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

Mostafa Dehghan[1], Anand Seetharam[2], Bo Jiang[1], Ting He[3], Theodoros Salonidis[3],
Jim Kurose[1], Don Towsley[1], and Ramesh Sitaraman[1,4]
[1]University of Massachusetts Amherst, [2]California State University Monterey Bay,
[3]IBM T.J. Watson Research Center, [4]Akamai Technologies Inc
{mdehghan, bjiang, kurose, towsley, ramesh}@cs.umass.edu,
aseetharam@csumb.edu, {the, tsaloni}@us.ibm.com

***Abstract*—We investigate the problem of optimal request routing and content caching in a heterogeneous network supporting in-network content caching with the goal of minimizing average content access delay. Here, 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 a piece of content, a user must decide whether to route its request to a cache or to the back-end server. Additionally, caches must decide which content to cache. We investigate the problem complexity of two problem formulations, where the direct path to the back-end server is modeled as *i)* a congestion-sensitive or *ii)* a congestion-insensitive path, reflecting whether or not the delay of the uncached path to the back-end server depends on the user request load, respectively. We show that the problem is NP-complete in both cases. We prove that under the congestion-insensitive 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 a structural property of the user-cache graph that potentially makes the problem NP-complete. For the congestion-sensitive 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 models within a $(1 - 1/e)$ factor of the optimal solution, and demonstrate a greedy algorithm that is found to be within 1% of optimal for small problem sizes. Through trace-driven simulations we evaluate the performance of our greedy algorithms, which show up to a 50% reduction in average delay over solutions based on LRU content caching.**

I. Introduction

In-network content caching has received considerable attention in recent years as a means to address the explosive growth in data access seen in today's networks. Its main premise is to store content at the network's edge – close to the end users – to reduce user content access delay and network bandwidth usage. The benefits of in-network content caching have been demonstrated in the context of CDN [1]–[3] as well as hybrid networks comprised of cellular and MANETs or femto-cell networks [4]–[6].

In this paper, we investigate a *joint* problem of in-network content caching and request routing in a hybrid network where stored content can be accessed through multiple heterogeneous network paths. We consider a scenario in which users send requests for content that is always available at a remote back-end server located in the network core but may also be present at multiple in-network caches. Access to the back-end server employs a potentially costly, congested, and/or slower uncached path, while the in-network caches may be reached through cheaper and faster network paths. This scenario arises in various hybrid network contexts where content can be accessed through multiple heterogeneous paths, including core/edge CDNs, macro/femto cell networks, cellular/MANET networks (*e.g.*, where the path to the network core is over cellular infrastructure and in-network caches are accessible via MANET paths), and cloud/edge cellular networks with edge storage at the cellular base stations. If a request is routed to an in-network cache that has the requested 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 additional delay. Additionally, the cache must decide whether or not to store the downloaded content.

We address the following question: how should users route their requests among the paths to in-network caches and the back-end server, and what in-network cache management policy should be adopted to minimize the average content access delay across all users? We consider two variants of the problem. First, we consider a *congestion-insensitive* delay model (termed *CI-model*), assuming that delays are independent of the traffic load on all paths. Second, we consider a *congestion-sensitive* delay model (termed *CS-model*), assuming that the delay to the back-end server (*i.e.*, the uncached path) depends on the traffic load. In a hybrid cellular/MANET network, the uncached path in the CI-model corresponds to GBR (guaranteed bit rate) 3GPP bearer service, while in the CS-model it corresponds to Non-GBR Aggregate Maximum Bit Rate (AMBR) bearer service [7].

Our goal in this paper is two-pronged. First, we seek a principled understanding of the computational complexity of

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.

the joint caching and routing problem: i) Can the general problem be solved optimally in polynomial time? ii) If not, are there problem instances that are tractable and which of the above modeling aspects make the general problem intractable? Second, we seek efficient approximate solutions to the joint routing/caching problem, with approximation guarantees, that work well in practice.

