Performance Evaluation 143 (2020) 102138

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Fair caching networks

Yuezhou Liu [a], Yuanyuan Li [a], Qian Ma [b,*], Stratis Ioannidis [a], Edmund Yeh [a]

[a] Northeastern University, Boston, MA, United States of America
[b] Sun Yat-sen University, Shenzhen, Guangdong, China

ARTICLE INFO

Article history:
Available online 28 September 2020

Keywords:
Caching network
Fairness
Utility maximization
Caching gain

ABSTRACT

We consider caching networks in which the routing cost for serving a content request can be reduced by caching the requested content item in cache nodes closer to the users. We refer to the cost reduction enabled by caching as the caching gain, and the product of the caching gain of a content request and its request rate as *caching gain rate*. We aim to study *fair* content allocation strategies through a utility-driven framework, where each request achieves a utility of its caching gain rate, and consider a family of α-fair utility functions to capture different degrees of fairness. The resulting problem is an NP-hard problem with a non-decreasing submodular objective function. Submodularity allows us to devise a deterministic allocation strategy with an optimality guarantee factor arbitrarily close to $1 - 1/e$. When $0 < \alpha \leq 1$, we further propose a randomized strategy that attains an improved optimality guarantee, $(1 - 1/e)^{1-\alpha}$, in expectation. Through extensive simulations over synthetic and real-world network topologies, we evaluate the performance of our proposed strategies and discuss the effect of fairness.

1. Introduction

In-network caching is a fundamental enabler of many applications, such as information-centric networks (ICNs) [1–3], content delivery networks (CDNs) [4,5], and micro/femtocell networks [6]. Storing content items in caching networks closer to users can reduce the routing cost (e.g., delay or financial expense) of content requests and alleviate the bandwidth pressure at content servers. There has been a rich body of literature on how to optimally allocate content items in cache storage to serve content requests, predominantly focusing on caching to optimize utilities of either cache hit rates [7–9] or throughput [10,11].

Motivated by a series of recent papers [6,12–15], we study fairness considerations in the context of the so-called *caching gain rate* [6,12]. Informally, given a caching strategy X, the caching gain rate of a flow of requests for an item is given by:

$$\lambda \cdot \Delta C(X),$$

where λ is the rate with which the item is requested, and $\Delta C(X)$ is the reduction of routing costs due to the caching strategy, compared to empty caches. Intuitively, this metric incorporates both the popularity of an item, as captured by λ, as well as the benefit of caching decisions in routing (measured in hops, distance traveled, or delay incurred). Moreover, in contrast to, e.g., cache hit rates or throughputs/rates alone, it incorporates networking costs in the network design

* Corresponding author.
E-mail addresses: liu.yuez@husky.neu.edu (Y. Liu), yuanyuanli@ece.neu.edu (Y. Li), maqian25@mail.sysu.edu.cn (Q. Ma), ioannidis@ece.neu.edu (S. Ioannidis), eyeh@ece.neu.edu (E. Yeh).

https://doi.org/10.1016/j.peva.2020.102138
0166-5316/

objective. Such costs are important: requests served with hit rate 1 at a distant server, in reality, have a lower utility than requests served locally with a lower hit rate.

Several recent works study optimization algorithms determining caching decisions that maximize the average caching gain rate [6,12–15]. In practice, content requests are heterogeneous, and derive different benefits from network storage resources. It is essential to find cache allocations which do not simply maximize the overall reduction in routing cost, but achieve a balance among the benefits different requests derive from caching. Thus, a key question is how to *fairly* allocate the cache storage resource among different requests. While there is an extensive literature on the fair allocation of bandwidth resources in traditional communication networks [16–19] and of storage resources in cache networks [7–9,11], to the best of our knowledge, fairness has not been explored in the context of utilities of the caching gain rate.

We propose a fair caching framework to achieve different degrees of fairness w.r.t. requests, content items, as well as users, respectively, in a caching network with arbitrary topology. We consider the family of α-fair utility functions which unify different notions of fairness [20], and aim to find an optimal storage resource allocation that maximizes the total utility as a function of the caching gain rate. To the best of our knowledge, this is the first work that studies fair caching w.r.t. caching gain rates in a multi-hop caching network with arbitrary topology. Our main contributions can be summarized as follows:

- We begin by studying request-based fairness, associating each request with a utility. In this context, we formulate fair caching as a utility maximization problem, which is NP-hard.
- We show that this fair caching problem is a submodular maximization problem under matroid constraints. We provide a deterministic solution based on the continuous-greedy algorithm which has an optimality guarantee factor arbitrarily close to $1 - 1/e \approx 0.63$.
- We provide a stationary randomized caching strategy which has a better optimality guarantee $(1 - 1/e)^{1-\alpha}$ in expectation when $\alpha \in (0, 1]$. This is tighter than the approximation obtained by the continuous greedy algorithm.
- We extend our fair caching framework to incorporate content and user fairness. Both of our proposed algorithms apply in these settings and achieve the same optimality guarantees as for request fairness.
- We formally characterize the optimal allocation for a simple linear-topology example, illustrating how content allocations are affected as α increases.
- We extensively evaluate our algorithms over several synthetic and real-world graphs and show that our proposed algorithms outperform traditional path replication caching policies. We also discuss the effect of fairness and price of fairness based on numerical results.

Our analysis provides new insights on how fairness w.r.t. caching gain rate affects caching decisions. We observe, for example, that the intuitive behavior of caching highly requested content towards the edge indeed occurs for $\alpha < 1$, but is reversed when $\alpha > 1$ (see, e.g., Figs. 3 and 8). From a technical standpoint, our analysis establishes the submodularity of classic α-fairness objectives when applied to caching gain rates, making it amenable to polynomial-time approximation with a $1 - 1/e$ approximation. Moreover, our algorithm under the stationary randomized regime is novel, and improves upon the above approximation ratio in the $\alpha \in (0, 1]$ regime.

The remainder of this paper is structured as follows: In Section 2, we review related work. We introduce the system model and present two toy examples over a linear network in Section 3. In Section 4, we demonstrate the submodularity of our problem and introduce caching algorithms with optimality guarantees. In Section 5, we study a stationary randomized caching strategy. We present extensions in Section 6. In Section 7, we conduct extensive simulations over several topologies. Finally, we conclude and discuss future directions in Section 8.

2. Related work

Caching Gain Rate Optimization. The maximization of caching gain rates has been considered in several recent papers. Shanmugam et al. [6] consider content placement and delivery to maximize the caching gain rate in a femtocaching system. They develop an approximation algorithm based on the pipage rounding method by Ageev and Sviridenko [21]. Ioannidis and Yeh [12] extend this work to study an online, adaptive and distributed setting for arbitrary network topologies, considering joint routing and caching in subsequent work [14]. Shukla and Abouzeid [22] consider both caching gain rates and content storage costs to determine where and how long a content should be cached. Yang et al. [23] study the caching problem in a parallel computing framework. They determine the data placement in distributed memory to minimize the computation costs. Li et al. [15] optimize the content allocation in a resilient network, where each cache may fail in some cases. Mahdian et al. [13] and Li and Ioannidis [24] maximize overall caching gain rate while take queuing into consideration.

To the best of our knowledge, we are the first to study fairness in the context of caching gain rates. We use the same network model as Ioannidis and Yeh [12], extended via α-fair utilities of caching gain rates. We note that departing from the linear utility ($\alpha = 0$) of [12] necessitates using altogether different approximation techniques.

General In-network Caching Optimization. A number of works focus on optimizing global networking objectives other than the caching gain. Content placements that maximize the number of requests served by caches are studied in hierarchical caching networks [25], in cellular networks with moving users [26], in arbitrary congestible networks [27],

and in multi-cell mobile edge computing networks with storage, computation, and communication constraints [28]. To study the interplay between content search and content placement, Domingues et al. [29] consider the metric of expected delay experienced by all the requester. The same metric is considered by Poularakis et al. [30] to study the content placement of layered-video.

TTL Caches. Time-to-Live (TTL) caches have drawn attention recently due to their connection to classic replacement policies. TTL caches assign a timer to each content, and an eviction occurs when a timer expires. Che et al. [31] show that the hit probability of LRU caches can be approximated through a TTL-based eviction scheme; a theoretical justification of this result has been provided recently by Jiang et al. [32]. This approach has also been refined and extended to other traditional replacement policies [33,34] as well as more general requests arrival processes [35], thus providing a general framework for analyzing different replacement policies. Several papers study the approximate and exact behavior of, e.g., hit probabilities and cache occupancies, in both individual TTL-caches as well as networks thereof [36,37].

Fairness in Caching Networks. As in the case for classic communication networks, fairness in caching networks is discussed using specific notions (e.g. proportional fairness and max–min fairness) or by considering concave utility functions (e.g. α-fair utility functions). Several works study fair storage allocations in TTL cache networks. TTL timers are used as tuning knobs to control the hit rate as well as the utility of each content. Dehghan et al. in [7] propose a utility-driven framework for a single cache by considering the utility of the cache hit rate of each content item, and provide the optimal online solution. Panigrahy et al. [38] further consider the utility of the cache hit rate of each content item in a general TTL cache network. Fairness among content providers is considered by Chu et al. [8], where the utility is associated with the total hit rate achieved by each content provider. The fractions of storage allocated to each content provider and the association between caches and users are determined to maximize the global utility.

Besides TTL-based caching, there are also works that study fairness w.r.t. direct content placement in caches. Wang et al. [39] analyze the proportional fairness of the total cost (storage costs and one-hop fetching costs to fetch contents from other nodes) of each node through a Nash bargaining game. Avrachenkov et al. [40] study the fair caching problem in a video-on-demand system. The utility is determined by the portion of a video of each quality stored at each node. They formulate the utility maximization as a potential game and develop a distributed algorithm. Bonald et al. [11] discuss content placement to achieve fairness of throughput for different contents in ICNs. Rezvani et al. [41] study proportional fairness and max–min fairness of the user delay in radio access networks where the user delay is determined by caching, radio allocation and user association.

We depart from all the aforementioned works in two ways by considering fairness w.r.t. the utilities of caching gain rates, as opposed to, e.g., hit rates, user delay, throughput, or abstract costs. In addition, with the exception of Panigrahy et al. [42], that indirectly consider routing costs via a discount factor capturing utility degradation over a path, all aforementioned works consider specific network topologies and ignore the multi-hop routing costs; in contrast, our study applies to algorithms of arbitrary topologies and demands.

Submodularity and Approximation Algorithms. The maximization of submodular objective subject to matroid constraints appears in many combinatorial optimization problems. A classic result by Nemhauser et al. [43] shows that the greedy algorithm produces a solution with $1/2$ optimality guarantee. Calinescu [44] and Vodrak et al. [45] show that continuous-greedy algorithm produces a solution with a $1 - 1/e$ optimality guarantee. Sviridenko et al. [46] propose an algorithm based on modifications of continuous-greedy algorithm and non-oblivious local search, which produces a solution within $(1 - c/e)$ of the optimal, where c is the total curvature of the objective function. For a brief review of the topic and applications, please refer to [47] and [48]. We show that α-fair utilities of the caching gain rate maintain submodularity. Using this fact, we employ the greedy and continuous-greedy algorithm to solve the submodular optimization problem in Section 4.

Ageev and Sviridenko [21] use a convex relaxation to solve a class of submodular maximization problems that include the maximum coverage problem. Shanmugam et al. [6] and Ioannidis and Yeh [12] use the same method to create approximate solutions. In Section 5, we show that this method can also be coupled with the fair caching framework we consider; doing so requires extending it beyond the linear functions considered in [6,12,21] in a non-trivial way. Our extension also improves the approximation guarantee of [6,12,21,44] when α is in (0,1].

