i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, December 1978.

[28] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 177–188, August 1978.

[29] A. Ageev and M. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, pp. 307–328, 2004.

[30] M. Skutella, "Approximating the singel source unsplittable min-cost flow problem," *Mathematical Programming*, vol. 91, pp. 493–514, 2002.

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

[32] T. Erlebach and A. Hall, "Np-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow," in *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002, pp. 194–202.

[33] S. Achleitner, N. Bartolini, T. He, T. La Porta, and D. Zad Tootaghaj, "Fast network configuration in software defined networking," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1249–1263, 2018.

[34] C. Peng, Y. Tan, and L. T. Yang, "New algorithms for the minimum-cost single-source unsplittable flow problem," in *Proceedings of the Advanced Information Networking and Applications Workshops*, 2007, pp. 136–141.

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

[36] M. X. Goemans and D. P. Williamson, "New 3/4-approximation algorithms for the maximum satisfiability problem," *SIAM Journal on Discrete Mathematics*, vol. 7, no. 4, pp. 656–666, 1994.

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

[38] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, December 1976.

[39] "What's in a name? Understanding the Google Cloud network 'edge'," https://cloud.google.com/blog/products/networking/understanding-google-cloud-network-edge-points.

[40] "Akamai API Gateway User Guide: Caching," https://learn.akamai.com/en-us/webhelp/api-gateway/api-gateway-user-guide/GUID-B717E657-4C07-4B76-934A-36F1C40F91AE.html.

[41] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable ICN," in *SIGCOMM*, 2013, p. 147–158.

[42] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*, August 2002.

[43] "scikit-learn: gaussian processes," https://scikit-learn.org/stable/modules/gaussian_process.html.

[44] P. M. Vaidya, "Speeding-up linear programming using fast matrix multiplication," in *30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 332–337.

[45] 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.

[46] "REPETITA: repeatable experiments for performance evaluation of traffic-engineering algorithms." [Online]. Available: https://github.com/svissicchio/Repetita

## APPENDIX A
## SUPPORTING PROOFS

### A. Proof of Lemma IV.1

*Proof.* It is easy to see that $\widetilde{F}_{\text{RNR}}(X)$ is monotone increasing in $X$, as caching one more item can only help to increase (or preserve) the cost saving under proper source selection.

For submodularity, we need to show that for any $X^{(1)} \subseteq X^{(2)} \subseteq V \times C$ and any $(v,i) \notin X^{(2)}$,

$$\widetilde{F}_{\text{RNR}}(\{(v,i)\} \cup X^{(1)}) - \widetilde{F}_{\text{RNR}}(X^{(1)}) \geq \widetilde{F}_{\text{RNR}}(\{(v,i)\} \cup X^{(2)}) - \widetilde{F}_{\text{RNR}}(X^{(2)}). \quad (16)$$

To this end, consider the set of requesters for content $i$ that will benefit from storing a replica of content $i$ at node $v$:

$$S_{vi}(X^{(j)}) := \{s \in V : (i,s) \in R,\ w_{v\to s} < \min_{u:(u,i)\in X^{(j)}} w_{u\to s}\}, \quad (17)$$

where $j = 1, 2$. Since $X^{(1)} \subseteq X^{(2)}$, $S_{vi}(X^{(1)}) \supseteq S_{vi}(X^{(2)})$, and $\min_{u:(u,i)\in X^{(1)}} w_{u\to s} \geq \min_{u:(u,i)\in X^{(2)}} w_{u\to s}$ for each $s \in V$. Therefore,

$$\begin{aligned}\text{LHS of (16)} &= \sum_{s\in S_{vi}(X^{(1)})} \lambda_{(i,s)} \Big( (\min_{u:(u,i)\in X^{(1)}} w_{u\to s}) - w_{v\to s} \Big) \\ &\geq \sum_{s\in S_{vi}(X^{(2)})} \lambda_{(i,s)} \Big( (\min_{u:(u,i)\in X^{(2)}} w_{u\to s}) - w_{v\to s} \Big) \\ &= \text{RHS of (16)},\end{aligned}$$

which completes the proof. □

### B. Proof of Lemma IV.2

*Proof.* We will use the Goemans-Williamson inequality [36], [35]: for any $y_i \in [0,\ 1]$ for $i = 1, \ldots, n$,

$$(1 - \frac{1}{e}) \min(1, \sum_{i=1}^{n} y_i) \leq 1 - \prod_{i=1}^{n} (1 - y_i) \leq \min(1, \sum_{i=1}^{n} y_i).$$