Toward our first goal, we provide a unified optimization formulation of the joint caching and routing problem for both models and show this problem is NP-complete in the general case. Then, we investigate which factors contribute to the problem complexity. For the CI-model, we prove that the optimal solution can be found in polynomial time in two special cases: a) when each user requests a single piece of content, or b) when there are at most two in-network caches. We also identify a condition which is potentially the root cause of the complexity of the problem in the general case – cycles with an odd number of users and caches in a graph that represents the network. For the CS-model, we prove that the problem is "harder": it remains NP-complete even if there is a single cache and each user accesses a single distinct content. These results provide valuable insights on the problem complexity.

Toward our second goal, we show that the problem of optimal joint caching and routing for both the CI-model and the CS-model can be formulated as maximization of a monotone submodular function subject to matroid constraints. This enables us to devise two greedy algorithms. The first one has a higher complexity but can produce solutions within a $(1-1/e)$ factor of the optimal solution for both the CI-model and CS-model. The second algorithm has a lower complexity but does not have known approximation guarantees. We evaluate the performance of these algorithms through numerical evaluations and trace-driven simulations on a large dataset of approximately 9 million requests for 3 million content items. The results show that both algorithms are within 1% of the optimal for small problem sizes where computing the optimal solution is feasible and that significant reductions (up to 50%) in content access delay can be achieved over traditional LRU-based content caching schemes.

Our contributions can be summarized as follows:

- We provide a unified optimization formulation for the joint caching and routing problem for the CI-model and the CS-model and prove that the problem is NP-complete in both cases.
- We derive insights into problem complexity by considering several special cases, some of which are shown to admit efficient solutions, while others remain computationally hard.
- 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 these algorithms through numerical evaluations and trace-driven simulations. Numerical results show that the greedy algorithms perform close to the optimal solution when computing the optimal solution is feasible. Our results from trace-driven simulations show that the greedy algorithms yield significant performance improvement compared to solutions based on traditional LRU caching policy.

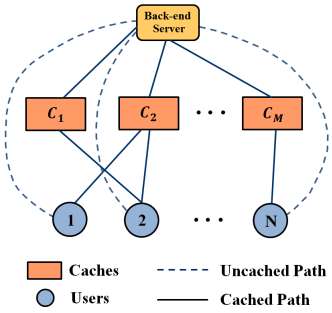


Fig. 1: Hybrid network with in-network caching

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 the connection between the 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 with aggregate rate λ_i. Aggregate request rate of all users is λ. We denote by q_{ij} the probability that user i generates a request for file j (referred to as the *file popularity*). The popularity of the same file can vary from one user to another.

Let $A = [a_{im}]$ denote the connections between users and caches, with $a_{im} = 1$ if user i is connected to cache m, and $a_{im} = 0$, otherwise. For the user-cache connections, let d^h_{im} and d^m_{im} denote the average delays incurred by user i in the event of a cache hit or miss at cache m, respectively. We assume without loss of generality that $d^m_{im} > d^h_{im}$, *i.e.*, cache misses always incur greater delays than cache hits. We consider two models for the delay over the path from users to

the back-end server. The first one is a congestion-insensitive model where the average delay experienced for a request by user i sent over the uncached path is d_i^b. The second model is a congestion-sensitive delay model where delays experienced over the uncached path depend on the traffic load. In the congestion-sensitive case, we assume the back-end server has service rate μ, and model the connections to the back-end server as an M/M/1 queue. The delay experienced over the uncached path then consists of an initial access delay with average d_i^b and a queuing (waiting plus service) delay with average $1/(\mu - \lambda_q)$, assuming λ_q is the request rate on the queue.

