On the Complexity of Optimal Request Routing and Content Caching in Heterogeneous Cache Networks

Mostafa Dehghan[1], Bo Jiang[1], Anand Seetharam[2], Ting He[3], Theodoros Salonidis[4],
Jim Kurose[1], Don Towsley[1], and Ramesh Sitaraman[1,5]
[1]University of Massachusetts Amherst, [2]SUNY Binghamton, [3]Pennsylvania State University,
[4]IBM T.J. Watson Research Center, [5]Akamai Technologies Inc
{mdehghan, bjiang, kurose, towsley, ramesh}@cs.umass.edu,
anand@cs.binghamton.edu, tzh58@psu.edu, tsaloni@us.ibm.com

***Abstract*—In-network content caching has been deployed in both the Internet and cellular networks to reduce content-access delay. We investigate the problem of developing optimal joint routing and caching policies in a network supporting in-network caching with the goal of minimizing expected content-access delay. Here, needed content can either be accessed directly from a back-end server (where content resides permanently) or be obtained from one of multiple in-network caches. To access content, users must thus decide whether to route their requests to a cache or to the back-end server. Additionally, caches must decide which content to cache. We investigate two variants of the problem where the paths to the back-end server can be considered as either *i)* congestion-sensitive or *ii)* congestion-insensitive, reflecting whether or not the delay experienced by a request sent to the back-end server depends on the request load, respectively. We show that the problem of optimal joint caching and routing is NP-complete in both cases. We prove that under the congestion-insensitive delay model, the problem can be solved optimally in polynomial time if each piece of content is requested by only one user, or when there are at most two caches in the network. We also identify the structural property of the user-cache graph that makes the problem NP-complete. For the congestion-sensitive delay model, we prove that the problem remains NP-complete even if there is only one cache in the network and each content is requested by only one user. We show that approximate solutions can be found for both cases within a $(1 - 1/e)$ factor from the optimal, and demonstrate a greedy solution that is numerically shown to be within 1% of optimal for small problem sizes. Through trace-driven simulations we evaluate the performance of our greedy solutions to joint caching and routing, which show up to 50% reduction in average delay over the solution of optimized routing to LRU caches.**

***Index Terms*—content placement, routing, joint optimization, complexity**

I. Introduction

With the rapid growth of data traffic over cellular networks, it has been widely acknowledged that the conventional macro-cell architecture (4G-LTE) will not be able to support such traffic growth [1]. Since content caching has proven to reduce server traffic by more than 60% [2], [3], in-network caching of content at storage-enabled nodes has received considerable attention as a means to reduce the use of network link bandwidth while improving the delay performance of end users by bringing content closer to the users. The benefits of in-network content caching has been demonstrated in the context of CDNs as well as in hybrid networks comprised of cellular and MANETs or femto-cell networks [4]–[6]. The *FemtoCaching* architecture [7] effectively replaces back-haul capacity with storage capacity, allowing user content requests to be satisfied by caches at the wireless edge, with back-haul links being used primarily to refresh cache content. Prior work [7]–[9] has focused on the content placement problem, *i.e.*, determining which content should be placed at which caches for a given topology and file popularity distribution, under the assumption that users greedily access content over the minimum delay path.

In this paper, we study a *joint* problem of caching and routing, considering the inter-related routing and caching decisions, with the goal of minimizing average content access delay over all user requests. We consider a scenario in which users request content that is permanently stored at a back-end server, and that can be accessed in one of two ways − either directly from the back-end server over an uncached path, or via one of the caches located within the network. These caches can be located either at the network edge as in the case of a CDN, or can be in-network caches in the case of a hybrid wireless network. In the latter setting, MANET-like routing might be used to route content requests to in-network caches, while a separate (and potentially costly, congested, and/or slower speed) cellular link might be used to directly access the back-end server. If a request is routed to an in-network cache that holds the content, the request is served immediately. Otherwise, the cache must download the content from the back-end server before serving it to the user, incurring an additional delay. Additionally, the cache must decide whether or not to store the downloaded content.

We address the following question − how should users route requests between the in-network caches and the back-end server, and what in-network cache management policy should be adopted to minimize overall network delay? We consider two variants of the problem. In the first case, referred to as the *congestion-insensitive case*, we assume that delays are independent of the traffic load on all paths. In the second case, referred to as the *congestion-sensitive case*, we assume that the delay to the back-end server depends on the traffic load; we model the congestion-sensitive delay using G/G/1 queues. In

a hybrid cellular/MANET network, the uncached path in the congestion-insensitive model corresponds to GBR (guaranteed bit rate) 3GPP bearer service, while in the congestion-sensitive model it corresponds to Non-GBR Aggregate Maximum Bit Rate (AMBR) bearer service [10]. We investigate the time complexity of finding the optimal solution for the joint caching and routing problem for both cases.

Our goal in this paper is twofold. First, we seek an understanding of the computational complexity of the joint caching and routing problem: Can the general problem be solved optimally in polynomial time? If not, are there problem instances that are tractable and what aspects make the general problem intractable? Second, we seek efficient approximate solutions to the joint caching and routing problem that perform well in practice.

Our contributions can be summarized as follows:

- We provide a unified optimization formulation for the joint caching and routing problem for the congestion-insensitive and congestion-sensitive models and prove that the problem is NP-complete in both cases.
- For the congestion-insensitive uncached path model, we show that the optimal solution can be found in polynomial time if each content is requested by only one user, or when the number of caches in the network is at most two. Moreover, we identify the root cause of the problem complexity in general cases – cycles with an odd number of users and caches in the bipartite graph representing connections between users and caches. For the congestion-sensitive uncached path model, however, we show that the problem remains NP-complete even if there is only one cache in the network and each content is requested by only one user.
- We develop a greedy caching and routing algorithm that achieves an average delay within a $(1-1/e)$ factor of the optimal solution and a second greedy algorithm of lower complexity.
- We evaluate the performance of the proposed greedy algorithms together with the optimal solution (via brute-force search) and a baseline solution based on LRU through numerical evaluations and trace-driven simulations. Numerical results show that the greedy algorithms perform close to optimal when computing the optimal solution is feasible. Results from trace-driven simulations show that the greedy algorithms yield significant performance improvement compared to solutions based on traditional LRU caching policy.

The remainder of this paper is organized as follows. In Section II, we describe our network model, and in Section III, we formulate the problem of optimal joint caching and routing. In Sections IV and V, we present our complexity results for the congestion-insensitive and congestion-sensitive delay models, respectively. Section VI explains the approximate algorithm, and Section VII presents simulation results. Section VIII reviews the related work, and Section IX concludes the paper.

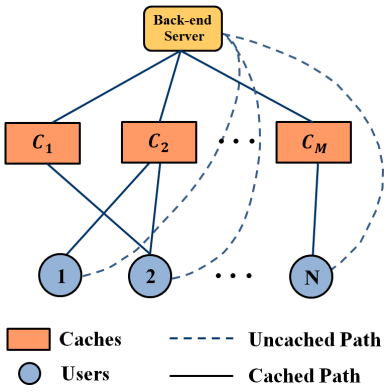

Fig. 1: Hybrid network with in-network caching

II. Network Model

In this section, we consider the network shown in Figure 1 with N users generating requests for a set of K unique files $F = \{f_1, f_2, \ldots, f_K\}$ of unit size. Throughout this paper, we will use the terms content and file interchangeably. We assume that these files reside permanently at the back-end server. As shown in Figure 1, there are M caches in the network that can serve user requests.

All files are available at the back-end server and users are directly connected to this server via a cellular infrastructure. We refer to the cellular path between the user and the back-end server as the *uncached path*. Each user can also access a subset of the M in-network caches where the content might be cached. We refer to a connection between a user and a cache as a *cached path*.

Let C_m denote the storage capacity of the m-th cache measured by the maximum number of files it can store. If user i requests file j and it is present in the cache, then the request is served immediately. We refer to this event as a cache hit. However, if content j is not present in the cache, the cache then forwards the request to the back-end server, downloads file j from the back-end server and forwards it to the user. We refer to this event as a cache miss, since it was necessary to download content from the back-end server in order to satisfy the request. *Note that in case of a cache miss, the cache can decide whether to keep the downloaded content.*

User i generates requests for the files in F according to a Poisson process of aggregate rate λ_i. Aggregate request rate of all users is λ. We assume the independent reference model (IRM) and denote by q_{ij} the probability that a request generated by user i is for file j (referred to as the *file popularity*). The popularity of the same file can vary from one user to another.

Let a_{im} denote the existence of a connection between user i and cache m, with $a_{im} = 1$ if user i is connected to cache m, and $a_{im} = 0$, otherwise. We consider two models for the delay over the paths to the back-end server. The first is a congestion-insensitive delay model where the delays are independent of the traffic load on the links to the back-end server. In this case, the average delay experienced for a request by user i sent over the uncached path is d_i^b. Also, for the user-

cache connections, we denote the average delays incurred by user i in the event of a cache hit or miss at cache m by d^h_{im} and d^c_{im}, respectively. We assume that $d^h_{im} < d^b_i < d^c_{im}$ if $a_{im} = 1$. The second model is a congestion-sensitive delay model where delays experienced over the paths to the back-end server depend on the traffic load. In this case, we assume that the requests sent over the uncached paths, and the requests missed from caches experience constant initial delays d^b_i and d^c_{im} as well as load-dependent (queueing) delays captured by convex functions $d_b(\cdot)$ and $d_c(\cdot)$, respectively.

III. Problem Formulation

In this work, we consider a joint caching and routing problem with the goal of minimizing average content access delay over all user requests for all files. The solution to this problem requires addressing two closely-related questions 1) How should cache contents be managed - which files should be kept in the caches, and what cache replacement strategy should be used? and 2) How should users route their requests between the cached and uncached paths?

For our routing policy, we define a decision variable p_{ijm} that denotes the fraction of the requests of user i for content j sent to cache m. User i sends the remaining $1 - \sum_m p_{ijm}$ fraction of her requests for content j to the back-end server through the uncached path.