3. Model

We consider a network of caches, where each node can potentially store a finite number of content items. Some nodes generate requests for content items which are routed over given routes. A request can be satisfied at a designated server node that permanently stores the requested item, while it can also be satisfied earlier upon hitting a cache that contains the same item. We wish to determine a content item allocation strategy under the following desiderata: on the one hand, we wish to reduce the total routing cost for requests in the network, and on the other hand, we also wish to fairly allocate the storage resource to different requests. We describe our model formally below. Table 1 summarizes our notation.

3.1. Caching network

Following Ioannidis and Yeh [12], we represent the caching network by a directed graph $G(V, E)$, where V is a set of cache nodes and E is a set of bidirectional edges with asymmetric edge costs. We denote by $\mathcal{C}$ the set of items of equal size (see Section 6 for an extension to contents with unequal sizes) to be stored in the caches. For each $i \in \mathcal{C}$, there is a set of *designated server nodes* $\mathcal{S}_i \subseteq V$ which store i permanently.

Table 1
Notation summary.

$G(V, E)$	Network graph, with nodes V and edges E
$\mathcal{C}$	Content item catalog
c_v	Cache capacity at node $c \in V$
w_{uv}	Routing cost of edge $(u, v) \in E$
$\mathcal{R}$	Set of requests (i, p), with $i \in \mathcal{C}$ and p a path
$\lambda_{(i,p)}$	Request rate for $(i, p) \in \mathcal{R}$
x_{vi}	Variable indicating $v \in V$ stores $i \in \mathcal{C}$
y_{vi}	Marginal probability that $v \in V$ stores $i \in \mathcal{C}$
X	$\|V\| \times \|C\|$ matrix of x_{vi}, for $v \in V$, $i \in \mathcal{C}$
Y	$\|V\| \times \|C\|$ matrix of y_{vi}, for $v \in V$, $i \in \mathcal{C}$
$\mathcal{D}_1$	Set of feasible caching strategy $X \in \{0, 1\}^{\|V\|\times\|C\|}$
$\mathcal{D}_2$	Convex hull of $\mathcal{D}_1$
U	The utility of caching gain rate
$F_{(i,p)}$	Caching gain (4) for request $(i, p) \in \mathcal{D}_1$
$L_{(i,p)}$	The concave approximation (22) of $F_{(i,p)}$
G	Total utility of all requests
K	The multilinear extension of G
m_k	Update direction of the continuous-greedy algorithm
$[X]_{+(v,i)}$	Matrix X with coordinate (v, i) being set to 1
$[X]_{-(v,i)}$	Matrix X with coordinate (v, i) being set to 0
H	The concave approximation (23a) of G
$\nabla F, \partial F$	Gradient and subgradient of function F, respectively

3.1.1. Caching strategy

Each node $v \in V$ is equipped with a cache that can store $c_v \in \mathbb{N}_+$ items. Let

$$x_{vi} \in \{0, 1\}, \quad \text{for all } v \in V, i \in \mathcal{C},$$

be the indicator variable indicating whether node v stores item i. We denote by the matrix $X = [x_{vi}]_{v\in V, i\in\mathcal{C}} \in \{0, 1\}^{|V|\times|\mathcal{C}|}$, the caching (content item allocation) strategy of the network. As each node can cache a finite number of items, we have capacity constraints:

$$\sum_{i\in\mathcal{C}} x_{vi} \leq c_v, \quad \text{for all } v \in V. \tag{1}$$

Note that if node v is a designated server for $i \in \mathcal{C}$, it stores i in its permanent storage, which is separate from its cache.

3.1.2. Content requests

We denote each content request by a pair (i, p), where $i \in \mathcal{C}$ is the item requested and p is the path over which the request message is routed. A path p of length $M \leq |V|$ is a sequence of nodes $\{p_1, p_2, p_3, \ldots, p_M\}$, where $p_m \in V$ and $(p_m, p_{m+1}) \in E$ for all $m \in \{1, 2, \ldots, M-1\}$. We denote the finite set of all such requests in the network by $\mathcal{R}$. We say that a request $(i, p) \in \mathcal{R}$ is *well-routed* if: *(1)* the path p contains no loops; *(2)* the last node in the path is a designated server node for i, i.e., $p_M \in S_i$; and *(3)* no other node in the path is a designated server node for i. As in earlier work on caching gain rate maximization [12,13,15], we assume that the path p towards a designated server is pre-established by an external routing algorithm, and is not part of the problem optimization. We note that this does not trivialize the problem, as determining caching decisions is NP-hard even when paths are given.

We also make the standard assumption [9,12,15,17,18,36] that requests arrive according to independent Poisson processes with rate $\lambda_{(i,p)} \geq 0$. A request (i, p) goes along the path p, until it reaches a node that stores item i. After a cache hit, the network generates a response message carrying the requested item, which follows path p in the reverse direction to the request node (see Fig. 1). The system is stable, and there are no losses, so the request rate $\lambda_{(i,p)}$ is equal to the throughput of responses for all $(i, p) \in \mathcal{R}$.

3.1.3. Routing cost

We denote by $w_{uv} \geq 0$ the routing cost (e.g., financial expense or delay) incurred when transferring an item across edge $(u, v) \in E$. Compared with content items, the size of request messages is relatively small. Thus, we assume the request forwarding costs are negligible [12]. Hence, the routing cost for serving request (i, p) is:

$$C_{(i,p)}(X) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'}i}). \tag{2}$$

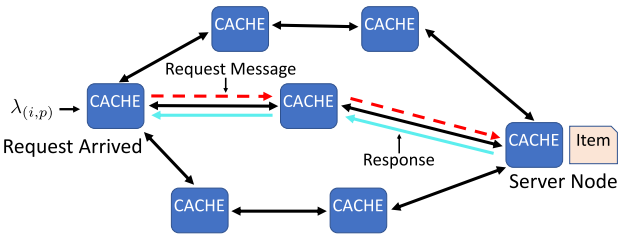


Fig. 1. A caching network with exogenous requests. A request $(i, p) \in \mathcal{R}$ arrives at a node and is routed to the server node that permanently stores the requested item. After a cache hit (either at the server node or at an intermediate node), the item is send back on the reverse path.

Intuitively, this formula shows that $C_{(i,p)}$ includes the cost of an edge (p_{k+1}, p_k) in the path p if item i is not stored by nodes $p_1, \ldots, p_k$. Without caching, i.e., $X = \mathbf{0}$, request (i, p) is served by the designed server node with a constant cost:

$$C_{(i,p)}(\mathbf{0}) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}. \tag{3}$$

For a given request, we define the difference between the routing cost without caching and the routing cost with caching as the *caching gain*:

$$F_{(i,p)}(X) = C_{(i,p)}(\mathbf{0}) - C_{(i,p)}(X) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - x_{p_{k'}i})\right). \tag{4}$$

Reducing the routing cost of request (i, p) is equivalent to increasing the caching gain, i.e., the cost reduction attained by caching at intermediate nodes.

3.2. Utility function and utility maximization

As the cache storage in a caching network is limited, we aim to allocate the cache storage resource fairly for requests using a utility-driven framework. We consider the utility of the *caching gain rate* associated with each request. Note that our algorithms are applicable for not only request, but also content and user fairness (see Section 6). We discuss fairness w.r.t. requests first for the sake of concreteness. Mathematically, given caching gain $F_{(i,p)}(X)$ and request rate $\lambda_{(i,p)}$, request (i, p) achieves utility $U(\lambda_{(i,p)}F_{(i,p)}(X))$. To capture fairness, we consider a class of α-fair utility functions, parameterized by $\alpha \in \mathbb{R}_+$. In particular, we assume that

$$U(z) = \begin{cases} \dfrac{z^{1-\alpha}}{1-\alpha} & \text{when } 0 \le \alpha < 1, \\ \log(z + \epsilon) & \text{when } \alpha = 1, \text{ or} \\ \dfrac{(z+\epsilon)^{1-\alpha}}{1-\alpha} & \text{when } \alpha > 1, \end{cases} \tag{5}$$

where $\epsilon \ge 0$ is a constant. For all α, the utility functions are continuously differentiable, non-decreasing, and strictly concave. This class of utility functions is classic, and unifies different notions of fairness, including *max–min* ($\alpha \to \infty$) and *proportional fairness* ($\alpha = 1$) as special cases [18].

Our goal is to maximize the total utility under the cache storage constraint:

$$\text{Maximize:} \quad G(X) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)}F_{(i,p)}(X)) \tag{6a}$$

$$\text{s.t.} \quad X \in \mathcal{D}_1, \tag{6b}$$

where $\mathcal{D}_1$ is the set of $X \in \mathbb{R}^{|V|\times|\mathcal{C}|}$ satisfying constraints:

$$\sum_{i\in\mathcal{C}} x_{vi} \le c_v \quad \text{for all } v \in V, \tag{7a}$$

$$x_{vi} \in \{0, 1\} \quad \text{for all } v \in V,\ i \in \mathcal{C}. \tag{7b}$$

Compared to classic α-fair utility functions, we add a small constant ϵ for cases $\alpha = 1$ and $\alpha > 1$ to ensure that the objective function G cannot become $-\infty$. Assuming utilities bounded from below is natural in practice. We note that, in contrast to classic α-fairness literature that relies on convexity to solve resource allocation problems (e.g., [7,18]), Problem (6) is combinatorial (it is in fact NP-hard even for $\alpha = 0$ [12]) for a caching network with arbitrary topology.

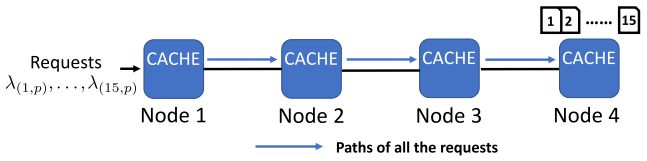
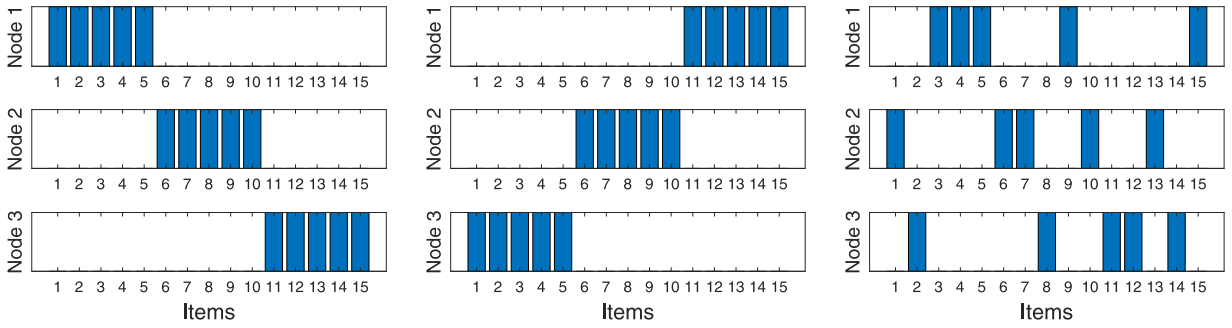


Fig. 2. A path graph with four nodes each equipped with a cache of size 5. All edges have the same cost w. Exogenous requests enter the network from node 1 and are routed to node 4 which is the server for all the items. The request rates for items 1–15 are in decreasing order, i.e., $\lambda_{(1,p)} > \lambda_{(2,p)} > \cdots > \lambda_{(15,p)}$.



(a) $\alpha < 1$ (b) $\alpha > 1$ (c) $\alpha = 1$

Fig. 3. Visualization of the optimal solutions in Example 1. From top to bottom, the three rectangles represent the caches of nodes 1–3. The bars in each rectangle represent the cached items: a bar at position i indicates the node stores item i (e.g., the upper rectangle in Fig. 3(a) shows that node 1 caches the first 5 items). This figure shows that when $\alpha < 1$, it tends to store the most popular items closer to the edge, however there is a full reversal behavior when $\alpha > 1$.