III. Problem Formulation

In this work, we consider a joint caching and routing problem with the goal of minimizing average content access delay over the requests of all users 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 the 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 [8] for a single cache that given a routing policy, *static caching* achieves minimum expected delay. 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 [8] was extended in [9] to a network of caches to show that static caching achieves minimum 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 content placement strategy $\mathbf{x} = [x_{jm}]$, and routing strategy $\mathbf{p} = [p_{ijm}]$. The optimal solution to the problem of joint caching and routing is therefore obtained by solving the following Mixed-Integer Program (MIP):

$$\begin{aligned} &\text{minimize } D(\mathbf{x}, \mathbf{p}) \\ &\text{such that } \sum_m p_{ijm} \le 1 && \forall i, j \\ &\qquad \sum_j x_{jm} \le C_m && \forall m \qquad (1) \\ &\qquad x_{jm} \in \{0,1\} && \forall j, m \\ &\qquad 0 \le p_{ijm} \le a_{im} && \forall i, j, m \end{aligned}$$

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_i^b, do not depend on the traffic load on the back-end server. For a given content placement $\mathbf{x}$ and routing policy $\mathbf{p}$, the average delay can be written as

$$D(\mathbf{x}, \mathbf{p}) = \frac{1}{\lambda} \sum_i \sum_j \lambda_i q_{ij} \Bigg(\sum_m p_{ijm} x_{jm} d_{im}^h + \sum_m p_{ijm}(1 - x_{jm}) d_{im}^m + \Big(1 - \sum_m p_{ijm}\Big) d_i^b \Bigg) \quad (2)$$

Without loss of generality, we assume that $d_{im}^h < d_i^b < d_{im}^m$ whenever user i is connected to cache m, *i.e.*, $a_{im} = 1$. Note that if $d_i^b \ge d_{im}^m, \forall i$, users connected to cache m will never use the uncached path. Also, if $d_i^b \le d_{im}^h, \forall i$, none of the users will actually 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. Therefore, the solution to the problem of joint caching and routing in the case of congestion-insensitive uncached path delays are obtained by solving the following binary linear program:

$$\begin{aligned} &\text{minimize } \frac{1}{\lambda} \sum_i \lambda_i \sum_j q_{ij} \left[\sum_m p_{ijm} d_{im}^h + (1 - \sum_m p_{ijm}) d_i^b \right] \\ &\text{such that } \sum_m p_{ijm} \le 1 \\ &\qquad \sum_j x_{jm} \le C_m \qquad (3) \\ &\qquad 0 \le p_{ijm} \le x_{jm} \cdot a_{im} \\ &\qquad x_{jm} \in \{0,1\}. \end{aligned}$$

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 $d_{im}^h < d_i^b$ and $d_i^b < d_{im}^m$, users have no incentive to split the traffic for any content between the cached and uncached paths, and hence there will be no routing variable, p_{ijm}, with a fractional value in the optimal solution, *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 [10]. Our formulation is more general as we consider non-homogeneous delays for the cached and uncached paths. HDP reduces to the optimization problem in (3) by setting $d_i^b = 1$, $d_{im}^h = 0$, and $C_m = C$, where C is the cache size at all caches in HDP.

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, and discuss 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 solution algorithms 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.

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.*, $C_m < N, \forall m$. The assumption that each user is interested in only one file allows us to re-write the objective function in (3) as

$$D(\mathbf{x},\mathbf{p}) = \frac{1}{\lambda}\sum_i \Big(\sum_m \lambda_i p_{iim} d^h_{im} + \lambda_i(1-\sum_m p_{iim}) d^b_i\Big)$$
$$= \frac{1}{\lambda}\Big(\sum_{i=1}^N \lambda_i d^b_i - \sum_i\sum_m \lambda_i p_{iim}(d^b_i - d^h_{im})\Big).$$

Since $\sum_{i=1}^N \lambda_i d^b_i$ is a constant independent of the decision variables, minimizing the above objective function is equivalent to maximizing $\sum_i \sum_m \lambda_i p_{iim}(d^b_i - d^h_{im})$. Note that $\lambda_i(d^b_i - d^h_{im})$ 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 $\{v^1_m, v^2_m, \ldots, v^{C_m}_m\}$ representing unit size micro-caches that form cache m. Let $V = \{v^1_1, v^2_1, \ldots, v^{C_1}_1, \ldots, v^1_M, \ldots, v^{C_M}_M\}$ 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 $\lambda_i(d^b_i - d^h_{im})$ as the weight of each edge connecting node u_i to nodes $v^s_m, \forall s \in \{1,2,\ldots,C_m\}$. 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.