It is shown in [11] that *static caching* minimizes expected delay for a single cache when user demands and routing are fixed. With static caching, a set of files is stored in the cache, and the cache content does not change in the event of a cache hit or miss. The argument in [11] was extended in [12] and [13] to a network of caches to show that static caching achieves minimum expected delay under a fixed routing policy. Hence, we define the binary variables $x_{jm} \in \{0, 1\}$ to denote the content placement in caches, where $x_{jm} = 1$ indicates file j is stored in cache m and $x_{jm} = 0$ indicates otherwise.

We denote by $D(\mathbf{x}, \mathbf{p})$ the expected delay obtained by a content placement strategy $\mathbf{x} = [x_{jm}]$, and a routing strategy $\mathbf{p} = [p_{ijm}]$. We also use $D_\varnothing$ to denote the expected delay when no content is cached, where $D_\varnothing$ is assumed to be finite. The caching *gain* can then be defined as $G(\mathbf{x}, \mathbf{p}) \triangleq D_\varnothing - D(\mathbf{x}, \mathbf{p})$. The goal of joint caching and routing is to maximize $G(\mathbf{x}, \mathbf{p})$ which can equivalently be obtained by solving the following Mixed-Integer Program (MIP):

$$\begin{aligned}
\text{minimize } D(\mathbf{x}, \mathbf{p}) = \frac{1}{\lambda}\Bigg[& \sum_i \sum_j \lambda_i q_{ij} \Bigg(\sum_m p_{ijm} x_{jm} d^h_{im} \\
& + (1 - \sum_m p_{ijm}) d^b_i + \sum_m (1 - x_{jm}) p_{ijm} d^c_{im} \Bigg) \\
& + \lambda_b(\mathbf{p}) d_b\Big(\lambda_b(\mathbf{p})\Big) + \lambda_c(\mathbf{x}, \mathbf{p}) d_c\Big(\lambda_c(\mathbf{x}, \mathbf{p})\Big) \Bigg], \\
\text{such that } & \sum_m p_{ijm} \le 1 \quad \forall i, j \qquad (1) \\
& \sum_j x_{jm} \le C_m \quad \forall m \\
& x_{jm} \in \{0, 1\} \quad \forall j, m \\
& 0 \le p_{ijm} \le a_{im} \quad \forall i, j, m.
\end{aligned}$$

In the formulation above,

$$\lambda_b(\mathbf{p}) \triangleq \sum_i \sum_j \lambda_i q_{ij} (1 - \sum_m p_{ijm}),$$

and

$$\lambda_c(\mathbf{x}, \mathbf{p}) \triangleq \sum_i \sum_j \lambda_i q_{ij} \sum_m (1 - x_{jm}) p_{ijm},$$

denote the request load over the uncached paths, and the load of requests missed from caches, respectively.

In the next two sections, we express the delay function $D(\mathbf{x}, \mathbf{p})$ for the cases of *i)* congestion-insensitive and *ii)* congestion-sensitive uncached path delay models, and discuss why the joint caching and routing problem is NP-complete.

IV. Congestion-Insensitive Uncached Path

First, we consider the case where delays on the uncached path, d^b_i, do not depend on the traffic load on the back-end server. Hence, throughout this section we assume that $d_b(\cdot) = d_c(\cdot) = 0$.

Without loss of generality, we assume that $d^h_{im} < d^b_i < d^c_{im}$ whenever user i is connected to cache m, *i.e.*, $a_{im} = 1$. The routing variables p_{ijm} for user i are easily determined when the above assumption does not hold. Note that if $d^b_i \ge d^c_{im}$, user i will never use the uncached path. Also, if $d^b_i \le d^h_{im}$, user i will never use cache m.

It is easy to see that with the congestion-insensitive model, given a content placement, the average minimum delay is obtained by routing requests for the cached content to caches, and routing the remaining requests to the uncached path. Note that under this routing policy no cache misses occur.

Note that $D(\mathbf{x}, \mathbf{p})$ is a linear function of the routing variables. Also note the additional constraint $p_{ijm} \le x_{jm} \cdot a_{im}$, which is due to the fact that only requests for cached content are routed to caches. Since $d^h_{im} < d^b_i < d^c_{im}$, users have no incentive to split the traffic for any content between the cached and uncached paths, and hence there will be an optimal solution such that no routing variable has a fractional value, *i.e.*, $p_{ijm} \in \{0, 1\}$.

A. Hardness of General Case

The above formulation of the joint caching and routing problem is a generalization of the Helper Decision Problem (HDP) proved to be NP-complete in [7]. Our formulation is more general as we consider non-homogeneous delays for the cached and uncached paths. Therefore, we have the following result.

Theorem 1. *The optimal joint caching and routing problem with congestion-insensitive uncached paths is NP-complete.*

Proof. HDP reduces to the optimization problem in (1) by setting $d_b(\cdot) = d_c(\cdot) = 0$, $d^b_i = 1$, $d^h_{im} = 0$, and $C_m = C$, where C is the cache size at all caches in HDP. Hence, joint

caching and routing problem is NP-hard. Moreover, for any given routing and caching, average delay can be computed in polynomial time. Therefore, the joint caching and routing problem in case of congestion-insensitive uncached paths is NP-complete. □

Although the problem is NP-complete in general, we will show that the joint caching and routing problem can be solved in polynomial time for several special cases. We will also identify what makes the problem "hard" in general. We first consider a restrictive setting where each user is interested in only one file and each file is requested by only one user. Next, we consider a network with two caches (but each user may be interested in an arbitrary number of files). We present polynomial time solutions for both cases. Finally, we present an example that demonstrates what we conjecture to be *the* source of the complexity of this problem.

B. Special Case: One File per User

Consider the network illustrated in Figure 1, but assume each user is interested in only one file, *i.e.*, $q_{ii} = 1$, and $q_{ij} = 0$ for $i \neq j$. In this case, the optimal solution to the joint caching and routing problem can be found in polynomial time based on a solution to the *maximum weighted matching* problem. A similar reduction of a caching problem to the maximum weighted matching problem was also previously presented in [14].

Note that in this case, the number of files equals the number of users, *i.e.*, $N = K$. To avoid triviality, we assume that the number of users is larger than the capacity of each cache in the network, *i.e.*, $C_m < N, \forall m$.

Theorem 2. *The solution to the joint caching and routing problem with congestion-insensitive uncached paths in case of one file per user can be computed in polynomial time.*

Proof. The assumption that each user is interested in only one file allows us to rewrite the objective function in (1) as

$$D(\mathbf{x}, \mathbf{p}) = \frac{1}{\lambda}\Big(\sum_{i=1}^{N} \lambda_i d_i^b - \sum_i \sum_m \lambda_i p_{iim}(d_i^b - d_{im}^h)\Big).$$

Since $\sum_{i=1}^{N} \lambda_i d_i^b$ is a constant independent of the decision variables, minimizing the above objective function is equivalent to maximizing $\sum_i \sum_m \lambda_i p_{iim}(d_i^b - d_{im}^h)$. Note that $\lambda_i(d_i^b - d_{im}^h)$ can be interpreted as the gain obtained by having file i in cache m. This problem can then be naturally seen as matching files to caches with the goal of maximizing the sum of individual gains. In what follows, we map this problem to the maximum weighted matching problem.

For each cache of size C_m, we introduce C_m nodes $\{v_m^1, v_m^2, \ldots, v_m^{C_m}\}$ representing unit size micro-caches that form cache m. Let $V = \{v_1^1, v_1^2, \ldots, v_1^{C_1}, \ldots, v_M^1, \ldots, v_M^{C_M}\}$ denote the set of all such nodes, and let $U = \{u_1, u_2, \ldots, u_N\}$ denote the set of all files. We define the bipartite graph $G(U, V, E)$ with $\lambda_i(d_i^b - d_{im}^h)$ as the weight of the edges connecting node u_i to nodes $v_m^s, \forall s \in \{1, 2, \ldots, C_m\}$. Figure 2 demonstrates a bipartite graph with user/file nodes u and the micro-cache nodes v with the edge weights shown for some of the edges. Note that the bipartite graph consists of $|U| + |V| = N + \sum_m C_m$ vertices and $|E| = O(N \sum_m C_m)$ edges.

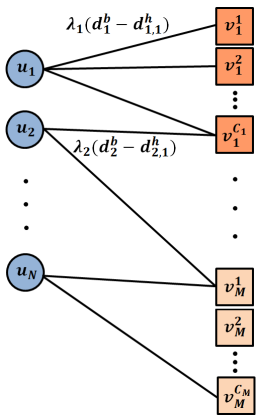

Fig. 2: Modeling content placement as a maximum weighted matching problem. Each user is interested in only one file and each file is requested by only one user. Problem can be solved by matching users to cache spaces.

The optimal solution to the joint content placement and routing problem corresponds to the maximum weighted matching for graph G. The edges selected in the maximum matching determine what content should be placed in which cache. Users then route to caches for cached content, and to the uncached path for the remaining files.

The maximum weighted matching problem for bipartite graphs can be solved in $O(|V|^2|E|)$ using the Hungarian algorithm [15]. In our context, the complexity is $O(M^3N^4)$. Note that $\sum_m C_m = O(MN)$ as we assume $C_m < N, \forall m$. Therefore, we can solve the joint caching and routing problem in polynomial time when users are interested in one file only. □

C. Special Case: Network with Two Caches

Next, we show that the optimal solution for the joint caching and routing problem can be found in polynomial time when there are only two caches in the network. Specifically, we prove that the solution to the integer program (1) can be found in polynomial time when there are two caches in the network.

By relaxing the integer constraints on content placement variables, x_{jm}, and allowing them to take real values, *i.e.*, $0 \leq x_{jm} \leq 1$, we obtain a linear problem (LP) that is generally referred to as the "relaxed" problem. Since the objective function in (1) is convex, the solution to the *relaxed problem* can be found in polynomial time for all instances of the problem. Note that the set of constraints in the relaxed version of (1), namely, *i)* $\sum_m p_{ijm} \leq 1$, *ii)* $\sum_j x_{jm} \leq C_m$, *iii)* $-x_{jm} \leq 0,\ x_{jm} \leq 1$, and *iv)* $-p_{ijm} \leq 0,\ p_{ijm} - x_{jm} \cdot a_{im} \leq 0$ can be written in the linear form $\mathbf{Az} \leq \mathbf{b}$ where the entries of $\mathbf{A}$ and $\mathbf{b}$ are all integers, and $\mathbf{z}$ consists of the x_{jm} and p_{ijm} entries. We will show that for a network with two caches solving the relaxed program will produce integral solutions.