By definition (3),

$$F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) = \sum_{(i,s)\in R} \lambda_{(i,s)} \sum_{v\in V} w_{\max} \cdot \left(1 - r_v^{(i,s)} \left(1 - \frac{x_{vi}(w_{\max} - w_{v\to s})}{w_{\max}}\right)\right). \quad (18)$$

As $y_1 := 1 - r_v^{(i,s)} \in [0,1]$ and $y_2 := \frac{x_{vi}(w_{\max} - w_{v\to s})}{w_{\max}} \in [0,1]$, applying the Goemans-Williamson inequality yields that

$$\begin{aligned}&(1 - \frac{1}{e}) \min\left(1, 1 - r_v^{(i,s)} + \frac{x_{vi}(w_{\max} - w_{v\to s})}{w_{\max}}\right) \\ &\leq 1 - r_v^{(i,s)} \left(1 - \frac{x_{vi}(w_{\max} - w_{v\to s})}{w_{\max}}\right) \\ &\leq \min\left(1, 1 - r_v^{(i,s)} + \frac{x_{vi}(w_{\max} - w_{v\to s})}{w_{\max}}\right), \quad (19)\end{aligned}$$

Plugging (19) into (18) completes the proof. □

### C. Proof of Lemma IV.3

*Proof.* Without loss of generality, suppose that $\tilde{\boldsymbol{x}}$ achieves "=" in (2c), as otherwise its value can be increased to achieve "=" without decreasing $F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$.

If $\exists \tilde{x}_{vi} \in (0,1)$, then there must exist another fractional variable $\tilde{x}_{vj} \in (0,1)$ for $j \neq i$ $(i,j \in C)$, as $\sum_{i'\in C} \tilde{x}_{vi'} = c_v \in \mathbb{N}$. Treating $x_{vi}, x_{vj}$ as the only variables, we can formulate a simple LP:

$$\begin{aligned}\max_{x_{vi},x_{vj}} \quad & F_{\text{RNR}}(x_{vi}, x_{vj} | \tilde{\boldsymbol{x}}_{-\{vi,vj\}}, \tilde{\boldsymbol{r}}) && (20a)\\ \text{s.t.} \quad & x_{vi} + x_{vj} \leq \tilde{x}_{vi} + \tilde{x}_{vj}, && (20b)\\ & x_{vi}, x_{vj} \in [0,1], && (20c)\end{aligned}$$

where $F_{\text{RNR}}(x_{vi}, x_{vj} | \tilde{\boldsymbol{x}}_{-\{vi,vj\}}, \tilde{\boldsymbol{r}})$ denotes $F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r})$ when $\boldsymbol{r} = \tilde{\boldsymbol{r}}$ and $\boldsymbol{x} = \tilde{\boldsymbol{x}}$ at all the entries except for $x_{vi}, x_{vj}$. As $F_{\text{RNR}}(x_{vi}, x_{vj} | \tilde{\boldsymbol{x}}_{-\{vi,vj\}}, \tilde{\boldsymbol{r}})$ can be written as

$$\begin{aligned}x_{vi} \sum_{s:(i,s)\in R} \lambda_{(i,s)} \tilde{r}_v^{(i,s)} (w_{\max} - w_{v\to s}) \\ + x_{vj} \sum_{s:(j,s)\in R} \lambda_{(j,s)} \tilde{r}_v^{(j,s)} (w_{\max} - w_{v\to s}) + c, \quad (21)\end{aligned}$$

where $c$ is a constant not depending on $x_{vi}$ or $x_{vj}$, there exists an optimal solution that makes $x_{vi}$ or $x_{vj}$ integral. Specifically, if $\sum_{s:(i,s)\in R} \lambda_{(i,s)} \tilde{r}_v^{(i,s)} (w_{\max} - w_{v\to s}) \geq \sum_{s:(j,s)\in R} \lambda_{(j,s)} \tilde{r}_v^{(j,s)} (w_{\max} - w_{v\to s})$, then $(x_{vi}, x_{vj})$ from (8) is an optimal solution to (20), which sets $x_{vj} = 0$ if $\tilde{x}_{vi} + \tilde{x}_{vj} \leq 1$ or $x_{vi} = 1$ if $\tilde{x}_{vi} + \tilde{x}_{vj} > 1$; similar argument holds in the other case. Moreover, as $(\tilde{x}_{vi}, \tilde{x}_{vj})$ is a feasible solution to (20), the optimal solution $(x_{vi}, x_{vj})$ guarantees that $F_{\text{RNR}}(x_{vi}, x_{vj} | \tilde{\boldsymbol{x}}_{-\{vi,vj\}}, \tilde{\boldsymbol{r}}) \geq F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$. Thus, one application of (8) or (9) reduces the number of fractional variables by at least one without decreasing $F_{\text{RNR}}$, at a time complexity of $O(|V|)$. By repeating this step $O(|V||C|)$ times, we can obtain an integral solution $\boldsymbol{x}$ such that $F_{\text{RNR}}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \geq F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$. □