3.3. Two simple examples

Before we study algorithms for solving Problem (6) over arbitrary topologies, we start with a simple caching network with a linear topology, similar to the linear network considered in Chapter 8.5 of Kelly and Yudovina [49]. This gives some intuition on how α-fairness over caching gain rates affects the optimal content allocation. We use two examples over this linear topology: one in which request paths are common but demands for items are heterogeneous, and one where demands are identical but paths are heterogeneous.

3.3.1. Example 1: Common path, diverse rates

We consider the network represented by a path graph shown in Fig. 2 and a content item catalog $\mathcal{C}$ of size 15. Node 4 is the designated server node for all items in $\mathcal{C}$. Requests are generated from node 1, and request rates for content items 1 to 15 are in strictly decreasing order, i.e.,

$$\lambda_{(1,p)} > \lambda_{(2,p)} > \cdots > \lambda_{(15,p)}.$$

Note that, in this example, all requests follow a common path, while the request rates are different.

We divide items into three equal-sized groups: group 1 (items $1,\ldots,5$), group 2 (items $6,\ldots,10$) and group 3 (items $11,\ldots,15$). The following theorem characterizes the optimal solution w.r.t. these groups:

Theorem 1. *The optimal solution for Example* 1 *is as follows:*

1. *When* $0 \leq \alpha < 1$, *the optimal allocation is to place items in group* 1 *at node* 1, *items in group* 2 *at node* 2 *and items in group* 3 *at node* 3 *(Fig.* 3(a)*);*
2. *When* $\alpha > 1$, *the optimal allocation is to place items in group* 1 *at node* 3, *items in group* 2 *at node* 2 *and items in group* 3 *at node* 1 *(Fig.* 3(b)*);*
3. *When* $\alpha = 1$ *and* $\epsilon = 0$, *a solution is optimal if and only if all caches are fully utilized, and no content item is stored in more than one cache (Fig.* 3(c)*).*

Proof. Please see Appendix A. ■

Theorem 1 gives us the following intuition on how optimal caching decisions depend on α. When $\alpha < 1$, items with high request rates/response throughput are prioritized, and hence are placed closer to the source of the requests (the "edge" of the network). When $\alpha > 1$, there is a full reversal of behavior: items with *low request rate/response throughput*

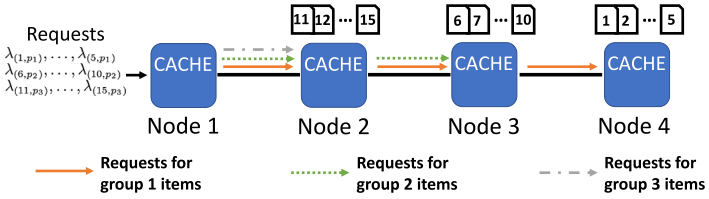
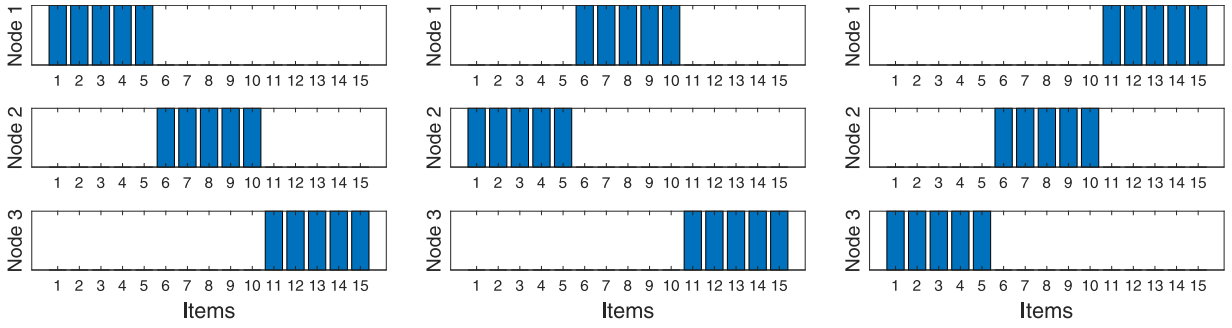


Fig. 4. The same path graph as Fig. 2. Node 4 is the server for group 1 items, node 3 is the server for group 2 items, and node 2 is the server for group 3 items. Exogenous requests enter the network from node 1 with a same request rate but are routed over different paths.



(a) $\alpha = 0$ (b) $0.1 \leq \alpha \leq 0.4$ (c) $\alpha \geq 0.5$

Fig. 5. Visualization of the optimal solutions in Example 2. When $\alpha = 0$, node 1 (edge node) stores items $1, \ldots, 5$ for which the server node is the furthest from the requests source. Node 2 stores items $6, \ldots, 10$ and node 3 stores items $11, \ldots, 15$. When $0.1 \leq \alpha \leq 0.4$, node 1 and node 2 exchange the items they store. And when $\alpha \geq 0.5$, there is a full reversal result of the case when $\alpha = 0$.

are brought closer to the edge; intuitively, their caching gain compensates for the lower throughput under which they are served. Finally, when $\alpha = 1$ and the utility is logarithmic, caching gain and rates are decoupled in the objective, and the optimal allocation does not depend on the request rates.

3.3.2. Example 2: Common rate, diverse paths

We now consider the path graph given in Fig. 4, and again group items in $\mathcal{C}$ into three groups as in Example 1. In this example, we assume that node 4 is the server node for items in group 1, node 3 is the server node for items in group 2, and node 2 is the server node for items in group 3. Requests for all items are generated from node 1 with the same rate. In this setting, the request rate for each item is the same, while paths differ across groups. We find the optimal solutions for $\alpha \in \{0, 0.1, \ldots, 1.9, 2\}$ via exhaustive search, and present them in Fig. 5.

When $\alpha = 0$, under the optimal allocation, items that have to traverse the longest distance (group 1) are prioritized, and placed closer to the edge. For w the weight of an edge, requests for items in group 1 have a caching gain $3 \cdot w$, requests for group 2 have a caching gain w, and requests for group 3 have zero caching gain. When $0.1 \leq \alpha \leq 0.4$, the distribution of caching gain becomes more equitable: the caching gains of requests for items in the three groups are $2 \cdot w$, $2 \cdot w$ and 0, respectively. Finally, when $\alpha \geq 0.5$, all groups receive exactly the same treatment, being cached exactly one hop away from their designated servers: the caching gains for all requests are exactly w. We see that as α increases, the caching allocation for requests in different groups becomes more fair w.r.t. the caching gain (and caching gain rate, as the request rates are same), switching from prioritizing caching decisions that yield large caching gain benefits to more equitable distribution of caching resources.

4. Deterministic offline strategy

In a general network over an arbitrary topology, demands and paths may vary simultaneously. Hence, the overall effect of fairness in an arbitrary caching network would depend on a combination of the phenomena we observed from the two simple examples we studied so far. Because of the NP-hardness of the general problem, we turn our attention to approximation algorithms. In this section, we first show that in a caching network with arbitrary topology, Problem (6) amounts to a submodular set function maximization subject to matroid constraints. Using this structure, we find a solution with an approximation guarantee via the so-called greedy [50] and continuous-greedy [44] algorithms. These algorithms are classic, but we review them here for completeness; some care needs to be taken to handle the possible negativity of the objective (see Theorem 3). In the next section, we discuss a novel method that improves on these guarantees in a stationary randomized setting.

Algorithm 1: Greedy

Input: $G : V \times \mathcal{C} \to \mathbb{R}_+$
$S \leftarrow \emptyset$
while $S^+ := \{(v, i) \in V \times \mathcal{C} : S \cup \{(v, i)\} \in \mathcal{D}_1\} \neq \emptyset$ **do**
  $(v^*, i^*) \leftarrow \arg\max_{(v,i)\in S^+} (G(S \cup \{(v, i)) - G(S))$
  $S \leftarrow S \cup \{(v^*, i^*)\}$
end
return S

4.1. Submodular maximization under matroid constraints

The objective G can be naturally expressed as a set function. For $S \subseteq V \times \mathcal{C}$, let $X_S \in \{0, 1\}^{|V|\times|\mathcal{C}|}$ be the binary vector whose support is S. Since there is a one-to-one correspondence between a binary vector X and its support $\text{supp}(X)$, we can interpret our objective $G : \{0, 1\}^{|V\|\mathcal{C}|} \to \mathbb{R}_+$ as a set function $G : V \times \mathcal{C} \to \mathbb{R}_+$ via $G(S) \triangleq G(X_S)$. We show the monotonicity and submodularity[1] of set function G in the following theorem.

Theorem 2. *The objective function $G(S) \triangleq G(X_S)$ of Problem (6) is a non-decreasing and submodular set function.*

Proof. Please see Appendix B. ■

Constraints (7) define a matroid[2] [6,44]. Hence, Problem (6) is a submodular maximization problem under matroid constraints. This problem is NP-hard in general; we discuss polynomial-time approximation algorithms below.

4.2. Greedy algorithm

The greedy algorithm [50], summarized in Algorithm 1, produces a solution within 1/2 approximation factor from the optimal [43]. Starting from $X = \mathbf{0}$, the greedy algorithm proceeds iteratively, adding one item in a cache at a time, maximizing the marginal improvement in the objective with each iteration. We can further improve the approximation guarantee to $1 - 1/e \approx 0.63$, using the *continuous-greedy* algorithm by Calinescu et al. [44]. We describe this algorithm in more detail in the next section.

4.3. Continuous-greedy algorithm

The algorithm maximizes a relaxation of the objective $G(X)$ over the reals, obtaining a fractional solution Y in the convex hull of $\mathcal{D}_1$. The fractional solution is then rounded to produce a solution to the original combinatorial problem. The relaxation is the *multilinear extension*; we introduce it below.

4.3.1. Multilinear extension

Suppose that x_{vi}, $v \in V$, $i \in \mathcal{C}$, are independent Bernoulli random variables with joint distribution μ defined over $\{0, 1\}^{|V|\times|\mathcal{C}|}$. Let y_{vi}, $v \in V$, $i \in \mathcal{C}$, be the marginal probability that v stores i, i.e.,

$$y_{vi} = \mathbf{P}_\mu[x_{vi} = 1] = \mathbb{E}_\mu[x_{vi}]. \tag{8}$$

Denote by $Y = [y_{vi}]_{v\in V, i\in\mathcal{C}} \in \mathbb{R}^{|V|\times|\mathcal{C}|}$ the matrix of the marginal probabilities. The multilinear extension is defined as $K(Y) = \mathbb{E}_\mu[G(X)]$, i.e.,

$$K(Y) = \sum_{X\in\{0,1\}^{|V|\times|\mathcal{C}|}} G(X) \times \prod_{(v,i)\in V\times\mathcal{C}} y_{vi}^{x_{vi}}(1 - y_{vi})^{1-x_{vi}}. \tag{9}$$

4.3.2. Continuous-greedy process

Now consider the following relaxed problem:

$$\text{Maximize:} \quad K(Y) \tag{10a}$$

$$\text{s.t.} \quad Y \in \mathcal{D}_2, \tag{10b}$$

[1] A set function $f : 2^{\mathcal{X}} \to \mathbb{R}$ is monotone (by which we mean non-decreasing) iff $f(A) \leq f(B)$ for all $A \subseteq B$, $A, B \subseteq \mathcal{X}$, and is submodular iff $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq \mathcal{X}$.