Before delving into the proof we introduce some definitions and results from [16]:

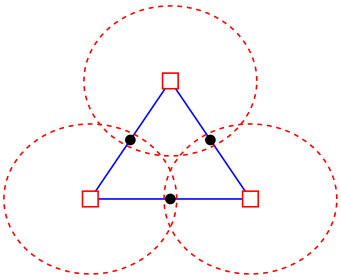

Fig. 3: A network with three users (solid circles) and three caches (squares). Each user is in the communication range of two of the caches.

Definition 1. *A square integer matrix is called* unimodular *if it has determinant* $+1$ *or* -1.

Definition 2. *An* $m \times n$ *integral matrix* $\mathbf{A}$ *is* totally unimodular *if the determinant of every square submatrix is* 0, 1, *or* -1.

Proposition 1. *If for a linear program* $\{\max \mathbf{c}^T\mathbf{z} : \mathbf{Az} \leq \mathbf{b}\}$, $\mathbf{A}$ *is totally unimodular and* $\mathbf{b}$ *is integral, then all vertex solutions of the linear program are integral.*

From Proposition 1, then, it suffices to show that the matrix $\mathbf{A}$ is totally unimodular for a network with two caches to prove that the optimization problem can be solved in polynomial time. To prove that the matrix $\mathbf{A}$ is totally unimodular we use the following result from [17]:

Proposition 2. *A matrix is totally unimodular if and only if for every subset* R *of rows, there is an assignment* $s : R \to \pm 1$ *of signs to rows so that the signed sum* $\sum_{r \in R} s(r)r$ *(which is a row vector of the same width as the matrix) has all its entries in* $\{0, \pm 1\}$.

Theorem 3. *For a network with two caches, the LP relaxation of* (1) *with* $d_b(\cdot) = d_c(\cdot) = 0$ *produces an integral solution in polynomial time.*

Proof. In Appendix A, we give a constructive proof showing that for any subset R of rows of $\mathbf{A}$ we can find an assignment s that satisfies Proposition 2. □

D. Complexity Discussion

Consider a network with three users and three caches as depicted in Figure 3. With each user connected to two of the caches, the user-cache connections can be seen to form a cycle as demonstrated in Figure 4a. Assume all paths from users to caches have equal hit and miss delays. Also, assume that each cache has the capacity of storing one file, and that all three users are interested in two files, noted here as green and red.

For the above network, the optimal content placement is to replicate one of the files in two of the caches, and have one copy of the other file in the third cache, as shown in Figure 4b. The solution to the *relaxed* optimization problem however would be to store half of each file in each cache, *i.e.*, $x_{1m} = x_{2m} = 0.5$, which achieves strictly smaller average

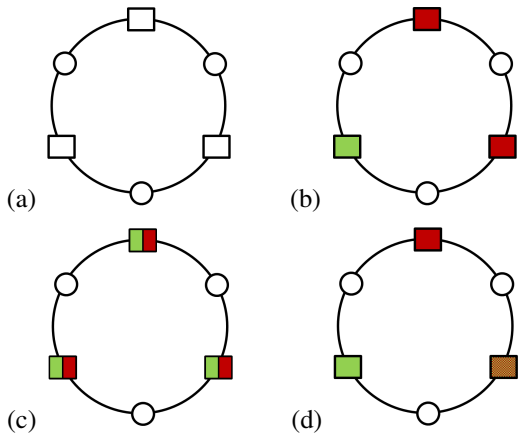

Fig. 4: (a) A network of three users (circles) connected to three caches (squares) forming a cycle. Users are equally interested in two files, red and green. (b) Optimal content placement according to binary placement decisions, *i.e.*, $x_{jm} \in \{0, 1\}$. (c) Optimal content placement assuming fractions of files can be stored in caches, *i.e.*, $0 \leq x_{jm} \leq 1$. (d) Optimal content placement with the possibility of content coding. A copy of the two files is stored in two of the caches, and the third cache keeps a coded copy, *e.g.*, XOR of the two files

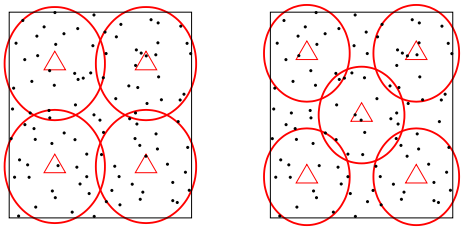

Fig. 5: Examples of network topologies conforming to conjecture criteria.

delay. This solution is illustrated in Figure 4c[1].

The above discussion shows how the solution to the MILP optimization problem differs from its relaxed counterpart for the network shown in Figure 3. Such mismatch between the two solutions is also observed for larger networks that contain odd number of users and odd number of caches connected in a way that form a cycle. We conjecture that these cycles are *the* source of complexity in the problem of joint caching and routing, and for networks that do not have any such cycles the solution to the optimization problem (1) matches that of the relaxed problem. More specifically we have:

Conjecture 1. *The optimal solution to the problem of joint caching and routing can be found in polynomial time if there are no cycles of length* $4k + 2$ *for any* $k \geq 1$ *in the bipartite graph corresponding to the user-cache connections.*

We have performed numerical simulations with thousands

[1]Note that we do not consider the solution of the relaxed problem as a legitimate content placement. Although it looks like all users can access the two files via the caches in Figure 4c, when splitting the files in halves, two of the caches will store the same half copy of a file, and the user connected to those caches will only get half of that file from the caches and still needs to use the uncached path for the other half. However, we acknowledge that with the possibility of coding, content placement can be done in such a way that users can get both files from caches, as is shown in Figure 4d. We are not considering coded content placement in this work.

of randomly generated sample problems similar to the ones shown in Figure 5, with networks of four and five caches and up to 100 users in the network. We have then solved the LP version of MILP (1) to compute the optimal caching and routing. For all these sample problems, we have observed that the optimal solutions are integral. Although not a proof, these results support our conjecture.

V. Congestion-Sensitive Uncached Path

Next, we consider the case where delays for the requests over the uncached paths and requests missed from caches depend on the load from such requests. Namely, we compute the delay over the uncached paths and the paths from the caches to the back-end server using the convex functions $d_b(\cdot)$ and $d_c(\cdot)$, respectively. The reason we treat requests over the uncached path and missed requests from caches separately is that these paths could use different infrastructures to reach the back-end server. For example, requests from mobile users over the uncached path could use the LTE infrastructure, while missed requests from caches deployed on WiFi access points could use a wireline broadband connection.

A. Hardness of General Case

Note that we can consider the congestion-insensitive delay model as a special case of the congestion-sensitive model where $d_b(\cdot) = d_c(\cdot) = 0$. Thus, this problem is NP-complete in general. In the remainder of this section, however, we will prove that the problem of joint caching and routing in the case of a congestion-sensitive delay model remains NP-complete even if there is only one cache in the network and each content is of interest to no more than one user.

B. Hardness of Single-Cache Case

Here, we consider a special case of the problem where the delays for the requests sent over the uncached paths are modeled as an M/M/1 queue, *i.e.*, $d_b(\lambda_b) = 1/(\mu_b - \lambda_b)$, where μ_b denotes the service rate. Also, the requests missed from caches are assumed to observe a constant delay d_i^c, *i.e.*, $d_c(\lambda_c) = 0$. Modifying the delay function $D(\mathbf{x}, \mathbf{p})$ in (1) for the case of one cache, *i.e.*, $M = 1$, and assuming each user is interested in only one file, *i.e.*, $q_{ii} = 1, \forall i$, and $q_{ij} = 0$ for $i \neq j$, we can rewrite the optimization problem as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{\lambda}\left[\sum_{i=1}^{N}\lambda_i x_i p_i d_i^h + \sum_{i=1}^{N}\lambda_i(1-x_i)p_i d_i^c \right. \\ & \left. + \sum_{i=1}^{N}\lambda_i(1-p_i)d_i^b + \frac{\sum_{i=1}^{N}\lambda_i(1-p_i)}{\mu_b - \sum_{i=1}^{N}\lambda_i(1-p_i)}\right] \\ \text{such that} \quad & \sum_{i=1}^{N} x_i \leq C \\ & 0 \leq p_i \leq a_i \\ & x_i \in \{0,1\}, \end{aligned} \tag{2}$$

where $p_i = p_{ii1}$ denotes the fraction of user i requests routed to the cache. Also, a_i denotes whether user i is connected to the cache.

To show that the above optimization problem is NP-complete, we consider the corresponding decision problem, Congestion Sensitive Delay Decision Problem (CSDDP).

Problem 1. *(Congestion Sensitive Delay Decision Problem) Let $\mathbf{\Lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_N]$ denote user request rates, and let $\mathbf{d}^h = [d_i^h]$, $\mathbf{d}^c = [d_i^c]$ and $\mathbf{d}^b = [d_i^b]$ denote the hit delay, miss delay and initial access delay of the uncached path, respectively. Also, let μ be the service rate of the back-end server, and C be the cache capacity.*
We ask the following question: given the parameters $(\mu_b, \mathbf{\Lambda}, \mathbf{d}^h, \mathbf{d}^c, \mathbf{d}^b, C)$ and a real number d, is there any assignment of $\mathbf{x} = [x_i]$ and $\mathbf{p} = [p_i]$ such that $D(\mathbf{x}, \mathbf{p}) \leq d$.

It is clear that for any given content placement $\mathbf{x}$ and routing policy $\mathbf{p}$ the answer to CSDDP can be verified in polynomial time, and hence CSDDP is in class NP. To prove that CSDDP is NP-hard, we use the fact that the Equal Cardinality Partition (ECP) problem define below is NP-hard.

Problem 2. *(Equal Cardinality Partition) Given a set A of n numbers, can A be partitioned into two disjoint subsets A_1 and A_2 such that $A = A_1 \cup A_2$, the sum of the numbers in A_1 equals the sum of the numbers in A_2 and that $|A_1| = |A_2|$?*

Lemma 1. *ECP is NP-hard.*

Proof. See Appendix B. □

By reducing ECP to CSDDP, we have the following result:

Theorem 4. *CSDDP is NP-complete.*

Proof. See Appendix C for a detailed proof. □

Although this problem is NP-complete even in a very restrictive case with one cache and each user requesting one file, in the next section we show that a greedy algorithm can find approximate solutions with guaranteed performance.