### D. Proof of Theorem IV.4

*Proof.* Let $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$ denote the optimal solution to (7). Then

$$\begin{aligned}F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}}) &\geq (1 - \frac{1}{e}) L_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}}) && (22)\\ &\geq (1 - \frac{1}{e}) L_{\text{RNR}}(\boldsymbol{x}^*, \boldsymbol{r}^*) && (23)\\ &\geq (1 - \frac{1}{e}) F_{\text{RNR}}(\boldsymbol{x}^*, \boldsymbol{r}^*), && (24)\end{aligned}$$

where (22) and (24) are due to Lemma IV.2, and (23) is due to the optimality of $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$ for (7). By Lemma IV.3, the rounded $\boldsymbol{x}$ satisfies $F_{\text{RNR}}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \geq F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}})$. Finally, given an integral $\boldsymbol{x}$, it is easy to see that RNR, i.e., $r_v^{(i,s)} = 1$ for $v = \arg\min_{u:x_{ui}=1} w_{u\to s}$ $(\forall (i,s) \in R)$, minimizes $C_{\text{RNR}}$ and hence maximizes $F_{\text{RNR}}$. Therefore,

$$F_{\text{RNR}}(\boldsymbol{x}, \boldsymbol{r}) \geq F_{\text{RNR}}(\boldsymbol{x}, \tilde{\boldsymbol{r}}) \geq F_{\text{RNR}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{r}}) \geq (1 - \frac{1}{e}) F_{\text{RNR}}(\boldsymbol{x}^*, \boldsymbol{r}^*).$$

In terms of complexity, line 1 can be done in $O(|V|(|E| + |V| \log |V|))$ time using Dijkstra's algorithm, line 2 can be done in $O(|R|^{2.5}|V|^{2.5})$ time using Vaidya's LP algorithm [44]

(as the number of variables and the number of constraints are both $O(|R||V|)$), line 3 takes $O(|V|^2|C|)$ time as in Lemma IV.3, and line 4 takes $O(|V||R|)$ time. Thus, the total complexity is $O(|V||E|+|R|^{2.5}|V|^{2.5})$ (assuming $|C| = O(|R|)$). □

### E. Proof of Lemma IV.5

*Proof.* Given a feasible joint source selection and routing solution $(\boldsymbol{r}, \boldsymbol{f})$ in $G$, we can construct an equivalent solution $\boldsymbol{f}'$ in $G'$ such that $f'^{(i,s)}_{uv} = f^{(i,s)}_{uv}$ for all $(i,s) \in R$ and $(u,v) \in E$, and $f'^{(i,s)}_{v_s v} = r^{(i,s)}_v$ for all $(i,s) \in R$ and $v \in V_s$. As the virtual links have no capacity constraints or costs, $\boldsymbol{f}'$ is a feasible routing solution in $G'$ with a single source $v_s$, and achieves the same cost as $\boldsymbol{f}$. Similarly, given a feasible routing solution $\boldsymbol{f}'$ in $G'$ that serves all the requests from $v_s$, the same construction yields a feasible joint source selection and routing solution $(\boldsymbol{r}, \boldsymbol{f})$ with the same cost. □

### F. Proof of Theorem IV.7

*Proof.* To show (i), since the optimal splittable flow $\boldsymbol{f}$ has a cost no larger than the cost of the optimal unsplittable flow, it suffices to show that $\sum_{i=1}^n \lambda_i \sum_{e\in p_i} w_e \le \sum_{i=1}^n \sum_{p\in P_i} f_p^{(i)} \sum_{e\in p} w_e$. To this end, we first note that reducing the flow in the descending order of path cost (line 4) ensures that for any commodity $i$ and any path $p$ with $\bar{f}_p^{(i)} > 0$,

$$(\lambda_i - \bar{\lambda}_i) \sum_{e\in p} w_e \le \sum_{p'\in P_i} (f_{p'}^{(i)} - \bar{f}_{p'}^{(i)}) \sum_{e\in p'} w_e. \tag{25}$$