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 [11]. In our context, the complexity is $O(M^3N^4)$. Note that $\sum_m C_m = O(MN)$ as we assumed $C_m < N, \forall m$. Therefore, we can solve the joint caching and routing problem in polynomial time when users are interested in one file only.

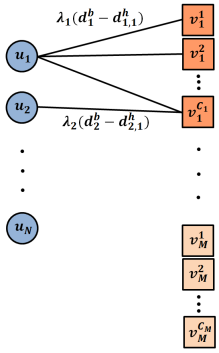


Fig. 2: Modeling content placement as maximum weighted matching problem

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 (3) can be found in polynomial time when there are two caches in the network. In the remainder of this section, we assume that x_{jm} take real values.

Before delving into the proof we introduce some definitions and results from [12]:

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

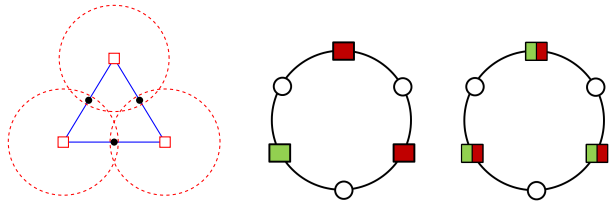
Definition 2. *An* $m \times n$ *integral matrix* A *is* totally unimodular *if the determinant of every square submatrix is* 0, 1, *or* -1.

Proposition 1. *If for a linear program* $\{\max c^T x : Ax \leq b\}$, A *is totally unimodular and* b *is integral, then there is an optimal solution to the linear program that is integral.*

Note that the three sets of constraints in the optimization problem in (3), namely, i) $\sum_m p_{ijm} \leq 1$, ii) $\sum_j x_{jm} \leq C_m$, and iii) $p_{ijm} - x_{jm} \cdot a_{im} \leq 0$ can be written in the form $Az \leq b$ where the entries of A and b are all integers, and z consists of the x_{jm} and p_{ijm} entries. From Proposition 1, then, it suffices to show that the matrix A is totally unimodular for a network with two caches to prove that optimization (3) can be solved in polynomial time. To prove that the matrix A is totally unimodular we use the following result from [13]:

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* $\{0, \pm 1\}$.

In [14], we give a constructive proof showing that for any subset R of rows of A we can find an assignment s that satisfies Proposition 2. Therefore, problem instances with at most two caches can be solved optimally in polynomial time.



(a) (b) (c)

Fig. 3: (a) A network with three users and three caches. User-cache connections form a cycle. (b) Optimal content placement according to binary content placement, *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. Complexity Discussion

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 another problem that is generally referred to as the "relaxation" of problem (3). Since the objective function in (3) is convex, the solution to the *relaxed problem* can be found in polynomial time for all instances of the problem. By comparing the solutions to the integer and the relaxed problems for a large number of instances of the optimization problem in (3) we observe that for most instances of the problem, solutions to problem (3) match those of the relaxed problem. Those instances that result in different solutions to the two problems exhibit a certain structure that we explain here.

Consider a network with three users and three caches as depicted in Figure 3a. With each user connected to two of the caches, the user-cache connections can be seen to form a cycle as demonstrated in Figure 3a. 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.

We show, in [14], that for all networks that consist of cycles formed by odd number of users and odd number of caches there are instances for which solutions of the original and relaxed problems differ. A detailed explanation is provided in [14]. Moreover, 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 (3) matches that of the relaxed problem. More specifically:

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 $4k+2, k \geq 1$ in the bipartite graph corresponding to the user-cache connections.*