Note that problem formulation (1) assumes a single queue shared by all caches to the back-end server. An alternative choice is to have distinct queues from each cache to the back-end server. In that case, $\lambda_c d_c(\lambda_c)$ in (1) should be replaced with a sum of M delay terms, where M is the number of caches. The model with distinct queues also results in an NP-complete problem, since problem (1) is NP-complete when there is only one cache in the network.

VI. Approximation Algorithms

In this section, we show that the problem of joint caching and routing (for both congestion-insensitive and congestion-sensitive delay models) can be formulated as the maximization of a monotone submodular function subject to matroid constraints. This enables us to devise algorithms with provable approximation guarantees.

We first review the definition and properties of matroids [18], and monotone [19] and submodular [17] functions, and then show our problem can be formulated as the maximization of a monotone submodular function subject to matroid constraints.

Definition 3. *A matroid M is a pair $M = (S, I)$, where S is a finite set and $I \subseteq 2^S$ is a family of subsets of S with the following properties:*

1) $\emptyset \in I$,
2) *I is downward closed, i.e., if $Y \in I$ and $X \subseteq Y$, then $X \in I$,*
3) *If $X, Y \in I$, and $|X| < |Y|$, then $\exists y \in Y \backslash X$ such that $X \cup \{y\} \in I$.*

Definition 4. *Let S be a finite set. A set function $f : 2^S \to \mathbb{R}$ is submodular if for every $X, Y \subseteq S$ with $X \subseteq Y$ and every $x \in S \backslash Y$ we have*

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y).$$

Definition 5. *A set function f is monotone increasing if $X \subseteq Y$ implies that $f(X) \leq f(Y)$.*

Let X_m denote the set of files stored in cache m, and define $X = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_M$ to be the set of files stored in the M caches, where $\sqcup$ denotes disjoint union. X is the set equivalent of the binary content placement $\mathbf{x}$ defined in (1). Note that $|X_m| \leq C_m$

Let $S_m = \{s_{1m}, s_{2m}, \ldots, s_{Km}\}$ denote the set of all possible files that could be placed in cache m where s_{jm} denotes the storage of file j in cache m. The set element s_{jm} corresponds to the binary variable x_{jm} defined in the optimization problem (1) such that $x_{jm} = 1$ if and only if the element $s_{jm} \in X$. Define the super set $S = S_1 \cup S_2 \cup \ldots \cup S_M$ as the set of all possible content placements in the M caches. We have the following lemma.

Lemma 2. *The constraints in* (1) *form a matroid on S.*

Proof. For a given content placement $\mathbf{x}$, the optimal routing policy can be computed in polynomial time since $D_{\mathbf{x}}(\mathbf{p}) = D(\mathbf{p}; \mathbf{x})$ is a convex function. With that in mind, we can write the average delay as a function of the content placement $X \subseteq S$. Thus, the constraints on the capacities of the caches can be expressed as $X \subseteq \mathcal{I}$ where

$$\mathcal{I} = \{X \subseteq S : |X \cap S_m| \leq C_m, \forall m = 1, \ldots, M\}.$$

Note that $(S, \mathcal{I})$ defines a matroid. □

Let $d_{ij}(\mathbf{x})$ denote the minimum average delay for user i accessing file j through a cached path, given content placement $\mathbf{x}$, excluding queueing delay for fetching content from the back-end server in the case of a cache miss. We have

$$d_{ij}(\mathbf{x}) = \min_m d_{ijm},$$

where d_{ijm} denotes the average delay of accessing content j from cache m, excluding the queueing delay, defined as (x_{jm} indicates that file j is in cache m)

$$d_{ijm} = d^h_{im} x_{jm} + d^c_{im}(1 - x_{jm}).$$

Similarly, we define

$$y_{ij} = \max_m a_{im} x_{jm},$$

denoting whether user i can access content j from a neighboring cache.

Given the content placement in the caches, let $p_{ij} \triangleq \sum_m p_{ijm}$ denote the fraction of the traffic for which user i uses the cached paths to access content j. Also, let

$$\lambda_c(\mathbf{p}) = \sum_{i,j} \lambda_i q_{ij}(1 - y_{ij}) p_{ij},$$

and

$$\lambda_b(\mathbf{p}) = \sum_{i,j} \lambda_i q_{i,j}(1 - p_{ij}),$$

denote the aggregate request rate for missed requests, and requests sent over the uncached paths, respectively. We rewrite the delay functions for the congestion-insensitive and the congestion-sensitive models as

$$D(\mathbf{p}; \mathbf{x}) = \frac{1}{\lambda} \left(\sum_{i,j} \lambda_i q_{ij} p_{ij} d_{ij}(\mathbf{x}) + \sum_{i,j} \lambda_i q_{ij}(1 - p_{ij}) d^b_i \right),$$

and

$$D(\mathbf{p}; \mathbf{x}) = \frac{1}{\lambda} \Bigg[\sum_{i,j} \lambda_i q_{ij} \Big(p_{ij} d_{ij}(\mathbf{x}) + (1 - p_{ij}) d^b_i \Big) + \lambda_b(\mathbf{p}) d_b\Big(\lambda_b(\mathbf{p})\Big) + \lambda_c(\mathbf{p}) d_c\Big(\lambda_c(\mathbf{p})\Big) \Bigg],$$

respectively. The optimal routing policy for a given content placement $\mathbf{x}$, then, is one that minimizes $D(\mathbf{p}; \mathbf{x})$, and can be found by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad D(\mathbf{p}; \mathbf{x}) \\ &\text{such that} \quad 0 \leq p_{ij} \leq 1 \quad \forall i, j. \end{aligned}$$

Note that $D(\mathbf{p}; \mathbf{x})$ is convex and the above optimization problem can be solved in polynomial time.

Let $\mathbf{x}_X$ be the equivalent binary representation of the content placement set X. It is clear that adding items to the set X can only decrease the value of $D(\mathbf{p}; \mathbf{x}_X)$. Moreover, one might expect that adding an item to a set containing a smaller number of files might decrease the delay by a larger amount compared to adding the item to a set containing a larger number of files. We formally prove this statement through the following lemma for both congestion-insensitive and congestion-sensitive delay models bearing in mind that $D_\emptyset$ denotes the expected delay with no files cached:

Lemma 3. *Let $\mathcal{P}$ denote all routing policies. For $X \subseteq S$, the function $G(X) = D_\emptyset - \min_{\mathbf{p} \in \mathcal{P}} D(\mathbf{p}; \mathbf{x}_X)$ is a monotone increasing and submodular function.*

Proof. See Appendix D for a detailed proof. □

A direct consequence of Lemma 3 is that the objective of the joint caching and routing problem is to maximize a monotone submodular function. Therefore,

Theorem 5. *The approximate solution obtained by the greedy algorithm in Algorithm 1 is within a $(1 - 1/e)$ factor of the*

optimal solution $G(X^*)$.

Proof. It was shown in [20] that the greedy algorithm for maximizing a monotone submodular set function with matriod constraints yields a $(1-1/e)$-approximation. □

Algorithm 1 starts with empty caches and at each step greedily adds a file to the cache that maximizes function G. This process continues until all caches are filled to capacity. Optimal routing is then determined based on the content placement.

Algorithm 1 GreedyWG: A greedy approximation with performance guarantees.

$S \leftarrow \{s_{jm} : 1 \leq j \leq K, 1 \leq m \leq M\}$
$X_m \leftarrow \emptyset, \forall m$
$X \leftarrow \emptyset$
for $c \leftarrow 1$ **to** $\sum_m C_m$ **do**
  $s_{j^*m^*} \leftarrow \arg\max_{s_{jm} \in S} G(X \cup \{s_{jm}\})$
  $X_{m^*} \leftarrow X_{m^*} \cup \{s_{j^*m^*}\}$
  $X \leftarrow X \cup \{s_{j^*m^*}\}$
  if $|X_{m^*}| = C_{m^*}$ **then**
    $S \leftarrow S \backslash s_{jm^*}, \forall j$
  else
    $S \leftarrow S \backslash s_{j^*m^*}$
Content placement is done according to X.
Determine the routing as $\mathbf{p}^* \leftarrow \arg\min_{\mathbf{p}} D(\mathbf{p}; \mathbf{x}_X)$.

Although the greedy algorithm in Algorithm 1 is guaranteed to find solutions within a $(1-1/e)$ factor of the optimal solution, its complexity is high, $O(M^2N^2K^2 \log(NK))$. We devise a second, computationally more efficient, greedy algorithm in Algorithm 2 with time complexity $O(M^3NK)$. We do not have accuracy guarantees for Algorithm 2, but in the next section, we will show that it performs very well in practice.

Algorithm 2 is based on the following ideas. It starts with empty caches and initializes the cache access delays for users as the miss delays to their closest caches. Then at each step a file is greedily selected to be placed in a cache that maximizes the change in the user access delays, $\sum_i \lambda_i q_{ij}(d_{ij} - \min\{d_{ij}, d^h_{im}\})$. This process continues until the caches are filled. Finally, similar to Algorithm 1, a routing policy that minimizes $D(\mathbf{p}; \mathbf{x})$ is determined.

VII. Performance Evaluation

In this section, we evaluate the performance of the approximate algorithms through discrete-event simulations. Our goal here is to evaluate 1) how well the solutions for the greedy algorithms compare to the optimal solutions (when computing the optimal solution is feasible), and 2) how well solutions from the greedy algorithms compare to those produced by a baseline.

Here, we consider a congestion-sensitive model where the requests over the uncached paths experience a queuing delay

Algorithm 2 Greedy: A greedy approximation without known performance guarantees.