Meanwhile, the partition in (12) ensures that for each $i \in S_j$,

$$\log\left(\frac{\lambda_{\max}}{\bar{\lambda}_i}\right) = -\frac{\lfloor K \log(\frac{\lambda_i}{\lambda_{\max}}) \rfloor}{K} = q_i - \frac{j}{K} \tag{26}$$

for $q_i := -\frac{\lfloor K \log(\lambda_i/\lambda_{\max})\rfloor}{K} + \frac{j}{K} \in \mathbb{N}$, which implies that $\bar{\lambda}_i = \lambda_{\max} 2^{j/K - q^{(j)}} \cdot 2^{q^{(j)} - q_i}$ for $q^{(j)} := \max_{i\in S_j} q_i$. That is, the rounded demands $(\bar{\lambda}_i)_{i\in S_j}$ satisfy the condition in Lemma IV.6. By Lemma IV.6, the converted unsplittable flow for $S_j$ $(j = 0, \dots, K-1)$ satisfies

$$\sum_{i\in S_j} \bar{\lambda}_i \sum_{e\in p_i} w_e \le \sum_{i\in S_j} \sum_{p\in P_i} \bar{f}_p^{(i)} \sum_{e\in p} w_e. \tag{27}$$

As the rounding in [30, Algorithm 2] ensures that each $p_i$ satisfies $\bar{f}_{p_i}^{(i)} > 0$, (25) implies

$$\sum_{i=1}^n (\lambda_i - \bar{\lambda}_i) \sum_{e\in p_i} w_e \le \sum_{i=1}^n \sum_{p\in P_i} (f_p^{(i)} - \bar{f}_p^{(i)}) \sum_{e\in p} w_e. \tag{28}$$

Moreover, as $\bigcup_{j=0}^{K-1} S_j = \{1, \dots, n\}$, (27) implies

$$\sum_{i=1}^n \bar{\lambda}_i \sum_{e\in p_i} w_e \le \sum_{i=1}^n \sum_{p\in P_i} \bar{f}_p^{(i)} \sum_{e\in p} w_e. \tag{29}$$

Summing (28) and (29) proves (i).

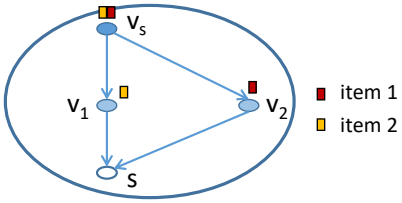



Fig. 10. Example for unbounded approximation ratio.

To show (ii), consider a given $e \in E$. Let $i_j := \arg\max_{i\in S_j: e\in p_i} \bar{\lambda}_i$, $\bar{f}_j(e)$ the load imposed on $e$ by $\bar{\boldsymbol{f}}_j$, and $f(e)$ the load imposed on $e$ by $\boldsymbol{f}$. We have

$$\sum_{i: e\in p_i} \lambda_i \le \sum_{j=0}^{K-1} \left( \lambda_{i_j} + 2^{1/K} \sum_{i\in S_j: e\in p_i, i\ne i_j} \bar{\lambda}_i \right) \tag{30}$$

$$< \sum_{j=0}^{K-1} \left( \lambda_{i_j} + 2^{1/K} \bar{f}_j(e) \right) \tag{31}$$

$$\le \sum_{j=0}^{K-1} \lambda_{i_j} + 2^{1/K} f(e) \tag{32}$$

$$< \frac{2^{1/K}}{2(2^{1/K}-1)} \lambda_{\max} + 2^{1/K} c_e, \tag{33}$$

where (30) is because $\lambda_i \le 2^{1/K} \bar{\lambda}_i$ as implied by (11), (31) is by Lemma IV.6, (32) is because $\bar{\boldsymbol{f}}$ is reduced from $\boldsymbol{f}$, and (33) is due to $f(e) \le c_e$ (as $\boldsymbol{f}$ is a feasible flow) and the fact that $\forall i \in S_j$, $\lambda_i \le \lambda_{\max}$ for $j = K-1$, and $\lambda_i < 2^{(j+1-K)/K} \lambda_{\max}$ for $j = 0, \dots, K-2$, implied by $-\frac{\lfloor K \log(\lambda_i/\lambda_{\max})\rfloor}{K} + \frac{j}{K} \ge 1$.