[2] A matroid $\mathcal{M} = (\mathcal{X}, \mathcal{I})$ is defined on a finite ground set $\mathcal{X}$, and $\mathcal{I}$ is a collection of independent subsets of $\mathcal{X}$. $\mathcal{I}$ satisfies: (1) if $A \subseteq B$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$; (2) if $A \in \mathcal{I}$, $B \in \mathcal{I}$, and $|B| > |A|$, then $\exists e \in B \setminus A$: $A \cup \{e\} \in \mathcal{I}$.

where $\mathcal{D}_2 = \text{conv}(\mathcal{D}_1)$ is the convex hull of $\mathcal{D}_1$, i.e.,

$$\sum_{i \in \mathcal{C}} y_{vi} \leq c_v \quad \text{for all } v \in V, \tag{11a}$$

$$y_{vi} \in [0, 1] \quad \text{for all } v \in V,\ i \in \mathcal{C}. \tag{11b}$$

Note that Problem (10) is not convex, as $K(X)$ is not concave. Despite the lack of convexity, the continuous-greedy algorithm summarized in Algorithm 2 solves Problem (10) via Frank–Wolfe iterations [51], producing a fractional vector $Y \in \mathcal{D}_2$ as follows: Starting from an initial point $Y_0 = \mathbf{0}$, the algorithm repeats the following steps in each iteration:

$$\boldsymbol{m_k} \in \arg\max_{\boldsymbol{m} \in \mathcal{D}_2} \left\langle \boldsymbol{m}, \widehat{\nabla K(Y_k)} \right\rangle \tag{12a}$$

$$Y_{k+1} = Y_k + \gamma_k \boldsymbol{m}_k \tag{12b}$$

where $\gamma_k \in [0, 1]$ is an appropriately chosen step size, and $\widehat{\nabla K(Y_k)}$ is an estimation of the gradient $\nabla K(Y_k)$.

An Estimator of the Gradient. Since the objective function $K(Y)$ in (9) comprises a summation of $2^{|V| \times |\mathcal{C}|}$ terms, the computation of gradient $\nabla K(Y_k)$ is challenging. A sampling-based estimator is typically used instead (see [44]). Since K is linear when restricted to each coordinate y_{vi}, for $v \in V$, $i \in \mathcal{C}$, the partial derivative of K w.r.t. y_{vi} can be written as [13]:

$$\frac{\partial K(Y)}{\partial y_{vi}} = E_Y[G(X)|x_{vi} = 1] - E_Y[G(X)|x_{vi} = 0]. \tag{13}$$

To estimate the gradient, one can generate T random samples $X^{(l)}$, $l = 1, \ldots, T$ of the random vector X according to marginal probability matrix Y, and for each pair $(v, i) \in V \times \mathcal{C}$, compute the average

$$\widehat{\frac{\partial K(Y)}{\partial y_{vi}}} = \frac{1}{T} \sum_{l=1}^{T} (G([X^{(l)}]_{+(v,i)}) - G([X^{(l)}]_{-(v,i)})), \tag{14}$$

where $[X]_{+(v,i)}$ and $[X]_{-(v,i)}$ represent vector X with coordinate (v, i) being set to 1 and 0, respectively. The continuous-greedy algorithm (12) ensures that the output solution Y_{out} satisfies constraint (10b) as the convex combination of $\boldsymbol{m_k} \in \mathcal{D}_2$. Y_{out} is also guaranteed to be within $1 - 1/e$ factor from the optimal solution $Y^* \in \mathcal{D}_2$ of Problem (10). We have the following lemma:

Lemma 1 (*Lemma 3.3 in [44]*). *If the discretization step γ_k is small enough, with high probability,*[3] *the fractional solution $Y_{out} \in \mathcal{D}_2$ found by continuous-greedy algorithm satisfies*

$$K(Y_{out}) \geq (1 - \frac{1}{e}) \cdot K(Y^*). \tag{15}$$

Pipage Rounding. The fractional solution Y_{out} of Algorithm 2 can then be rounded to produce an integer solution for Problem (6) with the same guarantee $1 - 1/e$. A rounding method called *pipage rounding* [21] can be used to get an integer solution $\hat{X} \in \mathcal{D}_1$ from $Y_{out} \in \mathcal{D}_2$, ensuring $K(\hat{X}) \geq K(Y_{out})$. We review pipage rounding in Appendix C.

The resulting solution is guaranteed to be within a factor of $(1 - 1/e)$ from the optimal; this result is due to Calinescu et al. (see Theorem 1.1 in [44]). Applied to our setting, this yields the following result:

Theorem 3. *If $\hat{X} \in \mathcal{D}_1$ is the integer solution produced by pipage rounding and $X^* \in \mathcal{D}_1$ is the optimal solution of Problem (6). then with high probability we have:*

$$G(\hat{X}) \geq (1 - \frac{1}{e}) G(X^*), \ \textit{for } 0 \leq \alpha < 1, \tag{16a}$$

$$G(\hat{X}) - G(\mathbf{0}) \geq (1 - \frac{1}{e})(G(X^*) - G(\mathbf{0})), \ \textit{for } \alpha \geq 1. \tag{16b}$$

Proof. Please see Appendix D. ■

We note that continuous-greedy algorithm proceeds in the same way for all $\alpha \in [0, \infty)$. The bounds (16a) and (16b) differ because the objective becomes negative for $\alpha \geq 1$, and shifting the objective function is necessary to derive an approximation ratio.

Time Complexity. To make sure Lemma 1 holds, the number of samples generated to estimate the gradient is $O((|V \parallel \mathcal{C}|)^2 \ln(|V \parallel \mathcal{C}|))$ [13]. Given the estimated gradient, (12) is a linear programming which can be solved in $O(|V \parallel \mathcal{C}|)$ time. The continuous-greedy algorithm finishes in $O(|V \parallel \mathcal{C}|)$ iterations [13]. Finally, the pipage rounding algorithm finishes in at most $|V \parallel C|$ iterations. Thus, the whole method runs in polynomial time in the number of nodes $|V|$ and the number of content items $|I|$.

[3] The term "with high probability" means here with probability at least $(1 - 1/\text{poly}(n))$, where $n = |V| \times |C|$.

Algorithm 2: Continuous-Greedy

Input: $K : \mathcal{D}_2 \to \mathbb{R}_+$, step size $\gamma_k \in (0, 1]$
$t \leftarrow 0$, $k \leftarrow 0$, $Y_0 \leftarrow 0$
while $t < 1$ **do**
  $\boldsymbol{m_k} \leftarrow \arg\max_{\boldsymbol{m} \in \mathcal{D}_2} \left\langle \boldsymbol{m}, \widehat{\nabla K(Y_k)} \right\rangle$
  $\gamma_k \leftarrow \min\{\gamma_k, 1 - t\}$
  $Y_{k+1} \leftarrow Y_k + \gamma_k \boldsymbol{m}_k$, $t \leftarrow t + \gamma_k$, $k \leftarrow k + 1$
end
$Y_{out} \leftarrow Y_k$
return Y_{out}

5. Stationary randomized strategy

In the previous section, we consider a deterministic offline setting: we decide and implement the content item placement at the beginning and study the total utility in the network. In this section, we consider a different setting; in particular, we focus on a time-slotted system, and do not make a deterministic content item allocation. Instead, in each time slot, each node independently samples a new content placement according to a stationary distribution and reshuffles the content items in its cache. We show that, in this setting, we can attain improved performance guarantees over the continuous greedy algorithm.

5.1. Total utility of expected caching gain rate

Formally, we assume that time is slotted, and that at each time slot $t \in \mathbb{N}$, a random caching strategy X is sampled from a joint distribution μ over $\mathcal{D}_1$. We assume that μ has a product form. Specifically, for every node v, there is a distribution μ_v, $v \in V$, such that:

$$\mu(X) = \prod_{v \in V} \mu_v(x_{v1}, \ldots, x_{v|\mathcal{C}|}). \tag{17}$$

Assumption (17) allows nodes to make independent caching decisions, which avoids synchronization among nodes. On the other hand, reshuffling contents incurs additional overhead; however, if the duration of each timeslot is large, this overhead can be considered negligible compared to the routing costs incurred while serving requests during a timeslot.

In contrast to the previous setting, we aim to decide the optimal μ to maximize the total utility of expected caching gain rate (UECGR) of the network, which is defined as:

$$\text{UECGR:} \sum_{(i,p) \in \mathcal{R}} U(\mathbb{E}_\mu[\lambda_{(i,p)} F_{(i,p)}(X)]). \tag{18}$$

Recall that μ is a distribution over feasible allocations, i.e., μ's support is a subset of $\mathcal{D}_1$. We denote by y_{vi}, $v \in V$, $i \in \mathcal{C}$ the marginal probability that node v stores item i, i.e.,

$$\mathbb{E}_{\mu_v}[x_{vi}] = \mathbf{P}_{\mu_v}[x_{vi}] = y_{vi}, \text{ for } v \in V, \ i \in \mathcal{C}. \tag{19}$$

Let $Y = [y_{vi}]_{v \in V, i \in \mathcal{C}}$ be the matrix of marginals y_{vi}. Then, $Y \in \mathcal{D}_2$, where $D_2 = \text{conv}(\mathcal{D}_1)$ is the convex hull of $\mathcal{D}_1$, given by (11). To compute the expectation $\mathbb{E}_\mu[\lambda_{(i,p)} F_{(i,p)}(X)]$ it suffices to evaluate $\lambda_{(i,p)} F_{(i,p)}$ over Y. Indeed, taking the expectation of $F_{(i,p)}$, we have

$$\mathbb{E}_\mu[F_{(i,p)}(X)] = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \mathbb{E}_\mu\left[\prod_{k'=1}^{k} (1 - x_{p_{k'}i})\right]\right) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - \mathbb{E}_\mu[x_{p_{k'}i}])\right) = F_{(i,p)}(Y). \tag{20}$$

Note that the second equality holds by the independence of x_{vi}, for all $v \in V$, and the fact that there is no repeating node along path p.

Maximizing the total utility of expected caching gain rate (18) w.r.t. μ over $\mathcal{D}_1$ is therefore equivalent to the following problem:

$$\text{Maximize:} \quad G(Y) = \sum_{(i,p) \in \mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(Y)) \tag{21a}$$

$$\text{s.t.} \quad Y \in \mathcal{D}_2, \tag{21b}$$

where $\mathcal{D}_2$ is the set of matrices $Y = [y_{vi}]_{v \in V, i \in \mathcal{C}} \in \mathbb{R}^{|V| \times |\mathcal{C}|}$, satisfying constraint (11). We note that the equivalence of the two problems is due to (20) and the fact that there exists an algorithm that can be used to construct μ from marginals $Y \in \mathcal{D}_2$ (see, e.g., [12,52]).

We therefore turn our attention to solving Problem (21). We propose a novel method, which we call the *L-method*, that solves Problem (21) and produces a solution $Y \in \mathcal{D}_2$ within a $(1-1/e)^{1-\alpha}$ factor from the optimal deterministic solution of Problem (6). In doing so, we extend the method by Ageev and Sviridenko [21] used earlier in the linear case ($\alpha = 0$) [6,12]. When $0 < \alpha \leq 1$, this factor is better than the $1-1/e$ ratio achieved by the continuous-greedy algorithm.

5.2. L-method

Problem (21) is not a convex optimization problem as it maximizes a non-concave function (21a). However, (21a) can be approximated as follows using the *L-function*. We define $L_{(i,p)} : \mathcal{D}_2 \rightarrow \mathbb{R}_+$:

$$L_{(i,p)}(Y) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\left\{1, \sum_{k'=1}^{k} y_{p_{k'}i}\right\}. \tag{22}$$

Note that $L_{(i,p)}$ is non-decreasing and concave as a pointwise minimum of linear functions. Consider now the problem:

$$\text{Maximize:} \quad H(Y) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} L_{(i,p)}(Y)) \tag{23a}$$

$$\text{s.t.} \quad Y \in \mathcal{D}_2. \tag{23b}$$

Since $L_{(i,p)}$ is non-decreasing and concave, and the utility function U is a concave function, the composite function $U(\lambda_{(i,p)}L_{(i,p)}(Y))$ is concave, and the objective function $H(Y)$ in (23a) is also concave. Thus, Problem (23) is a convex optimization problem. For any $Y \in \mathcal{D}_2$ and any $g \in \partial H(Y)$, g is bounded. Hence, by the definition of the subgradient and the Cauchy–Schwarz inequality, H is Lipschitz continuous. Thus, projected subgradient ascent finds an η-optimal point of Problem (23) in $\Theta(1/\eta^2)$ iterations (see Theorem 3.2 of [53]).