$X_m \leftarrow \emptyset, \forall m$
$X \leftarrow \emptyset$
$d_{ij} \leftarrow \min_m\{d^c_{im}\}, \forall i, j$
for $c \leftarrow 1$ **to** $\sum_m C_m$ **do**
  $G_{jm} \leftarrow [0]_{K \times M}$
  for $m \leftarrow 1$ **to** M **do**
    if $|X_m| < C_m$ **then**
      for $j \leftarrow 1$ **to** K **do**
        $G_{jm} \leftarrow \sum_i \lambda_i q_{ij}(d_{ij} - \min\{d_{ij}, d^h_{im}\})$
  $[j^*, m^*] \leftarrow \arg\max_{j,m} G_{jm}$
  $X_{m^*} \leftarrow X_{m^*} \cup \{s_{j^*m^*}\}$
  $X \leftarrow X \cup \{s_{j^*m^*}\}$
  $d_{ij^*} \leftarrow \min\{d_{ij^*}, d^h_{im^*}\}, \forall i$
Content placement is done according to X.
Determine the routing as $\mathbf{p}^* \leftarrow \arg\min_{\mathbf{p}} D(\mathbf{p}; \mathbf{x}_X)$

modeled as an M/M/1 queue[2], while the requests missed from caches experience a constant delay. For our baseline, we compare the approximate algorithms to the delay obtained by the following algorithm we refer to as *p-LRU*.

A. p-LRU

The cache replacement policy at all caches is Least Recently Used (LRU). For the routing policy, we assume that users that are not connected to any caches forward all their requests to the back-end servers. The remaining users, for each request, use a cached path with probability p and with probability $1-p$ forward the request to the uncached path. If user i decides to use a cached path, she chooses uniformly at random one of the n_i caches she is connected to. The value of p is the same for all users that have access to a cache, and is chosen to minimize the average delay.

First, assuming users equally split their traffic across the caches that they can access, the aggregate popularity for individual files is computed at each cache. Let r^m_j denote the normalized aggregate popularity of file j at cache m. We have

$$r^m_j = \frac{1}{\Lambda} \sum_{i \in \mathcal{I}_m} \lambda_i q_{ij}/n_i,$$

where $\mathcal{I}_m$ denotes the set of users connected to cache m, and Λ is the normalizing constant across all files. Note that r^m_j is independent of the parameter p. With the aggregate popularities at hand, hit probabilities are computed at each cache using the *characteristic time* approximation [21]. Let $\mathbb{P}(x_{jm} = 1)$ denote the probability that file j resides in cache m. From [21] we have

$$\mathbb{P}(x_{jm} = 1) = 1 - \exp(-r^m_j T_m),$$

[2]Note that our analysis is valid for a G/G/1 queue, and the M/M/1 queue is only assumed for evaluation purposes since there is no closed form formula for a G/G/1 queue.

where T_m is the characteristic time of cache m is the unique solution to the equation

$$C_m = \sum_j 1 - \exp\left(-r_j^m T_m\right).$$

Given the cache hit probabilities, the average delay in accessing content j from caches for user i equals

$$d_{ij}^c = \frac{1}{n_i} \sum_{m \in \mathcal{M}_i} \left[\mathbb{P}(x_{jm} = 1) d_{im}^h + (1 - \mathbb{P}(x_{jm} = 1)) d_{im}^c\right],$$

where $\mathcal{M}_i$ denotes the set of caches that user i is connected to. Note that $|\mathcal{M}_i| = n_i$.

Let $\mathcal{I}$ denote the set of users that are connected to at least one cache, and let $\lambda_{\mathcal{I}}$ denote the aggregate request rate of these users. The average delay to access content from caches equals

$$D_c = \frac{1}{\lambda_{\mathcal{I}}} \sum_{i \in \mathcal{I}} \sum_j \lambda_i q_{ij} d_{ij}^c.$$

Remember that some users may not be connected to any caches. Considering the traffic from all users, we can write the overall average delay as

$$D_{\text{LRU}} = \frac{1}{\lambda} \left[p \lambda_{\mathcal{I}} D_c + (1-p) \sum_{i \in \mathcal{I}} \lambda_i d_i^b + \sum_{i \notin \mathcal{I}} \lambda_i d_i^b + \frac{\mu}{\mu - (1-p) \sum_{i \in \mathcal{I}} \lambda_i - \sum_{i \notin \mathcal{I}} \lambda_i} - 1 \right].$$

By differentiating D_{LRU} with respect to p, the optimal value of p is found to be

$$p^* = \max\{0, \min\{1, \left(\sqrt{\frac{\mu \sum_{i \in \mathcal{I}} \lambda_i}{\lambda_{\mathcal{I}} D_c - \sum_{i \in \mathcal{I}} \lambda_i d_i^b}} - \mu + \lambda \right) / \lambda_{\mathcal{I}} \}\}.$$

B. Network Setup

We consider a network with users uniformly distributed in a 2-D square. We consider two architectures. First, we assume there is only one large cache at the center of the network as in Figure 6a. Second, we consider a network with five small caches with equal storage capacities as in Figure 6b. Figure 6 also shows the communication range of the caches in each case. In the single-cache network, the cache has a larger communication range and five times the capacity of each of the caches in the multi-cache network.

Users that are not in communication range of any caches can only use the uncached path to the back-end server. The hit delay for each user is linearly proportional to the distance from the cache and has the maximum value[3] of 12.5 time units and 5.5 time units for the single and multi-cache systems, respectively. For a cache miss, an additional delay of 25 time units is added to the hit delay. The initial access delay of the uncached path is set to five time units for each user, and the service rate is proportional to the aggregate request rate, where the scaling factor will be specified later.

[3]Here, delay aggregates all request propagation and download delays as well as the processing and queuing delays. We use normalized delay values instead of using any specific time unit.

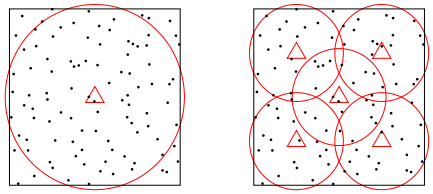

Fig. 6: A network with (a) one cache, and (b) five caches.

C. Numerical Evaluation

1) GreedyWG vs. Optimal: First, we compare the solution of GreedyWG the approximate algorithm in Algorithm 1 to the optimal solution. Due to the exponential complexity of finding the optimal solution, we are only able to compute the optimal solution for small problem instances. Here, we consider a network with five users and a single cache. User request rates are arbitrarily set to satisfy $\sum_i \lambda_i = 5$. We assume users are interested in 15 files, and that the aggregate user request popularities follow a Zipf distribution with skewness parameter 0.6. The service rate of the back-end server is $\mu = 1$.

Figure 7 shows the average delay and the 95% confidence interval over 100 runs of each algorithm. It is clear that GreedyWG performs very close to optimal. In fact, we observe that GreedyWG differs from the optimal solution less than 20% of the time, and the relative difference is never more than 1%.

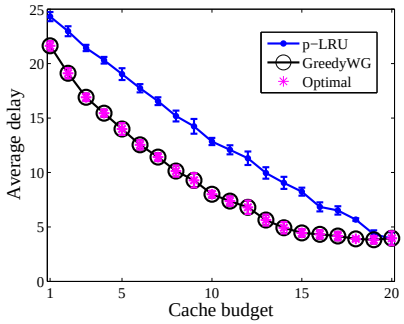

Fig. 7: Evaluation of GreedyWG against Optimal and p-LRU.

2) GreedyWG vs. Greedy: Next, we compare the solutions of GreedyWG against those of Greedy, the approximation algorithm, Algorithm 2, with lower computational complexity but no performance guarantees. We consider a network with five caches and 100 users uniformly distributed in a 10×10 field.

Figure 8 shows the average delay and the 95% confidence interval for different values of available cache budget. Greedy (red curve) is barely distinguishable from GreedyWG (black curve), meaning that Greedy performs very close to GreedyWG.

We also evaluate these algorithms over different values of the service rate at the back-end server. Figure 9 shows the average delay for μ between 2 to 7, with the aggregate traffic

rate set to $\lambda = 5$. Similar to Figure 8, Greedy performs very close to GreedyWG, and is always within 1% of GreedyWG.

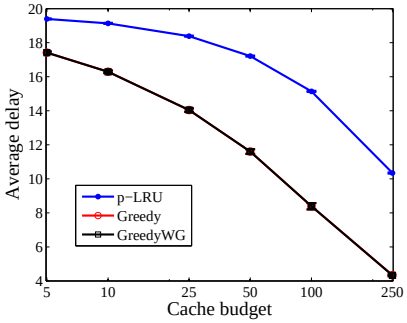

Fig. 8: Evaluation of the two greedy approximations over different values of the cache budget split equally between five caches. Aggregate user request rate is $\lambda = 5$, and service rate of the back-end server equals 2.5.

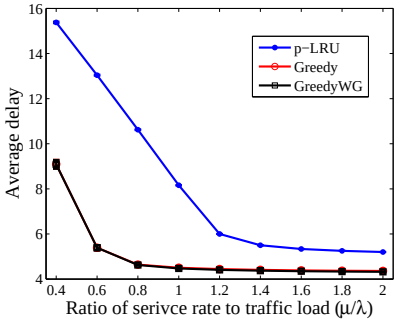

Fig. 9: Evaluation of the greedy algorithms for different values of the service rate at the back-end server. Aggregate user request rate is $\lambda = 5$, and the service rate varies from 2 to 10. Cache budget is set to 125.

D. Trace-driven Simulation

Here, we present trace-driven evaluation results where we use traces for web accesses collected from a gateway router at IBM research lab [22]. The trace consists of approximately 9 million requests generated for more around 3.3 million distinct files over a period of five hours. We only consider Greedy, the greedy algorithm presented in Algorithm 2, since it performs nearly as well as Algorithm 1, and has lower complexity.

The access delay to each cache equals one-tenth of the distance from the cache in case of a cache hit. For a cache miss, an additional delay of 25ms is added to the hit delay. The initial access delay of the uncached path is set to 5ms for each user, and the service rate is proportional to the aggregate request rate, where the scaling factor will be specified later.

To evaluate the Greedy algorithm using the trace data, we first divide the trace into smaller segments of approximately 120,000 requests. Each segment includes requests for approximately 40,000 distinct files, generated by approximately 2500 users. To simulate requests from the ith segment, we first compute the file popularities using the $(i-1)$st segment, and compute the optimal value of p for the p-LRU algorithm.