For complexity, line 1 solves an LP with $O(n|E|)$ variables and $O(|E| + n|V|)$ constraints, which takes $O(n^{2.5}(|V| + |E|)^{2.5})$ time by Vaidya's algorithm [44]. Line 2 takes $O(\sum_{i=1}^n |V||P_i|) = O(n|V||E|)$ time as $|P_i| \le |E|$. This is because in constructing $P_i$, we set initial "link capacities" to $(f_e^{(i)})_{e\in E}$, and iteratively find a $s \to d_i$ path with positive residual capacity and route the maximum flow on the path. As each iteration reduces the residual capacity of at least one link to zero, the number of iterations, i.e., the number of constructed paths $|P_i|$, is at most $|E|$. Lines 3–4 take $O(\sum_{i=1}^n |P_i| \log |P_i|) = O(n|E| \log |E|)$ time, dominated by the sorting of the path costs for each $P_i$ (the path costs can be computed while constructing $P_i$). Line 5 takes $O(n)$ time. Line 7 takes $O(|S_j||V| + |E| \log(\frac{\lambda_{\max}}{\lambda_{\min}}) + |V||E|)$ time by Lemma IV.6, and thus lines 6–7 take $O\left(n|V| + K|E| \left(\log(\frac{\lambda_{\max}}{\lambda_{\min}}) + |V|\right)\right)$ time. Summing up all these yields the overall complexity of Algorithm 2. □

### G. Proof of Proposition ??

*Proof.* We prove the claim by constructing an example with an arbitrarily large approximation ratio. Consider the scenario in Fig. 10, where client $s$ requests item 1 with rate $\lambda_{(1,s)} = \lambda$ and item 2 with rate $\lambda_{(2,s)} = \epsilon$. Suppose that the cache capacities of $v_1$, $v_2$, and $v_s$ are 1, 1, and 2, respectively. Suppose that all the links have capacity $\lambda$, $w_{v_1 s} = \epsilon$, and $w_{v_s v_1} = w_{v_s v_2} = w_{v_2 s} = w$. If the initial solution is to store item 1 on $v_2$ and item 2 on $v_1$, and serve requests for item 1 from $v_2$ and requests for item 2 from $v_1$, then it is easy to see that this

will be the final solution of alternating optimization (i.e., it is an NE), with a total cost of $\lambda w + \epsilon^2$. However, the optimal solution is to store item 1 on $v_1$ and item 2 on $v_2$, and serve requests for item 1 from $v_1$ and requests for item 2 from $v_2$, which has a total cost of $\epsilon(\lambda + w)$. The approximation ratio is unbounded as $\lim_{\epsilon \to 0} (\lambda w + \epsilon^2)/\big(\epsilon(\lambda + w)\big) = \infty$. □

### *H. Proof of Lemma* **??**

*Proof.* It is easy to see that $(V \times C, \mathcal{I})$ is an independence system, as $X = \emptyset$ is feasible for (**??**), and a subset of a feasible set remains feasible. Consider any $X \subseteq V \times C$ and any two maximal feasible subsets $X_1,\ X_2 \subseteq X$. To add an element $(v,i) \in X_2 \setminus X_1$ to $X_1$, we have to take out a set $X'$ of at most $\lceil / \rceil$ elements from $X_1$ such that $(X_1 \setminus X') \cup \{(v,i)\}$ remains feasible for (**??**). Repeating this swap for each element in $X_2 \setminus X_1$ shows that the cardinalities of the bases of $X$ differ by at most $p := \lceil / \rceil$ fold. Hence, $(V \times C, \mathcal{I})$ is a $p$-independence system by Definition **??**. □

### *I. Proof of Theorem* **??**

*Proof.* It is known [45] that for maximizing a monotone submodular function subject to a $p$-independence constraint, the greedy algorithm has an approximation ratio of $1/(1+p)$. □

### *J. Proof of Lemma* **??**

*Proof.* By (14), increasing elements of $\boldsymbol{x}$ can only increase the function value, which proves the monotonicity of . Moreover, for any $X^{(1)} \subseteq X^{(2)} \subseteq V \times C$ and any $(v,i) \notin X^{(2)}$, let $K_{p,i}(X^{(j)})$ $(j = 1, 2)$ denote the index of the node on path $p$ with the nearest replica when serving item $i$ to node $p_{|p|}$ along path $p$ under content placement $X^{(j)}$. It is easy to see that $K_{p,i}(X^{(1)}) \le K_{p,i}(X^{(2)})$. Thus, the increase in  by selecting $(v,i)$ on top of $X^{(j)}$ satisfies

$$
\begin{aligned}
&(\{(v,i)\} \cup X^{(1)}) - (X^{(1)}) \\
&= \sum_{i:(i,s)\in R} \sum_{\substack{p \in P_{(i,s)}: \\ p_{k_v} = v}} \lambda_p^{(i,s)} \max\left(0, \sum_{k=K_{p,i}(X^{(1)})}^{|p|-1} w_{p_k p_{k+1}} - \sum_{k=k_v}^{|p|-1} w_{p_k p_{k+1}}\right) \\
&\ge \sum_{i:(i,s)\in R} \sum_{\substack{p \in P_{(i,s)}: \\ p_{k_v} = v}} \lambda_p^{(i,s)} \max\left(0, \sum_{k=K_{p,i}(X^{(2)})}^{|p|-1} w_{p_k p_{k+1}} - \sum_{k=k_v}^{|p|-1} w_{p_k p_{k+1}}\right) \\
&= (\{(v,i)\} \cup X^{(2)}) - (X^{(2)}),
\end{aligned}
$$