V. Congestion-Sensitive Uncached Path

Next, we consider the case where delays on the uncached path depend on the traffic load on the back-end server. We model the uncached path as an M/M/1 queue with service rate μ. In addition to the queuing delay, we assume that user i observes an initial access delay to uncached path with average $d_i^b, i = 1, \ldots, N$. Here, we make no assumptions regarding d_i^b and d_{im}^h. Note that if $d_{im}^h \leq d_i^b$ and the needed object is in the cache user i will direct all her requests for that object to cache. If $d_{im}^h > d_i^b$, however, even if the needed content is in cache m, the user may prefer to use the uncached path, depending on the service rate and the load on the back-end server. For a given content placement $\mathbf{x}$ and routing policy $\mathbf{p}$, the average delay can be written as

$$D(\mathbf{x},\mathbf{p}) = \frac{1}{\lambda}\Bigg[\sum_i\sum_j \lambda_i q_{ij}\Bigg(\sum_m p_{ijm}x_{jm}d_{im}^h + \sum_m (1-x_{jm})p_{ijm}d_{im}^m + (1-\sum_m p_{ijm})d_i^b\Bigg) + \frac{\sum_i\sum_j \lambda_i q_{ij}(1-\sum_m p_{ijm})}{\mu - \sum_i\sum_j \lambda_i q_{ij}(1-\sum_m p_{ijm})}\Bigg]. \quad (4)$$

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 $\mu = +\infty$. This explains why 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

Modifying the delay function $D(\mathbf{x},\mathbf{p})$ in (4) 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 re-write the optimization problem as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{\lambda}\Bigg[\sum_{i=1}^N \lambda_i x_i p_i d_i^h + \sum_{i=1}^N \lambda_i (1-x_i)p_i d_i^m \\ & + \sum_{i=1}^N \lambda_i (1-p_i)d_i^b + \frac{\sum_{i=1}^N \lambda_i(1-p_i)}{\mu - \sum_{i=1}^N \lambda_i(1-p_i)}\Bigg] \\ \text{such that} \quad & \sum_{i=1}^N x_i \leq C \\ & 0 \leq p_i \leq a_i \\ & x_i \in \{0,1\}, \end{aligned} \quad (5)$$

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 the request rates of users, and let $\mathbf{d}^h = [d_i^h]$, $\mathbf{d}^m = [d_i^m]$ 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 are asking the following question: given the parameters $(\mu, \mathbf{\Lambda}, \mathbf{d}^h, \mathbf{d}^m, \mathbf{d}^b, C)$ 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 following problem 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. A proof of NP-hardness of a more general form of ECP is given in [15]. 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. □

Lemma 2. *CSDDP is NP-complete.*

Proof. A detailed proof of this lemma is given in [14], where we show ECP reduces to CSDDP. □

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.

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.

Let X_m denote the set of files stored in cache m, and define $X = X_1 \cup X_2 \cup \ldots \cup X_M$ to be the set of files stored in the M caches. 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 3. *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 $D(\mathbf{p}) = D(\mathbf{p}; \mathbf{x})$ 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 $\mathbf{x}$. We have $d_{ij}(\mathbf{x}) = \min_m d_{ijm}$, where d_{ijm} denotes the average delay of accessing content j from cache m, defined as (x_{jm} indicates that file j is in cache m)

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

Given the content placement in the caches, let p_{ij} denote the fraction of the traffic for which user i uses the cached paths to access content j. We can re-write the delay functions (2) and (4) 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

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

respectively. The optimal routing policy for a given content placement $\mathbf{x}$, then, is one that maximizes $-D(\mathbf{p}; \mathbf{x})$.

Let $\mathbf{x}_X$ be the equivalent binary representation of the content placement set X. We have the following lemma for both congestion-insensitive and congestion-sensitive delay models:

Lemma 4. *Let $\mathcal{P}$ denote all routing policies. For $X \subseteq S$, the function $F(X) = \max_{\mathbf{p} \in \mathcal{P}} -D(\mathbf{p}; \mathbf{x}_X)$ is a monotone increasing and submodular function.*

Proof. A detailed proof is given in [14]. □

A direct consequence of Lemma 4 is that minimizing the objective function in (2) or (4) is equivalent to maximizing a monotone submodular function. Therefore, the approximate solution obtained by the greedy algorithm in Algorithm 1 is within a $(1 - 1/e)$ factor of the optimal solution (see [16]).