Figure 10 compares the average delays for different cache budgets for the p-LRU and the Greedy algorithms for the single-cache (S) and multi-cache (M) networks. Significant reductions in average delay of up to 50% are observed for both

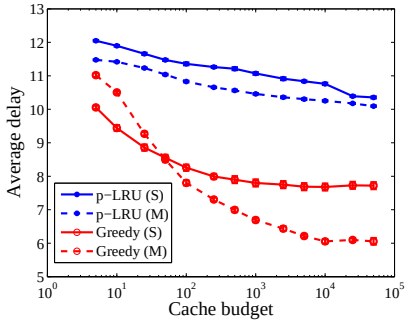

Fig. 10: Evaluation of the Greedy and p-LRU for the single-cache (S) and multi-cache (M) network setups for different values of the available cache budget. The service rate is set to be 0.8 times the aggregate traffic rate.

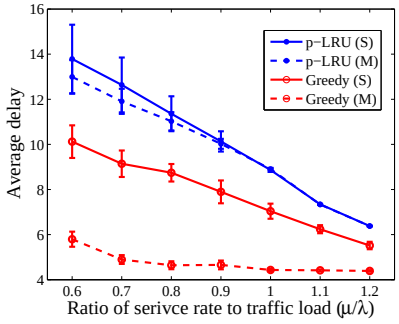

Fig. 11: Evaluation of the Greedy and p-LRU algorithms for different values of the service rate to aggregate traffic ratio for the single-cache (S) and multi-cache (M) network setups.

single-cache and multi-cache networks when using Greedy over p-LRU. While p-LRU yields similar performance in both single-cache and multi-cache architectures, Greedy shows the advantage of one architecture over the other depending on the cache budget. When the cache budget is small, it is better to have a single cache with larger cache size and coverage so that more users can access popular files from the cache; when the cache budget is large, it is better to have multiple caches, each with smaller size and coverage, so that users can access files from nearby caches with smaller hit delays.

We also evaluate the algorithms for different values of the service rate of the uncached path assuming the cache budget is fixed at $10,000$. Figure 11 shows the average delay when the ratio of service rate to the total request rate changes from 0.6 to 1.2. Similar to Figure 10, the Greedy algorithm significantly reduces the average content access delay. Again, the cache architecture makes little difference for p-LRU but significant affect to the performance of the Greedy algorithm. Moreover, the difference decreases as the service rate on the uncached path increases, as more traffic is offloaded to the uncached path.

VIII. Related Work

Benefits of content caching have been theoretically analyzed [6], [23]–[26]. [6], [26] demonstrate that the asymptotic throughput capacity of a network is significantly increased by adding caching capabilities to the nodes. In this paper, we have considered the *joint* routing and cache-content management problems. Numerous past research efforts have considered these problems separately. The problem of content placement

in caches, has received significant attention in the Internet, in hybrid networks such as those considered in this paper, and in sensor networks [4], [5], [8], [9], [27]. Baev *et al.* [8] prove that the problem of content placement with the objective of minimizing the access delay is NP-complete, and present approximate algorithms. More recently, Giovanidis *et al.* [28] introduced multi-LRU, a family of decentralized caching policies, that extends the classical LRU policy to cases where objects can be retrieved from more than one cache. The separate problem of efficient routing in cache networks has also been explored in the literature [29], [30]. Cache-aware routing schemes that calculate paths with minimum transportation costs based on given caching policy and request demand have been proposed in [25].

The joint caching and routing problem, with the objective of minimizing content access delay, has recently been studied in [4], [5], where the authors consider a hybrid network consisting of multiple femtocell caches and a cellular infrastructure. Both papers assume that users greedily choose the minimum delay path to access content, *i.e.*, requests for cached content are routed to caches (where content is know to reside), whereas remaining requests are routed to the (uncached) cellular network. They assume that the delays are constant and independent of the request rate.

Our work differs from much of the previous research discussed above by considering a joint caching and routing problem, where we determine the optimal routes users should take for accessing content as well as the optimal caching policy. Our research differs from [4], [5] in that we consider heterogeneous delays between users and caches, consider a congestion-insensitive delay model for the uncached path as well as a congestion-sensitive model, investigate the problem's time complexity, and propose bounded approximate solutions for both congestion-insensitive and congestion-sensitive scenarios. We also determine scenarios for which the optimal solution can be found in polynomial time for the congestion-insensitive delay model, and ascertain the root cause of the complexity of the general problem.

Algorithms for joint caching and routing schemes were previously proposed in [31] and [32] based on the primal-dual method. These algorithms are based on the Lagrangian relaxation method and rely on iterative algorithms to reach a solution with certain optimality criteria. As such, there is no efficiency guarantee on the results nor the running time of these algorithms. In contrast, our proposed approximation algorithms require fixed running time, and are guaranteed to be withing a factor $1 - 1/e$ of the optimal solution.

IX. Conclusion

In this paper, we have considered the problem of joint content placement and routing in heterogeneous networks that support in-network caching but also provide a separate (uncached) path to a back-end content server; we considered cases in which paths to the back-end server were modeled as congestion-insensitive, constant-delay paths, and congestion-sensitive paths modeled by a convex delay rate function. We provided fundamental complexity results showing that the problem of joint caching and routing is NP-complete in both cases, developed a greedy algorithm with guaranteed performance of $(1 - 1/e)$ of the optimal solution as well as a lower complexity heuristic that was empirically found to provide average delay performance that was within 1% of optimal (for small instances of the problem) and that significantly reduce the average content access delay over the case of optimized traditional LRU caching. Our investigation of special-case scenarios – the congestion-insensitive multiple-cache single-file-of-interest case (where we demonstrated an optimal polynomial time solution) and the congestion-sensitive single-cache single-file-of-interest case (which we demonstrated remained NP-complete) – helped illuminate what makes the problem "hard" in general. Our future work is aimed at developing distributed algorithms for content placement and routing, and on developing solutions for the case of time-varying content popularity.

Acknowledgment

This work was supported in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001 and the National Science Foundation under Grant No. CNS-1413998 and CNS-1117764. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] V. Chandrasekhar, J. Andrews, and A. Gatherer, "Femtocell networks: a survey," *IEEE Communications Magazine*, vol. 46, no. 9, pp. 59–67, September 2008.

[2] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid cdn-p2p: Why limelight needs its own red swoosh," in *NOSSDAV*, May 2008, pp. 75–80.

[3] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing content simplifies isp traffic engineering," in *SIGMETRICS*, June 2013, pp. 229–242.

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

[5] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation caching and routing algorithms for massive mobile data delivery," in *Globecom*, 2013.

[6] B. Azimdoost, C. Westphal, and H. Sadjadpour, "On the throughput capacity of information-centric networks," in *ITC*, 2013, pp. 1–9.

[7] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *INFOCOM*, 2012.

[8] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM Journal on Computing*, vol. 38, no. 4, pp. 1411–1429, 2008.

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

[10] IXIA White paper, "Quality of service (qos) and policy management in mobile data networks," *915-2731-01 Rev. D*, December 2013.

[11] Z. Liu, N. Nain, P Niclausse, and D. Towsley, "Static caching of web servers," in *Multimedia Computing and Networking Conference*, 1998.

[12] M. Dehghan, A. Seetharam, T. He, T. Salonidis, J. Kurose, and D. Towsley, "Optimal caching and routing in hybrid networks," in *IEEE MILCOM 2014*, October 2014, pp. 1072–1078.
[13] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *INFOCOM*, 2015.
[14] M. Taghizadeh, K. Micinski, C. Ofria, E. Torng, and S. Biswas, "Distributed cooperative caching in social wireless networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1037–1053, 2013.
[15] D. B. West, *Introduction to graph theory (2nd ed.)*. Chapter 3, Prentice Hall, 2001.
[16] A. J. Hoffman and J. B. Kruskal, "Integral boundary points of convex polyhedra," in *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 49–76.
[17] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003, vol. 24.
[18] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*. Wiley New York, 1988, vol. 18.
[19] R. G. Bartle, *The elements of real analysis (2nd ed.)*. Wiley New York, 1976.
[20] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a submodular set function subject to a matroid constraint (extended abstract)," in *IPCO*, 2007, pp. 182–196.
[21] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *INFOCOM*, 2001.
[22] P. Zerfos, M. Srivatsa, H. Yu, D. Dennerline, H. Franke, and D. Agrawal, "Platform and applications for massive-scale streaming network analytics," *IBM Journal for Research and Development: Special Edition on Massive Scale Analytics*, vol. 57, pp. 1–11, May 2013.
[23] E. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *IEEE INFOCOM*, 2010.
[24] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, 2013.
[25] V. Sourlas, P. Flegkas, and L. Tassiulas, "Cache-aware routing in information-centric networks," in *IFIP/IEEE International Symposium on Integrated Network Management*, 2013.
[26] S. Gitzenis, G. Paschos, and L. Tassiulas, "Asymptotic laws for joint content replication and delivery in wireless networks," *IEEE Transactions on Information Theory*, vol. 59, no. 5, pp. 2760–2776, May 2013.
[27] B. Tang and H. Gupta, "Cache placement in sensor networks under an update cost constraint," *Journal of Discrete Algorithms*, vol. 5, no. 3, pp. 422–435, 2007.
[28] A. Giovanidis and A. Avranas, "Spatial multi-lru caching for wireless networks with coverage overlaps," in *SIGMETRICS*. ACM, 2016, pp. 403–405.
[29] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *IFIP Networking*, 2012.
[30] I. Psaras, W. Chai, and G. Pavlou, "Coordinating in-network caching in content-centric networks: Model and analysis," in *In ACM SIGCOMM Workshop on Information-Centric Networking*, 2012.
[31] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi, "Orchestrating massively distributed cdns," in *CoNEXT*, 2012, pp. 133–144.
[32] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *INFOCOM*, March 2012, pp. 2444–2452.
[33] S. E. A. P. Cieliebak, Mark and K. Schlude, "On the complexity of variations of equal sum subsets," *Nordic Journal of Computing*, vol. 14, no. 3, pp. 151–172, 2008.
[34] K. Fridgeirsdottir and S. Chiu, "A note on convexity of the expected delay cost in single-server queues," *Operations research*, vol. 53, no. 3, pp. 568–570, 2005.