which proves the submodularity of . □

## APPENDIX B
## ADDITIONAL EXPLANATIONS

### *A. Conversion between the Case of Binary Cache Capacities and MSUFP*

Given the auxiliary graph $G'$ constructed from the original topology $G$ and the locations of caches $V_s$ as in Fig. 2, the corresponding MSUFP problem aims at finding an unsplittable flow in $G'$ within the link capacities at the minimum cost, to route each commodity $(i,s) \in R$ of demand $\lambda_{(i,s)}$ from the virtual source $v_s$ to the content requester $s$. For example, if

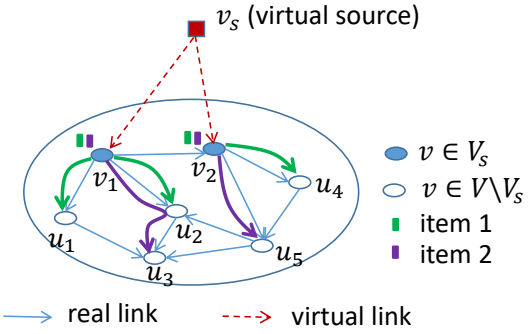



Fig. 11. Example: MSUFP in auxiliary graph $G'$.

$R = \{(1,u_1), (1,u_2), (1,u_4), (2,u_3), (2,u_5)\}$ in the network illustrated in Fig. 11, then the constructed MSUFP problem has $n = 5$ commodities, all with the same source $v_s$, and the commodity $(1,u_1) \in R$ has destination $u_1$ and demand $\lambda_{(1,u_1)}$. After solving the MSUFP problem, e.g., by Algorithm 2, we will obtain a routing path for each commodity in $G'$, ignoring the first hop of which gives the routing path in the original topology. For example, if the path given by MSUFP for commodity $(1,u_1)$ in Fig. 11 is $v_s \to v_1 \to u_1$, then the source for serving item 1 to requester $u_1$ is $v_1$, and the path is $v_1 \to u_1$.

### *B. Ideas in Solving MSUFP*

We first interpret the utility of the existing rounding algorithm in [30] according to Lemma IV.6. Given an optimal splittable flow $\boldsymbol{f}$ that minimizes the routing cost under the given link capacities, we know that the cost of $\boldsymbol{f}$ is no more than the minimum cost of an unsplittable flow under the same link capacities, and the traffic load $f(e)$ it imposes on each link $e \in E$ is no more than the link capacity $c_e$. Lemma IV.6 states that if the demands only differ by factors that are integer powers of 2, then we can use an algorithm in [30] to convert $\boldsymbol{f}$ to an unsplittable flow that routes each commodity on a single path, such that (i) the routing cost is no larger than the cost of any feasible unsplittable flow, and (ii) the excess load on each link (beyond its capacity) is no more than the rate of the largest flow traversing it. Hence, the key is to convert arbitrary demands $(\lambda_i)_{i=1}^n$ to demands that differ by integer powers of 2.

To this end, we round each demand $\lambda_i$ down to $\bar{\lambda}_i$ as in (11) and partition the rounded demands into $K$ subsets $\{S_j : j = 0, \ldots, K-1\}$ as in (12). This partition guarantees that the rounded demands in each subset $(\bar{\lambda}_i)_{i \in S_j}$ only differ by integer powers of 2. This is because by (12), $\forall i \in S_j$, $\exists q_i \in \mathbb{N}$ such that

$$
-\frac{\lfloor K \log(\lambda_i / \lambda_{\max}) \rfloor}{K} + \frac{j}{K} = q_i, \tag{34}
$$

which implies that

$$
\bar{\lambda}_i = \lambda_{\max} 2^{\lfloor K \log(\lambda_i/\lambda_{\max}) \rfloor / K} = \lambda_{\max} 2^{j/K - q_i}. \tag{35}
$$