Our proposed L-method finds marginals Y by solving the convex optimization problem (23). This solution is guaranteed to be within a constant factor from the optimal solution of Problem (21), as well as from Problem (6):

Theorem 4. *Let X^*, Y^* and Y^{**} be optimal solutions to Problems (6), (21) and (23) respectively. Then for $\alpha \neq 1$*

$$G(Y^{**}) \geq (1-\frac{1}{e})^{1-\alpha} G(Y^*) \geq (1-\frac{1}{e})^{1-\alpha} G(X^*), \tag{24}$$

while for $\alpha = 1$,

$$G(Y^{**}) \geq G(Y^*) - c \geq G(X^*) - c, \tag{25}$$

where $c = |\mathcal{R}| \log \frac{e}{e-1}$.

Proof. Please see Appendix E. ■

We note that the above bounds are sharper than the bound attained by continuous greedy in Theorem 3 for $0 < \alpha \leq 1$. The L-method therefore attains better performance under the stationary randomized strategy regime.

Randomized Rounding. Given the marginal probability matrix $Y \in \mathcal{D}_2$ produced by the L-method, any suitable randomized rounding policy can be used at each node $v \in V$ to produce a joint distribution μ over $\mathcal{D}_1$ that satisfies (19). An efficient randomized rounding algorithm to generate such a distribution is given in Algorithm 2 of [12] and, independently, in [52].

6. Extensions

In this section, we explore extensions of of our setting. As mentioned in Sections 1 and 2, one important notion is *content fairness*, where storage resources are fairly allocated to content items based on their respective popularities. Another is *user fairness*, where storage resources are fairly allocated to users based on their respective demands. We will show that our analysis applies to all these extensions, as well as to a scenario where contents have unequal sizes.

Content and User Fairness. Suppose each content item i has an associated utility, a function of the total caching gain rate of all requests requesting item i. We denote by $\mathcal{R}_i$, $i \in \mathcal{C}$, the set of all requests that request item i, such that $\cup_{i\in\mathcal{C}}\mathcal{R}_i = \mathcal{R}$. We then have the following objective corresponding to content fairness:

$$G(X) = \sum_{i\in\mathcal{C}} U\left(\sum_{(i,p)\in\mathcal{R}_i} \lambda_{(i,p)} F_{(i,p)}(X)\right). \tag{26}$$

For user fairness, suppose each user v has an associated utility, a function of the total caching gain rate of all requests generated from v. We denote $\mathcal{R}_v$, for $v \in V$, the set of all requests generated from node v, such that $\cup_{v\in V}\mathcal{R}_v = \mathcal{R}$. We

Table 2
Graph topologies and experiment parameters.

Graph	$\|V\|$	$\|E\|$	$\|\mathcal{C}\|$	$\|\mathcal{R}\|$	$\|Q\|$	c_v
hypercube	128	896	300	1K	20	3
balanced-tree	341	680	300	1K	20	3
grid-2d	100	360	300	1K	20	3
erdos-renyi	100	1042	300	1K	20	3
small-world	100	491	300	1K	20	3
barabasi-albert	100	768	300	1K	20	3
geant	22	66	10	100	10	2
abilene	9	26	10	100	4	2
dtelekom	68	546	300	1K	20	3

then have the following objective corresponding to user fairness:

$$G(X) = \sum_{v \in V} U\left(\sum_{(i,p)\in\mathcal{R}_v} \lambda_{(i,p)} F_{(i,p)}(X)\right). \tag{27}$$

By maximizing objective functions (26) and (27) over $X \in \mathcal{D}_1$, we obtain content item allocations that yield content fairness and user fairness, respectively. The algorithms we discussed in the previous two sections still work for both cases, yielding the same approximation ratio. This is because (a) the objective functions are still non-decreasing and submodular and (b) we can still approximate $F_{(i,p)}(X)$ by $L_{(i,p)}(X)$ in the stationary randomized strategy setting, thereby solving a convex optimization problem.

Unequally-sized Contents. We can extend our model to contents with different sizes, as contents can be partitioned into equal sized "chunks". Such "chunks" can be treated as distinct items in our model [54]. To request a content, a user sends simultaneous requests for all chunks/items of this content to the server node, that is assumed to permanently store the content, following the same path. The caching gain can then be expressed as the sum of caching gains per chunk/item.

More formally, using index j for contents, we define $\mathcal{C}_j$ the set of equal-sized chunks/items of content j. We let $\mathcal{R}'$ be the set of all requests for contents (unequally sized). Then, the caching gain rate of a content request $(j, p) \in \mathcal{R}'$ is $\lambda_{(j,p)} \sum_{i\in\mathcal{C}_j} F_{(i,p)}(X)$. To study the request fairness of unequal-sized contents, we can maximize the following objective:

$$G(X) = \sum_{(j,p)\in\mathcal{R}'} U\left(\lambda_{(j,p)} \sum_{i\in\mathcal{C}_j} F_{(i,p)}(X)\right). \tag{28}$$

We can also study content fairness or user fairness by grouping up the requests of same content or generated from the same user, as in (26) and (27). Again, our analysis, algorithms, and guarantees directly extend to these settings.

7. Numerical study

To evaluate the greedy (GRD), continuous-greedy (CG) and L-method (L) algorithms, we perform extensive simulations over a number of synthetic and real networks, and compare their performance to the performance of path replication algorithms combined with the LRU, LFU, FIFO and random replacement (RR) caching strategies. In addition to comparing the algorithms in terms of performance, we also discuss how different notions of fairness and different degrees of fairness (utility functions with different α) influence the caching gains of requests as well as the content allocation in the networks.

7.1. Experimental setting

Network Topologies. The network topologies we consider are summarized in Table 2. We consider four synthetic topologies: 7-dimensional hypercube (HC), balanced-tree (BT) with depth 9, two dimensional grid graph grid-2d, erdos-renyi (ER) which is the Erdős–Rényi with parameter $p = 0.1$, small-world (SW) by [55], and the preferential attachment model of barabasi-albert (BA). We also consider three real backbone networks, geant, abilene and dtelekom (Deutsche Telekom) [3].

Demand. Given a graph $G = (V, E)$ and an item catalog $\mathcal{C}$, we choose a set of designated server nodes S_i uniformly at random (u.a.r.) from V for each item $i \in \mathcal{C}$. In ICNs and CDNs, the majority of the traffic is usually caused by a small subset of content items. Thus, we focus on the caching and delivery of these items. Accordingly, we set a moderate catalog size of 300 for most topologies. Each edge $(u, v) \in E$ is associated with a weight w_{uv} u.a.r. sampled from 1 to 5. We denote by $\mathcal{R}$ the set of all requests and generate it as follows: For $r \in \mathcal{R}$, we choose the source node of r u.a.r. from Q, which is the set of query nodes chosen u.a.r. from V. The item i^r that r requests is selected according to a Zipf distribution with

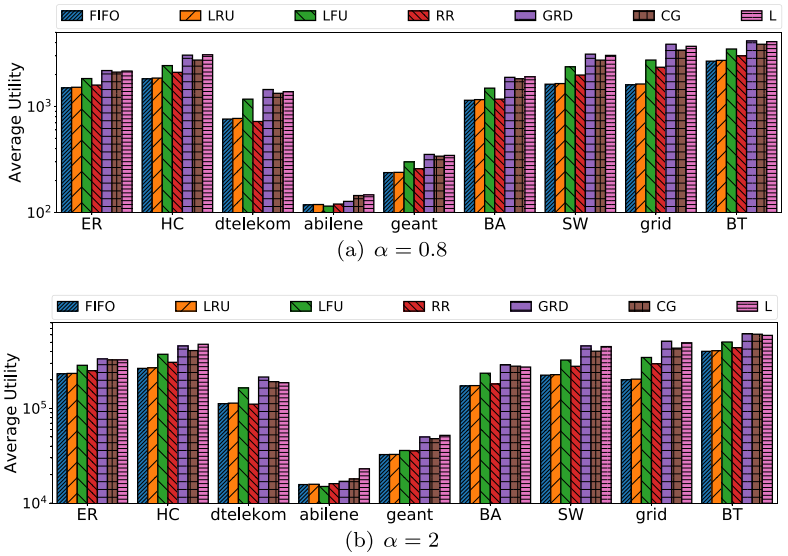


Fig. 6. Average utilities. Note that when $\alpha = 2$, the average utilities are subtracted by their lower bound $G(\mathbf{0})$ (see Theorem 3). L-method, greedy and continuous-greedy algorithms outperform traditional path replication policies in all the topologies.

parameter 1.1. The path p^r is the shortest path between the source node and the server nodes of the requested item. We set $\lambda_r = 1.0$ per time unit for all requests $r \in \mathcal{R}$.

Algorithms. We implement the greedy (`GRD`), continuous-greedy (`CG`) and L-method (`L`) algorithms for each α. We also implement path replication [56] combined with `FIFO`, `LRU`, `LFU`, and random replacement (`RR`) eviction policies. We note that our algorithm for setting $\alpha = 0$ (i.e., linear utility) corresponds to the offline algorithm presented in [12]; as such, $\alpha = 0$ can be treated as an additional baseline for comparison purposes.

Simulation Parameters. The algorithms compared are simulated for 5000 time units in all topologies. To leverage the PASTA property, we collect measurements at epochs of a Poisson process with rate 1.0. At each measurement epoch, we extract the current content allocation X produced by each algorithm and compute the total utility $G(X)$, given by (6a). Finally, we take the average of all the $G(X)$ we obtained during the 5000 time units simulation.

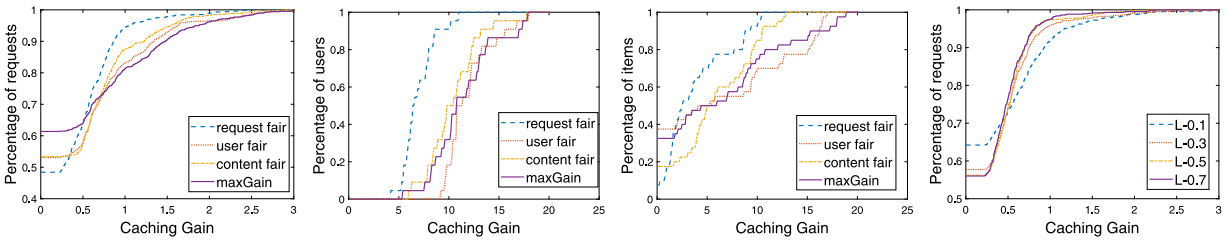
7.2. Results

Algorithm Comparison. In Fig. 6, we plot the time-average total utilities achieved by different algorithms in all nine network topologies for the cases $\alpha = 0.8$ and $\alpha = 2$ when we consider request fairness, i.e., problem (6). We can see that in all cases, the continuous-greedy (`CG`) and L-method (`L`) outperform the four path replication algorithms. Although attaining a lower approximation guarantee, the greedy algorithm is comparable to `CG` and `L` for several topologies, which was also observed in, e.g., [13]. However, it attains suboptimal solutions for `abilene`, where `CG` and `L` perform better.

Fairness Variants. While request fairness aims to achieve fair resource sharing among different requests, content fairness and user fairness aim to achieve fairness w.r.t. content items and users that generate requests, respectively. We study the effect of different notions of fairness by observing the distribution of caching gains, i.e., how they are spread across requests, users and content items.

Figs. 7(a)–7(c) plot the CDFs of caching gains for different notions of fairness across requests, users and content items, respectively, for the `geant` topology. Specifically, we solve problem (6), (26) and (27) for $\alpha = 2$ using the L-method and compute the CDFs. We also use the result derived from maximizing total caching gain rate as a baseline which corresponds to solving problem (6) for $\alpha = 0$. The sharpness of a CDF reflects how concentrated (and, thereby, fair) the caching gains are in this network. The Area Above the Curve (AAC) equals the expected caching gain of requests, users, and contents in Figs. 7(a)–7(c) respectively. In each figure, we scrutinize fairness w.r.t. the sharpness of CDF (sharper is better) as well as the AAC (larger is better).