APPENDIX A
NETWORK WITH TWO CACHES

Proof. Consider the highlighted elements of the matrix in Figure 12, and let r_1 denote the first row of the matrix. Also, let r_2 and r_3 denote the first two rows below the second horizontal line. It is easy to see that if these three rows are selected to be in R, any assignment satisfying Proposition 2 should have $-s(r_1) = s(r_2) = s(r_3)$. Otherwise, the signed

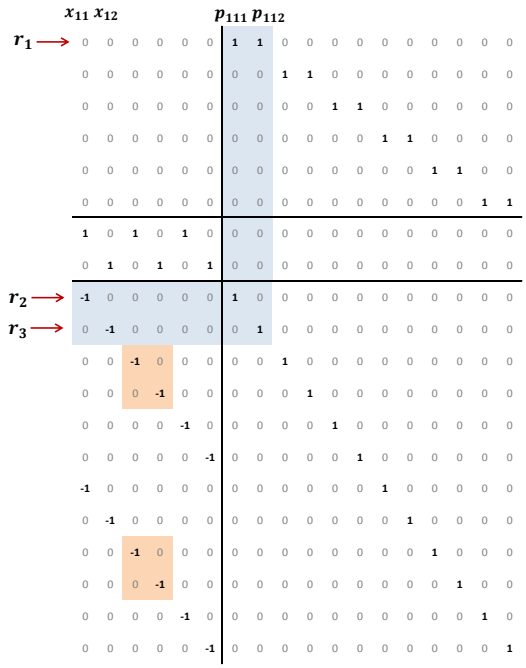

Fig. 12: An example of the constraints matrix A for a network with two caches, two users and three files

sum of the rows will have entries other than $\{0, \pm 1\}$. This observation can be easily extended to see that rows below the second horizontal line can be considered in groups of two such that if the two rows are selected to be in R they will be assigned the same sign.

We sign the rows in R starting from the rows below the second horizontal line. Considering the groups of two rows, we make assignments such that the elements to the left of the vertical line of the signed sum of the rows are in $\{0, -1\}$ only. To see why this is possible, note that the non-zero elements of the matrix to the left of the vertical line can be seen as small blocks of 2×2 matrices. It is easy to see that the signed sum of any subset of these blocks can be made to have elements only in $\{0, -1\}$, with rows in the same group getting the same assignment. The rows between the two horizontal lines are always signed $+1$. The sign of the rows above the first horizontal line follows the assignment of the lines below the second horizontal line based on the previous discussion.

With the above procedure, the sum of the signed vectors will have entries in $\{0, \pm 1\}$ for any set of rows R, and from Proposition 2 it follows that the matrix A is totally unimodular, and hence the solution for the optimization problem in (1) for a network with two caches can be found in polynomial time. □

APPENDIX B
PROOF OF LEMMA 1

Proof. A proof of NP-hardness of a more general form of ECP is given in [33]. Here, we give a simpler proof by a reduction from the Partition problem.

Problem 3. *(Partition) Given a set A of n positive integers, can A be partitioned into two disjoint subsets A_1 and A_2 such*

that $A = A_1 \cup A_2$ and the sum of the numbers in A_1 equals the sum of the numbers in A_2?

For each instance of Partition with input $A = \{a_1, \ldots, a_n\}$ create an instance $A' = \{a_1, \ldots, a_n, 0, \ldots, 0\}$ by adding n zeros to A. It is easy to see that A' can be partitioned into two subsets with equal cardinality if and only if A can be partitioned. Therefore, Partition $\leq_P$ ECP, and ECP is NP-hard. □

APPENDIX C
PROOF OF THEOREM 4

Proof. It is easy to see that given some $\mathbf{x}, \mathbf{p}$ the expected delay $D(\mathbf{x}, \mathbf{p})$ can be computed in polynomial time, and hence CSDDP is in NP. To show it is NP-hard, we reduce the problem of Equal Cardinality Partition (ECP) to our problem. For an instance of the ECP(A) problem we create the instance CSDDP($S, A, [\frac{4}{S}], [+\infty], [\frac{4}{a_i}], \frac{n}{2}$) where $S = \sum_{a_i \in A} a_i$.

Now, the set A can be partitioned into subsets A_1 and A_2 with $|A_1| = |A_2|$ if and only if CSDDP achieves delay $(2n+3)/S$.

To see more clearly why the reduction works, first note that with the delay values being set to $d_i^h = 4/S$ and $d_i^b = 4/a_i$, since $d_i^h < d_i^b$ if a file exists in the cache all the requests for that file will be directed to the cache. Also, since $d_i^c = +\infty$, if a file is not in the cache all the requests for that file will be requested from the back-end server. Therefore, we have $p_i = x_i, \forall i$. Now, with the service rate set to $\mu = S$, we can re-write the optimization problem in (2) as follows

$$\begin{aligned} \text{minimize } & \frac{1}{S}\left[\frac{4}{S}\sum_{i=1}^{n} a_i x_i + 4\sum_{i=1}^{n}(1-x_i) + \frac{S}{\sum_{i=1}^{n} a_i x_i} - 1\right] \\ \text{such that } & \sum_{i=1}^{n} x_i \leq \frac{n}{2} \\ & x_i \in \{0,1\} \end{aligned}$$

Now, looking at the objective function in the above problem, we can see that

$$Z_1 = 4\sum_{i=1}^{n}(1-x_i) \geq 2n$$

since we should have $\sum_{i=1}^{n} x_i \leq n/2$. Moreover, $Z_1 = 2n$ if $\sum_{i=1}^{n} x_i = n/2$ meaning that exactly half of the files are in the cache. We also have that

$$Z_2 = \frac{4}{S}\sum_{i=1}^{n} a_i x_i + \frac{S}{\sum_{i=1}^{n} a_i x_i} \geq 4,$$

and $Z_2 = 4$ only if $\sum_{i=1}^{n} a_i x_i = S/2$.

Hence, $Z_1 + Z_2 - 1 = 2n+3$ if and only if $\sum_{i=1}^{n} a_i x_i = S/2$ and $\sum_{i=1}^{n} x_i = n/2$.

Therefore, if CSDDP($S, A, [\frac{4}{S}], [+\infty], [\frac{4}{a_i}], \frac{n}{2}$) achieves minimum delay $(2n+3)/S$ then A can be partitioned into equal cardinality subsets.

It is easy to see that if A can be partitioned into two subsets of equal cardinality, then CSDDP($S, A, [\frac{4}{S}], [+\infty], [\frac{4}{a_i}], \frac{n}{2}$) has minimum delay of $(2n+3)/S$. □

APPENDIX D
PROOF OF LEMMA 3

Here, we will first prove the lemma for the more general case of convex delay rate function, and as an example consider G/G/1 queues. Given a cache configuration X, let

$$(\mathbf{d}_X^h)_{ij} = \inf\left\{d_{im}^h : X_{jm} = 1\right\},$$
$$(\mathbf{d}_X^c)_{ij} = \inf\left\{d_{im}^c : X_{jm} = 0\right\},$$

denote the minimum cache hit and miss delays for user i accessing file j, with the convention $\inf \emptyset = +\infty$. We assume that $d_{im}^h \leq d_{im}^c$ and hence $0 \leq \mathbf{d}_X^h \leq \mathbf{d}_X^c \leq +\infty$.

Define $(\mathbf{\Lambda})_{ij} = \lambda_i q_{ij}$, and let $(\boldsymbol{\lambda}_X^h)_{ij}$, $(\boldsymbol{\lambda}_X^c)_{ij}$, $(\boldsymbol{\lambda}_X^b)_{ij}$ denote the rate of requests for file j sent by user i that are routed through caches containing file j, routed through caches without file j, and directly routed to the back-end server, respectively. We suppress the subscript X when no confusion arises.

The average delay can be written as

$$D(\boldsymbol{\lambda}^h, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^b) = \boldsymbol{\lambda}^h \cdot \mathbf{d}^h + \boldsymbol{\lambda}^c \cdot \mathbf{d}^c + f(\lambda^c) + g(\lambda^b),$$

where[4] $\lambda^c = \boldsymbol{\lambda}^c \cdot \mathbf{1}$ and $\lambda^b = \boldsymbol{\lambda}^b \cdot \mathbf{1}$, and $f(\cdot)$ and $g(\cdot)$ denote the *total expected delay cost rate* for the G/G/1 queues representing the paths from caches to the back-end server, and direct paths from users to back-end servers, respectively. The total expected delay cost rate function for G/G/1 queues is proved to be convex in [34].

Now we have the following optimization problem,

$$\begin{aligned} \text{minimize} \quad & D(\boldsymbol{\lambda}^h, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^b) \\ \text{subject to} \quad & \boldsymbol{\lambda}^h, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^b \geq \mathbf{0}, \\ & \boldsymbol{\lambda}^h + \boldsymbol{\lambda}^c + \boldsymbol{\lambda}^b = \mathbf{\Lambda}. \end{aligned} \tag{3}$$

Since Slater's condition holds, the optimal delay $D^\star$ can be found by solving the dual optimization problem. For notational simplicity, we first assume that $\mathbf{d}^h$ and $\mathbf{d}^c$ are finite. This assumption can be easily removed by setting to zero the components of $\boldsymbol{\lambda}^h$ and $\boldsymbol{\lambda}^c$ corresponding to the infinite components of $\mathbf{d}^h$ and $\mathbf{d}^c$, which simply reduces the number of decision variables.