Therefore, if $i^*$ is the commodity in $S_j$ with the minimum rounded demand, then

$$
\frac{\bar{\lambda}_i}{\bar{\lambda}_{i^*}} = 2^{q_{i^*} - q_i}, \tag{36}
$$

TABLE I
EXECUTION TIME UNDER CHUNK-LEVEL SIMULATION

| scenario | algorithm | avg execution time (s) |
|---|---|---|
| $c_{uv} = \infty$ | **Alg. 1** | **0.7260** |
| | [2] ('k shortest paths') | 11.1574 |
| | [35] ('shortest path') | 0.0171 |
| $c_v = 0/\|C\|$ | **Alg. 2** ($K = 1000$) | **1.4030** |
| | [30] | 1.4032 |
| | [2] ('RNR') | 0.0154 |
| general | **alternating** | **9.6714** |
| | [35] ('SP') | 0.0313 |
| | [2] ('SP + RNR') | 0.0493 |
| | [2] ('k-SP + RNR') | 11.3286 |

TABLE II
EXECUTION TIME UNDER FILE-LEVEL SIMULATION

| scenario | algorithm | avg execution time (s) |
|---|---|---|
| $c_{uv} = \infty$ | **greedy** | **0.0226** |
| | [2] ('k shortest paths') | 0.4364 |
| | [35] ('shortest path') | 0.0155 |
| $c_v = 0/\|C\|$ | **Alg. 2** ($K = 1000$) | **1.5836** |
| | [30] | 1.5837 |
| | [2] ('RNR') | 0.0167 |
| general | **alternating** | **1.0910** |
| | [35] ('SP') | 0.0161 |
| | [2] ('SP + RNR') | 0.0459 |
| | [2] ('k-SP + RNR') | 10.4771 |

where $q_{i^*} - q_i \in \mathbb{N}$ because $\bar{\lambda}_{i^*} \leq \bar{\lambda}_i$. Note that this argument also holds for $\lambda_i = \lambda_{\max}$, which belongs to $S_{K-1}$. The rounding and partitioning decompose the original MSUFP problem with arbitrary demands into $K$ smaller MSUFP problems with demands satisfying the condition in Lemma IV.6, which allows us to use the algorithm in [30] to convert the rounded splittable flows into unsplittable flows with optimal cost and bounded congestion; see details in the proof of Theorem IV.7.

Finally, we comment on the role of the design parameter $K$. Although $K$ can be any positive integer, we expect a larger value of $K$ to yield less congestion. This is because in rounding the demands, we reduce each demand by a factor of at least $2^{-1/K}$. Intuitively, a larger $K$ will lead to a smaller demand reduction, and thus less congestion when computing the routes based on the reduced demands but using the routes to support the original demands. This intuition has been confirmed by our evaluation results in Fig. 6.

## APPENDIX C
## EXECUTION TIME EVALUATION

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, shown in Tables I–II. All the times are measured under IC-IR, which is the most computationally challenging case. We see that the proposed algorithms are sufficiently fast to be applied to adjust caching and routing decisions on a regular basis.

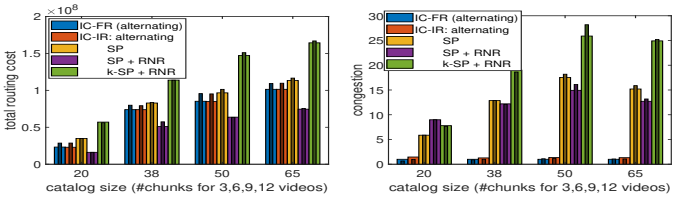



Fig. 12. Varying catalog size by varying #videos (light: true demand; dark: predicted demand).

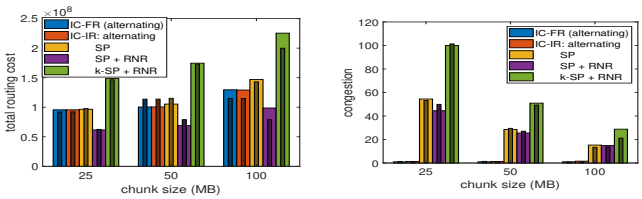



Fig. 13. Varying chunk size under the same #videos (light: true demand; dark: predicted demand).

## APPENDIX D
## ADDITIONAL EXPERIMENTS

While the observations in Section V are based on a fixed set of environment parameters, such as network topology, catalog size, chunk size, and demand prediction method, we will show via additional simulations that these observations remain valid as these parameters vary. In the sequel, we will only focus on the case of caching/routing chunked videos under limited cache and link capacities, as this is the case of primary interest in this work.

### A. Varying Number of Videos

As the catalog includes chunks (of 100MB each) from more videos, shown in Fig. 12, the relative performance of the evaluated algorithms exhibits the same trend as in Table **??**. Meanwhile, both the routing cost and the congestion tend to increase, as more demands are contending for the same amount of cache and link capacities.