We make the following observations: First, in Fig. 7(a), showing CDFs across requests, the curve derived by request fairness exhibits the sharpest increase, which indicates there are smaller variations in the caching gains of requests when we consider request fairness. Second, in Figs. 7(b) and 7(c), user fairness and content fairness give the best performance in terms of AAC respectively, which is consistent with the aims of these two fairness notions. The curves corresponding to request fairness are also sharp here; this is not surprising, as request fairness is a more stringent notion (request fairness implies user and item fairness). However, the CDFs of request fairness have smaller AAC, which means request fairness



(a) CDFs across requests (b) CDFs across users (c) CDFs across content items (d) CDFs for different α

Fig. 7. CDFs of caching gains given by different fairness notations. The curve for `maxGain` is generated by maximizing the total caching gain rate (maximizing total utility when $\alpha = 0$). Figs. 7(a)–7(c) show that request fairness, user fairness and content fairness generate the curve with sharpest increase in the CDFs across requests, users and content items, respectively. Fig. 7(d) shows that for a specific fairness notion, we obtain a sharper CDF with a larger α.

Table 3
Objective obtained by different allocations.

Objective	Allocation		
	Request	User	Item
Request	−**5.3e+05**	−5.8e+05	−6.1e+05
User	−3.0126	−**2.1550**	−2.8576
Item	−4.0e+03	−1.5e+04	−**2.0e+03**

results in less total caching gain of users and contents. Third, maximizing the total caching gain rate always results in the least sharp CDFs in Figs. 7(a)–7(c), meaning that it is a poor strategy in terms of fairly allocating storage resources.

Besides intuitively observing the fairness by CDFs via sharpness, we use the objective $G(X)$ for request, user, and item fairness, respectively, as a quantitative metric. To compare different notions of fairness, we can fix an objective under a fairness notion (e.g., request, user, item), and compare that objective attained by content allocations optimizing a different objective. The resulting comparison is shown in Table 3 derived under the `geant` topology and $\alpha = 2$. The table shows the objective under different fairness notions attained by content allocations optimizing different objectives. We can clearly see that values are maximized on the diagonal; note that the result is non-obvious, as we are using approximation algorithms.

Impact of α. We first demonstrate the impact of α on the CDFs of caching gains across requests in Fig. 7(d). The results are shown for request fairness in the `geant` topology, using the L-method. It is clear to see that as α increases, the CDFs become sharper: fewer requests receive very small or very large caching gains, and more requests receive similar caching gains.

As we wish to achieve fairness by fairly allocating the limited cache space in the network, it is also interesting to observe how content allocation in the caches is affected by the degree of fairness. In Section 3.3, we have already shown by two examples that we obtain different content allocations in the network under different degrees of fairness (different α values). We now examine content allocation in a more complex network with more complex balanced tree topology. Different from the balanced tree we consider in Table 2, this is a 4-ary tree of height 3: it has 85 nodes, with 4^ℓ nodes in layer $\ell = 0, \ldots, 3$. The cache capacity of each node is 2. We consider a content catalog with 20 items. Only the leaf nodes (i.e., the "edge") generate requests. For each leaf node, the request rate for each item follows a Zipf distribution with parameter 0.8 (popularities are in decreasing order of the item indices).

We consider request fairness by solving the problem (6) with L-method. Figs. 8(a)–8(f) plot the resulting content allocations with $\alpha = 0$, 0.2, 0.4, 0.6, 0.8 and 1. The horizontal axis of each figure is the content index and the vertical axis is the fractions of the total cache space in the respective layers allocated to the respective content items. In Fig. 8(a), for which $\alpha = 0$, it is shown that nodes in layer 3 cache only items 1 and 2 (the two most popular items), layer 2 nodes cache items 3 and 4, and layer 1 nodes cache items 5 and 6. In other words, when we maximize the overall caching gain and do not consider fairness, the network tends to cache the most popular items close to the edge nodes. As α increases to 0.2, some nodes in layer 1 and layer 2 start to cache other items, though layer 3 nodes still cache only the two most popular items. Layer 3 nodes start to allocate their storage to other content items when α increases to 0.4. When $\alpha = 1$, the items are more evenly distributed in each layer. These observations lead us to the following insight: as we increase the degree of fairness, we no longer only cache items with high popularity closer to the edge, but more fairly allocate the storage resources to all items.

We use the variance of caching gains across requests in the network and the objective when $\alpha = 1$ as two quantitative metrics of fairness. We plot the results with $\alpha \in [0, 1]$ in Fig. 9. We note that it is a prior knowledge that the allocation we derived by setting $\alpha = 1$ is the fairest one, so we use the objective when $\alpha = 1$ as a quantitative metric, and a higher

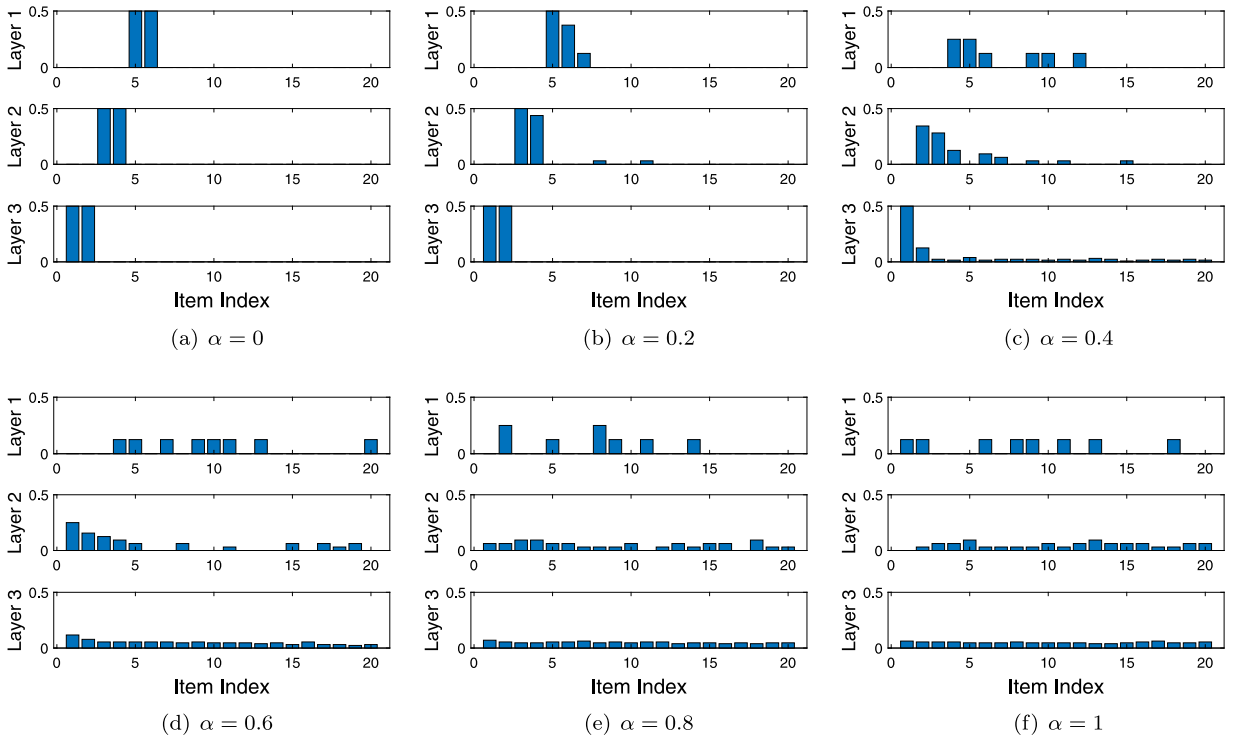


Fig. 8. The content allocation in a balanced tree caching network. A bar at position $i \in \{1, \ldots, 20\}$ represents the fraction of cache space in a layer that is allocated to item i. The items are more evenly distributed in each layer as α increases.

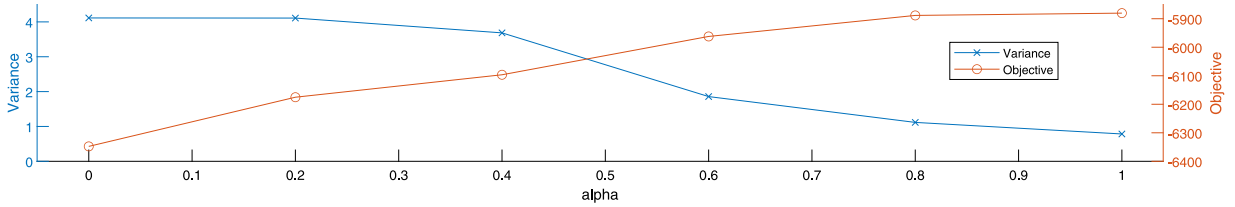


Fig. 9. Quantitative evaluation of fairness with different α in a balanced tree caching network. It plots the variance of caching gains of requests in the network, and the objective when $\alpha = 1$, derived by considering request fairness with $\alpha \in [0, 1]$. As α increases, the variance of caching gains decreases and the objective increases.

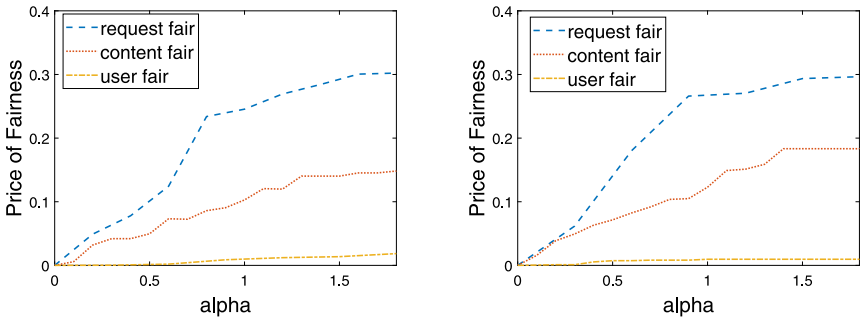
value corresponds to a fairer allocation. As expected, the results show that has α increases, the variance of caching gains decreases and the objective value increases; both indicate the emergence of a fairer allocation.

Price of Fairness. Fairly allocating the network resources by maximizing the aggregated global utility could cause a degradation in the aggregated global caching gain. To characterize this degradation, we compute the Price of Fairness (PoF) for different α. The PoF is defined as the relative reduction in the total caching gain when we consider fairness ($\alpha > 0$), compared to the total caching gain when we do not consider fairness ($\alpha = 0$) [57]. A higher PoF indicates a larger sacrifice w.r.t. the global caching gain.

Figs. 10(a) and 10(b) plot the PoF results of different fairness notions derived with `geant` topology and `grid-2d` topology (see Table 2). The PoF increases as the degree of fairness increases (α increases) for all fairness notions. Request fairness produces a higher PoF which is around 30% when α is greater than 1.5. As the number of content items is much higher than the number of request nodes or users, the PoF of content fairness is higher than the PoF of user fairness. When α is smaller than 0.4, all fairness notions produce a PoF within 10% in both topologies.

8. Conclusion

We address the problem of achieving different degrees of fairness for requests, content items and users in caching networks. To the best of our knowledge, this is the first work that models these three fairness notions (requests, content,



(a) `geant` topology (b) `grid-2d` topology

Fig. 10. Price of Fairness (PoF) of different fairness notions. The PoF is a value in [0, 1] which represents the relative caching gain reduction in the network.

users/nodes) over the caching gain rate of a multi-hop caching network with arbitrary topology. We show that the greedy, continuous-greedy and L-method algorithms yield theoretical optimality guarantees as well as desirable performance in numerical experiments, when compared to baseline algorithms.