The Lagrangian for (3)

$$\begin{aligned} & L(\boldsymbol{\lambda}^h, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^b, \boldsymbol{\nu}, \boldsymbol{\xi}^h, \boldsymbol{\xi}^c, \boldsymbol{\xi}^b) \\ &= D + \boldsymbol{\nu} \cdot (\mathbf{\Lambda} - \boldsymbol{\lambda}^h - \boldsymbol{\lambda}^c - \boldsymbol{\lambda}^b) - \boldsymbol{\xi}^h \cdot \boldsymbol{\lambda}^h - \boldsymbol{\xi}^c \cdot \boldsymbol{\lambda}^c - \boldsymbol{\xi}^b \cdot \boldsymbol{\lambda}^b \\ &= \boldsymbol{\nu} \cdot \mathbf{\Lambda} - \boldsymbol{\eta}^c \cdot \boldsymbol{\lambda}^h - \boldsymbol{\eta}^u \cdot \boldsymbol{\lambda}^c - \boldsymbol{\eta}^s \cdot \boldsymbol{\lambda}^b + f(\lambda^c) + g(\lambda^b), \end{aligned}$$

where $\boldsymbol{\eta}^h = \boldsymbol{\nu} + \boldsymbol{\xi}^h - \mathbf{d}^h$, $\boldsymbol{\eta}^c = \boldsymbol{\nu} + \boldsymbol{\xi}^c - \mathbf{d}^c$ and $\boldsymbol{\eta}^b = \boldsymbol{\nu} + \boldsymbol{\xi}^b$. The dual function is

$$\begin{aligned} \hat{D}(\boldsymbol{\nu}, \boldsymbol{\xi}^h, \boldsymbol{\xi}^c, \boldsymbol{\xi}^b) &= \inf_{\boldsymbol{\lambda}^h, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^h} L(\boldsymbol{\lambda}^h, \boldsymbol{\lambda}^c, \boldsymbol{\lambda}^b, \boldsymbol{\nu}, \boldsymbol{\xi}^h, \boldsymbol{\xi}^c, \boldsymbol{\xi}^b) \\ &= \inf_{\boldsymbol{\lambda}^h}(\boldsymbol{\nu} \cdot \mathbf{\Lambda} - \boldsymbol{\eta}^h \cdot \boldsymbol{\lambda}^h) + \inf_{\boldsymbol{\lambda}^c}[f(\lambda^c) - \boldsymbol{\eta}^c \cdot \boldsymbol{\lambda}^c] \\ &\quad + \inf_{\boldsymbol{\lambda}^b}[g(\lambda^b) - \boldsymbol{\eta}^b \cdot \boldsymbol{\lambda}^b]. \end{aligned}$$

[4]The dot product is defined by $\mathbf{a} \cdot \mathbf{b} = \sum_{ij} a_{ij} b_{ij}$.

For $\hat{D} > -\infty$, we need $\boldsymbol{\eta}^h = \mathbf{0}$, $\boldsymbol{\eta}^c \geq \mathbf{0}$, $\boldsymbol{\eta}^b \geq \mathbf{0}$. When this condition is met,

$$\hat{D}(\boldsymbol{\nu}, \boldsymbol{\xi}^h, \boldsymbol{\xi}^c, \boldsymbol{\xi}^b) = \boldsymbol{\nu} \cdot \boldsymbol{\Lambda} - f^*(\max(\boldsymbol{\eta}^c)) - g^*(\max(\boldsymbol{\eta}^b)),$$

where f^* and g^* are the conjugates of f and g respectively, defined by

$$f^*(y) = \sup_x [xy - f(x)], \quad g^*(y) = \sup_x [xy - g(x)].$$

Thus the dual problem becomes

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{\nu} \cdot \boldsymbol{\Lambda} - f^*(\max(\boldsymbol{\eta}^c)) - g^*(\max(\boldsymbol{\eta}^b)) \\ \text{subject to} \quad & \boldsymbol{\eta}^c, \boldsymbol{\eta}^b \geq \mathbf{0}, \\ & \mathbf{d}^h - \boldsymbol{\nu} \geq \mathbf{0}, \\ & \boldsymbol{\eta}^c + \mathbf{d}^c - \boldsymbol{\nu} \geq \mathbf{0}, \\ & \boldsymbol{\eta}^b - \boldsymbol{\nu} \geq \mathbf{0}. \end{aligned}$$

or

$$\begin{aligned} \text{maximize} \quad & \tilde{D}(s,u) = \mathbf{m}(s,u) \cdot \boldsymbol{\Lambda} - f^*(u) - g^*(s) \\ \text{subject to} \quad & s, u \geq 0, \end{aligned} \tag{4}$$

where $\mathbf{m}(s,u) = \mathbf{d}^h \wedge (s\mathbf{1}) \wedge (\mathbf{d}^c + u\mathbf{1})$ with $\wedge$ denoting component-wise minimum.

Now we make explicit the dependence on cache configuration. Given the cache configuration X, let $s_X^\star$ and $u_X^\star$ be an optimal solution of the dual problem (4) and $D_X^\star$ the optimal delay.

Let X, Y be two cache configurations. We want to show

$$D_X^\star + D_Y^\star \leq D_{X\cup Y}^\star + D_{X\cap Y}^\star. \tag{5}$$

Assume that $X \setminus Y \neq \emptyset$ and $Y \setminus X \neq \emptyset$; otherwise, (5) holds trivially. Without loss of generality, assume $u_X^\star \leq u_Y^\star$. Then

$$\begin{aligned} & D_X^\star + D_Y^\star - D_{X\cup Y}^\star - D_{X\cap Y}^\star \\ &= \tilde{D}_X(s_X^\star, u_X^\star) + \tilde{D}_Y(s_Y^\star, u_Y^\star) \\ &\quad - \tilde{D}_{X\cup Y}(s_{X\cup Y}^\star, u_{X\cup Y}^\star) - \tilde{D}_{X\cap Y}(s_{X\cap Y}^\star, u_{X\cap Y}^\star) \\ &\leq \tilde{D}_X(s_X^\star, u_X^\star) + \tilde{D}_Y(s_Y^\star, u_Y^\star) \\ &\quad - \tilde{D}_{X\cup Y}(s_X^\star \wedge s_Y^\star, u_X^\star) - \tilde{D}_{X\cap Y}(s_X^\star \vee s_Y^\star, u_Y^\star) \\ &= \boldsymbol{\Lambda} \cdot [\mathbf{m}_X(s_X^\star, u_X^\star) + \mathbf{m}_Y(s_Y^\star, u_Y^\star) \\ &\quad - \mathbf{m}_{X\cup Y}(s_X^\star \wedge s_Y^\star, u_X^\star) - \mathbf{m}_{X\cap Y}(s_X^\star \vee s_Y^\star, u_Y^\star)] \\ &= \boldsymbol{\Lambda} \cdot [\boldsymbol{\Delta}_1 - \boldsymbol{\Delta}_2], \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{\Delta}_1 &= \mathbf{m}_X(s_X^\star, u_X^\star) - \mathbf{m}_{X\cup Y}(s_X^\star \wedge s_Y^\star, u_X^\star), \\ \boldsymbol{\Delta}_2 &= \mathbf{m}_{X\cap Y}(s_X^\star \vee s_Y^\star, u_Y^\star) - \mathbf{m}_Y(s_Y^\star, u_Y^\star). \end{aligned}$$

Define $h(x,y,z) = x \wedge y - x \wedge z$, which is nonnegative and increasing in x for $y \geq z$. Now consider two cases,

1) $s_X^\star \geq s_Y^\star$.

$$\begin{aligned} \boldsymbol{\Delta}_1 &= \mathbf{d}_X^h \wedge (\mathbf{d}_{\bar{X}}^c + u_X^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}) \\ &\quad - \mathbf{d}_{X\cup Y}^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_X^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}) \\ &= h\Big(\mathbf{d}_X^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_X^\star \mathbf{1}), \\ &\qquad (\mathbf{d}_{Y\setminus X}^c + u_X^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}), \mathbf{d}_{Y\setminus X}^h \wedge (s_Y^\star \mathbf{1})\Big) \\ &\leq h\Big(\mathbf{d}_X^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_Y^\star \mathbf{1}), \\ &\qquad (\mathbf{d}_{Y\setminus X}^c + u_X^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}), \mathbf{d}_{Y\setminus X}^h \wedge (s_Y^\star \mathbf{1})\Big), \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{\Delta}_2 &= \mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\overline{X\cap Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}) \\ &\quad - \mathbf{d}_Y^h \wedge (\mathbf{d}_{\bar{Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}) \\ &= h\Big(\mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\bar{Y}}^c + u_Y^\star \mathbf{1}), \\ &\qquad (\mathbf{d}_{Y\setminus X}^c + u_Y^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}), \mathbf{d}_{Y\setminus X}^h \wedge (s_Y^\star \mathbf{1})\Big). \end{aligned}$$

Since

$$\begin{aligned} & \mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\bar{Y}}^c + u_Y^\star \mathbf{1}) - \mathbf{d}_X^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_Y^\star \mathbf{1}) \\ &= h\Big(\mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_Y^\star \mathbf{1}), \mathbf{d}_{X\setminus Y}^c + u_Y^\star \mathbf{1}, \mathbf{d}_{X\setminus Y}^h\Big) \geq \mathbf{0}, \end{aligned}$$

it follows that $\boldsymbol{\Delta}_1 \leq \boldsymbol{\Delta}_2$.

2) $s_X^\star \leq s_Y^\star$.

$$\begin{aligned} \boldsymbol{\Delta}_1 &= \mathbf{d}_X^h \wedge (\mathbf{d}_{\bar{X}}^c + u_X^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}) \\ &\quad - \mathbf{d}_{X\cup Y}^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_X^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}), \\ &= h\Big(\mathbf{d}_X^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_X^\star \mathbf{1}) \wedge (s_X^\star \mathbf{1}), \\ &\qquad \mathbf{d}_{Y\setminus X}^c + u_Y^\star \mathbf{1}, \mathbf{d}_{Y\setminus X}^h\Big) \\ &\leq h\Big(\mathbf{d}_X^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}), \\ &\qquad \mathbf{d}_{Y\setminus X}^c + u_Y^\star \mathbf{1}, \mathbf{d}_{Y\setminus X}^h\Big), \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{\Delta}_2 &= \mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\overline{X\cap Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}) \\ &\quad - \mathbf{d}_Y^h \wedge (\mathbf{d}_{\bar{Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}) \\ &= h\Big(\mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\bar{Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}), \\ &\qquad \mathbf{d}_{Y\setminus X}^c + u_Y^\star \mathbf{1}, \mathbf{d}_{Y\setminus X}^h\Big). \end{aligned}$$

Since

$$\mathbf{d}_X^h \wedge (\mathbf{d}_{\overline{X\cup Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}) \leq \mathbf{d}_{X\cap Y}^h \wedge (\mathbf{d}_{\bar{Y}}^c + u_Y^\star \mathbf{1}) \wedge (s_Y^\star \mathbf{1}),$$

it follows that $\boldsymbol{\Delta}_1 \leq \boldsymbol{\Delta}_2$.

In both cases, (5) holds and hence $D^\star$ is supermodular, and $D_\emptyset - D^\star$ is submodular.