### B. Varying Chunk Size

Another way to grow the catalog size is to decrease the size of each chunk while keeping the set of videos the same. The top-10 videos in our trace correspond to $|C| = 199$ chunks of size 25MB, $|C| = 103$ chunks of size 50MB, and $|C| = 54$ chunks of size 100MB. Fig. 13 shows the result based on demands from the first 10 hours; the same trend has been observed for other hours. We see that the proposed algorithms ('alternating') achieve slightly smaller costs as the chunk size decreases, because smaller chunks allow for more fine-grained optimization of caching and routing. Meanwhile, the congestion of the benchmark algorithms ('SP', 'SP + RNR', 'k-SP + RNR') gets worse, as they become greedier in cost reduction.

### C. Varying Prediction Accuracy

As mentioned before, the prediction of the demands is not our focus. Nevertheless, it is desirable to understand the sensitivity of our solutions to prediction errors. To this end, we perform a sensitivity analysis by synthetically generating prediction errors according to a normal distribution $\mathcal{N}(0, \sigma^2)$,

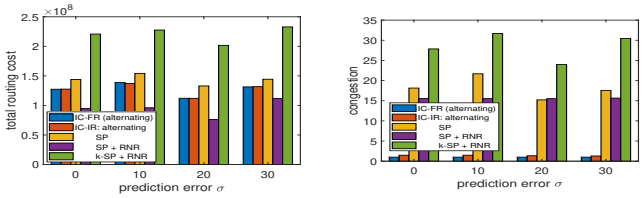



Fig. 14. Varying prediction error $\sigma$ ($\sigma = 0$ represents true demand).

TABLE III
TOPOLOGIES AND PARAMETERS IN EVALUATION

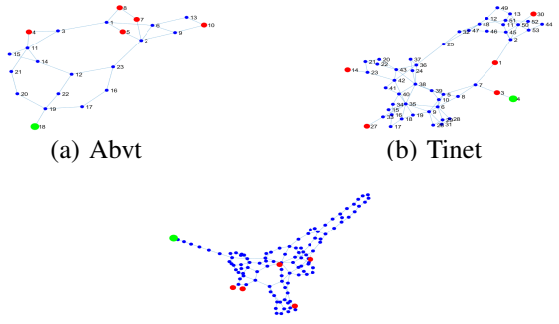
| Topology | $\|V\|$ | $\|E\|$ | link capacity |
|---|---|---|---|
| Abvt | 23 | 31 | 1 Gbps |
| Tinet | 53 | 89 | 1 Gbps |
| Deltacom | 113 | 161 | 1 Gbps |

and testing the performance of joint caching and routing under different levels of error as measured by the root mean squared error $\sigma$. The results in Fig. 14 indicate that our proposed algorithms ('alternating') are reasonably robust to prediction errors, maintaining notable advantages over the benchmarks from [35], [2] in both cost and congestion over a wide range of $\sigma$ values.

### D. Varying Network Topology

Finally, we evaluate the impact of network size and topology by repeating our experiments on three networks of different sizes, with parameters in Table III. To make the setting more realistic, we use real bandwidths as link capacities. All the data are from[6] [46]. We simulate edge caching in each network under the same setting as in Section V, except that the topology and the default link capacity are changed according to the dataset. In each network, we set the lowest-degree node as the origin server and the next 5 lowest-degree nodes as the edge nodes, as shown in Fig. 15. Fig. 16 shows the performance for each network. We see that the topology does have a notable impact on the absolute performance in cost and congestion, where a network tends to have a higher cost or congestion if the size is larger or the edge nodes are more scattered. However, our proposed algorithms consistently outperform the benchmarks in all the simulated networks.

[6]Although 'Abvt' also stands for Abovenet, the topology in [46] is from a different source (Internet Topology Zoo) and different from the topology in Fig. 3 (which is from Rocketfuel). We switch to this dataset to experiment with real link capacities.



(a) Abvt (b) Tinet

(c) Deltacom

Fig. 15. Network topology with varing size; ●: origin server, ●: edge nodes, ●: internal nodes.

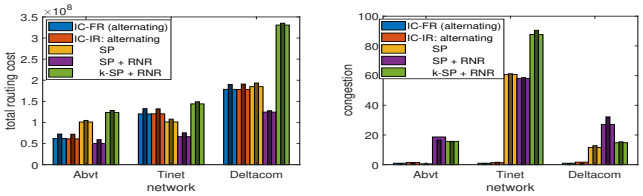



Fig. 16. Varying network topology with real link capacities (light: true demand; dark: predicted demand).