Our utility-based framework enables network operators and service providers to achieve a balance between minimizing routing costs and assigning fair caching allocations. We minimize the total routing cost by solving the problem with $\alpha = 0$, and place more emphasis on realizing fair content allocation with larger α. With this framework, network operators and service providers can determine the content allocation by setting an appropriate value of α, based on the requirements of global caching gain and fairness.

As this work mainly focuses on fair resource allocation along a path, the routing can be done by any policy, and we use the shortest path in the simulations. There is a possibility to extend our work to include both routing and caching optimization, as, e.g., in [14]. Jointly optimizing caching decisions and request admission rates is also a possible direction for future research. In this scenario, one would aim to decide the request rates which are allowed to enter the network, and jointly study the fair allocation of both storage and bandwidth resources. This would pose a computational challenge, as the resulting mixed integer problem would not be amenable to analysis using a submodularity argument.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors express their gratitude to the reviewers and the editor for their thorough and insightful feedback, which significantly improved the paper.

Funding

This work was supported by the National Science Foundation [Grant No. 1718355], research grants by Intel Corp. and Cisco Systems, and the Natural Science Foundation of China [Grant No. 62002399].

Appendix A. Proof of Theorem 1

Proof. In Example 1, all requests are generated at node 1 and routed to node 4. Consider the requests for two items i and j with request rate $\lambda_i \geq \lambda_j$, and two nodes to store these two items which induce caching gains $F_u \geq F_w$. When $0 \leq \alpha < 1$ the following inequality always holds,

$$U(\lambda_i F_u) + U(\lambda_j F_w) \geq U(\lambda_i F_w) + U(\lambda_j F_u),$$

which implies that, when $0 \leq \alpha < 1$, it is better to assign higher caching gain to requests with higher request rates for higher total utility, and any allocation differing from Fig. 3(a) is suboptimal. When $\alpha > 1$, we have a reverse result,

$$U(\lambda_i F_u) + U(\lambda_j F_w) \leq U(\lambda_i F_w) + U(\lambda_j F_u).$$

Then any allocation differing from Fig. 3(b) is suboptimal. Finally, when $\alpha = 1$ and $\epsilon = 0$,

$$U(\lambda_i F_u) + U(\lambda_j F_w) = U(\lambda_i F_w) + U(\lambda_j F_u),$$

due to the log utility function. We thus can exchange the positions of two cached items without changing the total utility. Any allocation in which items are not cached repeatedly is optimal as the total cache size equals the number of items. □

Appendix B. Proof of Theorem 2

Proof. To begin with, we note that the caching gain $F_{(i,p)}$ of request $(i,p) \in \mathcal{R}$ is non-decreasing and submodular [12]. Intuitively, when we cache more content items, the cost of a request is non-increasing, thus the caching gain is non-decreasing. Furthermore, storing one more item in the network induces more marginal caching gain increment when less items have already been cached.

To prove that the set function G in (6a) is also non-decreasing and submodular, we first introduce and prove the following lemma:

Lemma 2. *Let $f : \mathbb{R} \to \mathbb{R}$ be a non-decreasing concave function and let $g : \mathcal{X} \to \mathbb{R}$ be a non-decreasing submodular set function. Then $h(x) \triangleq f(g(x))$ is also submodular and non-decreasing.*

Proof. Since g is non-decreasing, for any $x, x' \subseteq \mathcal{X}$ we have

$$g(x \cap x') \leq g(x) \leq g(x \cup x'),$$
$$g(x \cap x') \leq g(x') \leq g(x \cup x').$$

So that we can find $\alpha, \alpha' \in [0,1]$ such that

$$g(x) = (1-\alpha)g(x \cap x') + \alpha g(x \cup x'),$$
$$g(x') = (1-\alpha')g(x \cap x') + \alpha' g(x \cup x').$$

Due to submodularity of g

$$\begin{aligned} g(x) + g(x') &= g(x \cap x') + g(x \cup x') + (1 - \alpha - \alpha')(g(x \cap x') - g(x \cup x')) \\ &\geq g(x \cap x') + g(x \cup x'), \end{aligned}$$

which implies that $\alpha + \alpha' \geq 1$. Then we have

$$\begin{aligned} f(g(x)) + f(g(x')) &\geq (1-\alpha)f(g(x \cap x')) + \alpha f(g(x \cup x')) + (1-\alpha')f(g(x \cap x')) + \alpha' f(g(x \cup x')) \\ &= f(g(x \cap x')) + f(g(x \cup x')) + (1 - \alpha - \alpha')(f(g(x \cap x')) - f(g(x \cup x'))) \\ &\geq f(g(x \cap x')) + f(g(x \cup x')), \end{aligned}$$

where the first inequality is due to concavity of f, and the second one is because $\alpha + \alpha' \geq 1$ and $f(g(\cdot))$ is non-decreasing. □

Since the utility function $U(\cdot)$ is non-decreasing and concave, $U(\lambda_{(i,p)} F_{(i,p)}(X))$ is a non-decreasing submodular set function for all $(i,p) \in \mathcal{R}$. Hence, as a linear combination of such functions, the objective function $G(X)$ in (6a) is non-decreasing and submodular. □

Appendix C. Pipage rounding

Pipage rounding is a simple deterministic rounding method utilizing the *ϵ-convexity* property [21] of the objective function. Suppose we have a fractional solution y that maximizes a function over a polytope, i.e., $\max\{F(y) : y \in \mathrm{P}(\mathcal{M})\}$ where $\mathrm{P}(\mathcal{M})$ is the matroid polytope of $\mathcal{M} = (\mathcal{X}, \mathcal{I})$. Pipage rounding aims to convert y into an integral solution, corresponding to a vertex of polytope $\mathrm{P}(\mathcal{M})$, without decreasing the objective function F. In particular, in each iteration, we move a point $y \in \mathrm{P}(\mathcal{M})$ in a direction $\mathbf{e}_i - \mathbf{e}_j$ or $\mathbf{e}_j - \mathbf{e}_i$, where $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is a canonical basis vector.

The crucial property (ϵ-convexity) of $F(y)$ that makes this procedure work is the following: for any $y \in [0,1]^{\mathcal{X}}$ and $i, j \in X$, the function $F_{ij}^{y}(t) = F(y + t(\mathbf{e}_i - \mathbf{e}_j))$ is convex. It allows us to choose one of two possible directions, $\mathbf{e}_i - \mathbf{e}_j$ or $\mathbf{e}_j - \mathbf{e}_i$, in each step, so that the value of $F(y)$ does not decrease.

The multilinear extension of a monotone submodular function has this ϵ-convexity property (see Section 2.4 [44]). Apply pipage rounding to our setting: given a fractional solution $Y \in \mathcal{D}_2$, there are at least two fractional variables y_{vi} and $y_{vi'}$, such that transferring mass from one to the other, *(1)* makes at least one of them integer, *(2)* produces a new solution $\hat{Y} \in \mathcal{D}_2$, and *(3)* ensures $K(\hat{Y}) \geq K(Y)$. We repeat this process until it produces an integer solution $\hat{X} \in \mathcal{D}_1$, which satisfies $K(\hat{X}) \geq K(Y)$.

Appendix D. Proof of Theorem 3

Proof. We first focus on the case when $\alpha < 1$. Suppose X^* and Y^* are optimal solutions of Problem (6) and Problem (10), and let $\hat{X}$ be the rounded integer solution we get from fractional solution Y_{out} produced by Algorithm 2. If the step size γ_k of the continuous greedy algorithm is small enough, with high probability, we have the following expression for $0 \leq \alpha < 1$:

$$G(\hat{X}) = K(\hat{X}) \geq K(Y_{out}) \overset{(15)}{\geq} (1 - \frac{1}{e})K(Y^*) \geq (1 - \frac{1}{e})K(X^*) = (1 - \frac{1}{e})G(X^*).$$

The first inequality is guaranteed by pipage rounding method. Moreover, inequality $K(Y^*) \geq K(X^*)$ is ensured by the optimality of Y^*. Note that, the two equalities hold, because G is equal to its expectation when there is no randomness of the variables.

For $\alpha \geq 1$, by Theorem 2, the objective function G is still non-decreasing and submodular, but it can assume negative values. Note that the proof of Lemma 1 applies only to positive objective functions [44]. Thus, to provide an optimality factor, we consider $\hat{G}(\cdot) = G(\cdot) - G(\mathbf{0})$ instead, and above analysis still holds. □

Appendix E. Proof of Theorem 4

Proof. To begin with, the second inequality holds, as Problem (21) maximizes the same objective function over a larger domain. To prove the first inequality, we use the following bound by Ioannidis and Yeh [12]; for all $Y \in \mathcal{D}_2$,

$$L_{(i,p)}(Y) \geq F_{(i,p)}(Y) \geq (1 - \frac{1}{e})L_{(i,p)}(Y).$$

Since U is a non-decreasing function and $\lambda_{(i,p)}$ is a non-negative constant, for $\alpha \neq 1$, we have

$$U(\lambda_{(i,p)}L_{(i,p)}(Y)) \geq U(\lambda_{(i,p)}F_{(i,p)}(Y)) \geq (1 - \frac{1}{e})^{1-\alpha}\ U(\lambda_{(i,p)}L_{(i,p)}(Y)). \tag{E.1}$$

Summing (E.1) over $(i, p) \in \mathcal{R}$,

$$H(Y) \geq G(Y) \geq (1 - \frac{1}{e})^{1-\alpha}\ H(Y).$$

By the optimality of Y^{**},

$$(1 - \frac{1}{e})^{\alpha-1}G(Y^{**}) \geq H(Y^{**}) \geq H(Y^*) \geq G(Y^*),$$

which proves the first inequality.

For $\alpha = 1$, due to the logarithmic utility function we have

$$U(\lambda_{(i,p)}L_{(i,p)}(Y)) \geq U(\lambda_{(i,p)}F_{(i,p)}(Y)) \geq U(\lambda_{(i,p)}L_{(i,p)}(Y)) + \log(1 - \frac{1}{e}),$$

and

$$H(Y) \geq G(Y) \geq H(Y) - c,$$

where $c = |\mathcal{R}| \cdot \log(\frac{e}{e-1})$ is a positive constant. Then, we have

$$G(Y^{**}) \geq H(Y^{**}) - c \geq H(Y^*) - c \geq G(Y^*) - c \geq G(X^*) - c. \quad \square$$

References

[1] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard, Networking named content, in: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, ACM, 2009, pp. 1–12, http://dx.doi.org/10.1145/1658939.1658941.

[2] E.J. Rosensweig, D.S. Menasche, J. Kurose, On the steady-state of cache networks, in: IEEE Conference on Computer Communications, INFOCOM 2013, 2013, pp. 863–871, http://dx.doi.org/10.1109/INFCOM.2013.6566874.

[3] D. Rossi, G. Rossini, Caching Performance of Content Centric Networks under Multi-Path Routing (and More), Technical Report, Relatório técnico, Telecom ParisTech, 2011.

[4] S. Borst, V. Gupta, A. Walid, Distributed caching algorithms for content distribution networks, in: IEEE Conference on Computer Communications, INFOCOM 2010, 2010, pp. 1–9, http://dx.doi.org/10.1109/INFCOM.2010.5461964.

[5] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, R. Sitaraman, On the complexity of optimal routing and content caching in heterogeneous networks, in: IEEE Conference on Computer Communications, INFOCOM 2015, 2015, pp. 936–944, http://dx.doi.org/10.1109/INFOCOM.2015.7218465.

[6] K. Shanmugam, N. Golrezaei, A.G. Dimakis, A.F. Molisch, G. Caire, Femtocaching: Wireless content delivery through distributed caching helpers, IEEE Trans. Inform. Theory 59 (12) (2013) 8402–8413, http://dx.doi.org/10.1109/TIT.2013.2281606.

[7] M. Dehghan, L. Massoulie, D. Towsley, D.S. Menasche, Y.C. Tay, A utility optimization approach to network cache design, IEEE/ACM Trans. Netw. 27 (3) (2019) 1013–1027, http://dx.doi.org/10.1109/TNET.2019.2913677.

[8] W. Chu, M. Dehghan, J.C. Lui, D. Towsley, Z. Zhang, Joint cache resource allocation and request routing for in-network caching services, Comput. Netw. 131 (2018) 1–14, http://dx.doi.org/10.1016/j.comnet.2017.11.009.

[9] M. Dehghan, W. Chu, P. Nain, D. Towsley, Z. Zhang, Sharing cache resources among content providers: A utility-based approach, IEEE/ACM Trans. Netw. 27 (2) (2019) 477–490, http://dx.doi.org/10.1109/TNET.2018.2890512.

[10] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, D. Leong, Vip: A framework for joint dynamic forwarding and caching in named data networks, in: Proceedings of the 1st ACM Conference on Information-Centric Networking, ICN, ACM, 2014, pp. 117–126, http://dx.doi.org/10.1145/2660129.2660151.

[11] T. Bonald, L. Mekinda, L. Muscariello, Fair throughput allocation in information-centric networks, Comput. Netw. 125 (2017) 122–131, http://dx.doi.org/10.1016/j.comnet.2017.05.019.

[12] S. Ioannidis, E. Yeh, Adaptive caching networks with optimality guarantees, ACM SIGMETRICS Perform. Eval. Rev. 44 (1) (2016) 113–124, http://dx.doi.org/10.1145/2964791.2901467.

[13] M. Mahdian, A. Moharrer, S. Ioannidis, E. Yeh, Kelly cache networks, IEEE/ACM Trans. Netw. 28 (3) (2020) 1130–1143, http://dx.doi.org/10.1109/TNET.2020.2982863.
[14] S. Ioannidis, E. Yeh, Jointly optimal routing and caching for arbitrary network topologies, IEEE J. Sel. Areas Commun. 36 (6) (2018) 1258–1275, http://dx.doi.org/10.1109/JSAC.2018.2844981.
[15] J. Li, T.K. Phan, W.K. Chai, D. Tuncer, G. Pavlou, D. Griffin, M. Rio, Dr-cache: Distributed resilient caching with latency guarantees, in: IEEE Conference on Computer Communications, INFOCOM 2018, 2018, pp. 441–449, http://dx.doi.org/10.1109/INFOCOM.2018.8486316.
[16] S. Jagabathula, D. Shah, Fair scheduling in networks through packet election, IEEE Trans. Inform. Theory 57 (3) (2011) 1368–1381, http://dx.doi.org/10.1109/TIT.2010.2103851.
[17] F. Paganini, A. Tang, A. Ferragut, L.L. Andrew, Network stability under alpha fair bandwidth allocation with general file size distribution, IEEE Trans. Automat. Control 57 (3) (2011) 579–591, http://dx.doi.org/10.1109/TAC.2011.2160013.
[18] R. Srikant, The Mathematics of Internet Congestion Control, Springer Science & Business Media, Birkhäuser Boston, MA, 2012, http://dx.doi.org/10.1007/978-0-8176-8216-3.
[19] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, S. Katti, Numfabric: Fast and flexible bandwidth allocation in datacenters, in: Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 188–201, http://dx.doi.org/10.1145/2934872.2934890.
[20] J. Mo, J. Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Trans. Netw. 8 (5) (2000) 556–567, http://dx.doi.org/10.1109/90.879343.
[21] A.A. Ageev, M.I. Sviridenko, Pipage rounding: A new method of constructing algorithms with proven performance guarantee, J. Comb. Optim. 8 (3) (2004) 307–328, http://dx.doi.org/10.1023/b:joco.0000038913.96607.c2.
[22] S. Shukla, A.A. Abouzeid, Proactive retention aware caching, in: IEEE Conference on Computer Communications, INFOCOM 2017, 2017, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2017.8057029.
[23] Z. Yang, D. Jia, S. Ioannidis, N. Mi, B. Sheng, Intermediate data caching optimization for multi-stage and parallel big data frameworks, in: 2018 IEEE 11th International Conference on Cloud Computing, 2018, pp. 277–284, http://dx.doi.org/10.1109/CLOUD.2018.00042.
[24] Y. Li, S. Ioannidis, Universally stable cache networks, in: IEEE Conference on Computer Communications, INFOCOM 2020, 2020, http://dx.doi.org/10.1109/INFOCOM41043.2020.9155416.
[25] K. Poularakis, L. Tassiulas, On the complexity of optimal content placement in hierarchical caching networks, IEEE Trans. Commun. 64 (5) (2016) 2092–2103, http://dx.doi.org/10.1109/TCOMM.2016.2545655.
[26] K. Poularakis, L. Tassiulas, Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks, IEEE Trans. Mob. Comput. 16 (3) (2016) 675–687, http://dx.doi.org/10.1109/TMC.2016.2575837.
[27] B. Liu, K. Poularakis, L. Tassiulas, T. Jiang, Joint caching and routing in congestible networks of arbitrary topology, IEEE Internet Things J. 6 (6) (2019) 10105–10118, http://dx.doi.org/10.1109/JIOT.2019.2935742.
[28] K. Poularakis, J. Llorca, A.M. Tulino, I. Taylor, L. Tassiulas, Service placement and request routing in MEC networks with storage, computation, and communication constraints, IEEE/ACM Trans. Netw. 28 (3) (2020) 1047–1060, http://dx.doi.org/10.1109/TNET.2020.2980175.
[29] G. Domingues, E.d.S. e Silva, R.M. Leao, D.S. Menasche, D. Towsley, Enabling opportunistic search and placement in cache networks, Comput. Netw. 119 (2017) 17–34, http://dx.doi.org/10.1016/j.comnet.2017.03.005.
[30] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, L. Tassiulas, Distributed caching algorithms in the realm of layered video streaming, IEEE Trans. Mob. Comput. 18 (4) (2018) 757–770, http://dx.doi.org/10.1109/TMC.2018.2850818.
[31] H. Che, Y. Tung, Z. Wang, Hierarchical web caching systems: Modeling, design and experimental results, IEEE J. Sel. Areas Commun. 20 (7) (2002) 1305–1314, http://dx.doi.org/10.1109/JSAC.2002.801752.
[32] B. Jiang, P. Nain, D. Towsley, On the convergence of the TTL approximation for an LRU cache under independent stationary request processes, ACM Trans. Model. Perform. Eval. Comput. Syst. 3 (4) (2018) 1–31, http://dx.doi.org/10.1145/3239164.
[33] C. Fricker, P. Robert, J. Roberts, A versatile and accurate approximation for LRU cache performance, in: 2012 24th International Teletraffic Congress, ITC 24, IEEE, 2012, pp. 1–8.
[34] V. Martina, M. Garetto, E. Leonardi, A unified approach to the performance analysis of caching systems, in: IEEE Conference on Computer Communications, INFOCOM 2014, 2014, pp. 2040–2048, http://dx.doi.org/10.1109/INFOCOM.2014.6848145.
[35] G. Bianchi, A. Detti, A. Caponi, N. Blefari Melazzi, Check before storing: What is the performance price of content integrity verification in LRU caching? ACM SIGCOMM Comput. Commun. Rev. 43 (3) (2013) 59–67, http://dx.doi.org/10.1145/2500098.2500106.
[36] N.C. Fofack, P. Nain, G. Neglia, D. Towsley, Analysis of TTL-based cache networks, in: IEEE 6th International ICST Conference on Performance Evaluation Methodologies and Tools, 2012, pp. 1–10.
[37] D.S. Berger, P. Gland, S. Singla, F. Ciucu, Exact analysis of TTL cache networks, Perform. Eval. 79 (2014) 2–23, http://dx.doi.org/10.1016/j.peva.2014.07.001.
[38] N.K. Panigrahy, J. Li, F. Zafari, D. Towsley, P. Yu, Optimizing timer-based policies for general cache networks, arXiv preprint arXiv:1711.03941.
[39] L. Wang, G. Tyson, J. Kangasharju, J. Crowcroft, Faircache: Introducing fairness to ICN caching, in: IEEE 24th International Conference on Network Protocols, ICNP, 2016, pp. 1–10, http://dx.doi.org/10.1109/ICNP.2016.7784440.
[40] K. Avrachenkov, J. Goseling, B. Serbetci, Distributed cooperative caching for utility maximization of vod systems, in: IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications, SPAWC 2019, 2019, http://dx.doi.org/10.1109/SPAWC.2019.8815571.
[41] S. Rezvani, N. Mokari, M.R. Javan, E. Jorswieck, Fairness and transmission-aware caching and delivery policies in ofdma-based hetnets, IEEE Trans. Mob. Comput. 19 (2) (2019) 331–346, http://dx.doi.org/10.1109/TMC.2019.2892978.
[42] N. Panigrahy, J. Li, F. Zafari, D. Towsley, P. Yu, Jointly compressing and caching data in wireless sensor networks, in: IEEE International Conference on Smart Computing, SMARTCOMP 2019, 2019, pp. 57–62, http://dx.doi.org/10.1109/SMARTCOMP.2019.00029.
[43] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functions, Math. Program. 14 (1) (1978) 265–294.
[44] G. Calinescu, C. Chekuri, M. Pál, J. Vondrák, Maximizing a monotone submodular function subject to a matroid constraint, SIAM J. Comput. 40 (6) (2011) 1740–1766, http://dx.doi.org/10.1137/080733991.
[45] J. Vondrák, Optimal approximation for the submodular welfare problem in the value oracle model, in: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, 2008, pp. 67–74, http://dx.doi.org/10.1145/1374376.1374389.
[46] M. Sviridenko, J. Vondrák, J. Ward, Optimal approximation for submodular and supermodular optimization with bounded curvature, Math. Oper. Res. 42 (4) (2017) 1197–1218, http://dx.doi.org/10.1287/moor.2016.0842.
[47] P.R. Goundan, A.S. Schulz, Revisiting the greedy approach to submodular set function maximization, Working paper, Massachusetts Institute of Technology, 2007.
[48] A. Krause, D. Golovin, Tractability: Practical Approaches to Hard Problems, Cambridge University Press, Cambridge, United Kindom, 2014.
[49] F. Kelly, E. Yudovina, Stochastic Networks, Vol. 2, Cambridge University Press, Cambridge, United Kindom, 2014.
[50] G. Calinescu, C. Chekuri, M. Pál, J. Vondrák, Maximizing a submodular set function subject to a matroid constraint, in: International Conference on Integer Programming and Combinatorial Optimization, Springer, 2007, pp. 182–196, http://dx.doi.org/10.1007/978-3-540-72792-7_15.
[51] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, New York, 2006.

[52] B. Blaszczyszyn, A. Giovanidis, Optimal geographic caching in cellular networks, in: 2015 IEEE International Conference on Communications, ICC, 2015, http://dx.doi.org/10.1109/ICC.2015.7248843.
[53] S. Bubeck, Convex optimization: Algorithms and complexity, Found. Trends Mach. Learn. 8 (3–4) (2015) 231–357.
[54] S. Ioannidis, E. Yeh, Adaptive caching networks with optimality guarantees, IEEE/ACM Trans. Netw. 26 (2) (2018) 737–750, http://dx.doi.org/10.1109/TNET.2018.2793581.
[55] J. Kleinberg, The small-world phenomenon: An algorithmic perspective, in: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, 2000, pp. 163–170, http://dx.doi.org/10.1145/335305.335325.
[56] E. Cohen, S. Shenker, Replication strategies in unstructured peer-to-peer networks, ACM SIGCOMM Comput. Commun. Rev. 32 (2002) 177–190, http://dx.doi.org/10.1145/964725.633043.
[57] D. Bertsimas, V.F. Farias, N. Trichakis, The price of fairness, Oper. Res. 59 (1) (2011) 17–31, http://dx.doi.org/10.1287/opre.1100.0865.