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

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, $O(M^2 N^2 K^2 \log(NK))$. We

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**
  $s_{j^*m^*} \leftarrow \arg\max_{s_{jm} \in S} F(X \cup \{s_{jm}\})$
  $X_{m^*} \leftarrow X_{m^*} \cup \{s_{j^*m^*}\}$
  $X \leftarrow X \cup \{s_{j^*m^*}\}$
  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 routing as $\mathbf{p}^* \leftarrow \arg\min_{\mathbf{p}} D(\mathbf{p}; \mathbf{x}_X)$.

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

$X_m \leftarrow \emptyset, \forall m$
$X \leftarrow \emptyset$
$d_{ij} \leftarrow \min_c\{d_{ic}^m\}, \forall i, j$
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**
        $G_{jm} \leftarrow \sum_i \lambda_i q_{ij}(d_{ij} - \min\{d_{ij}, d_{im}^h\})$
  $[j^*, m^*] \leftarrow \arg\max_{j,m} G_{jm}$
  $X_{m^*} \leftarrow X_{m^*} \cup \{s_{j^*m^*}\}$
  $X \leftarrow X \cup \{s_{j^*m^*}\}$
  $d_{ij^*} \leftarrow \min\{d_{ij^*}, d_{im^*}^h\}, \forall i$
Content placement is done according to X.
Determine routing as $\mathbf{p}^* \leftarrow \arg\min_{\mathbf{p}} D(\mathbf{p}; \mathbf{x}_X)$

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, $\sum_i \lambda_i q_{ij}(d_{ij} - \min\{d_{ij}, d_{im}^h\})$. 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. Our goal is to evaluate 1) how well solutions of the greedy algorithms compare to the optimal solution (when computing the optimal solution is feasible), and 2) how well solutions from the greedy algorithms compare to those produced by a baseline. Due to lack of space, we only consider the more general case of congestion-sensitive delay model. For our baseline, we compare the approximate algorithms to the delay obtained by the following algorithm we will refer to as *p-LRU*.

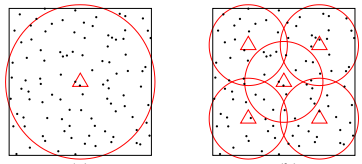
A. p-LRU

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 server. 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 optimized to minimize the average delay as explained in [14]. We use the *characteristic time* approximation [17], shown to be an accurate approximation, to compute the average cache access delays.

B. Network Setup

We consider a network with users uniformly distributed in a square field. We consider two architectures. First, we assume there is only one large cache at the center of the network as in Figure 4a. Second, we consider a network with five small caches with equal storage capacity as in Figure 4b. Figure 4 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 the 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 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 5 time units for each user, and the service rate is proportional to the aggregate request rate, where the scaling factor will be specified later.


(a) (b)
Fig. 4: 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 $\sum_i \lambda_i = 5$ requests per time unit. 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 equals $\mu = 1$.

Figure 5 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 in less than 20% of the time, and the relative inaccuracy is never more than 1%.

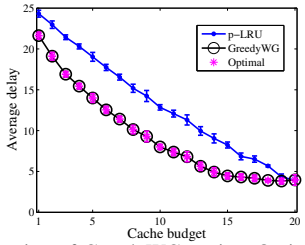


Fig. 5: Evaluation of GreedyWG against Optimal and p-LRU.

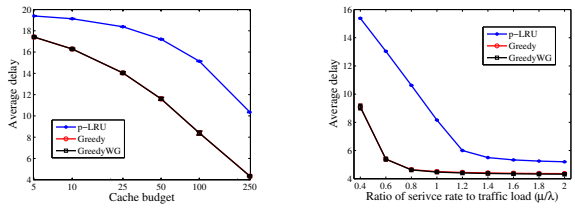
2) GreedyWG vs. Greedy: Next, we compare the solutions of GreedyWG against those of Greedy, the approximate algorithm in Algorithm 2 that has lower computational complexity but no performance guarantees. We consider a network with five caches and 100 users uniformly distributed in a 10×10 field.

Figure 6a 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 6b shows the average delay when the ratio of service rate to the total request rate changes from 0.4 to 2, with the aggregate traffic rate set to $\lambda = 5$. Similar to Figure 6a, Greedy performs very close to GreedyWG, and is always within 1% of GreedyWG.

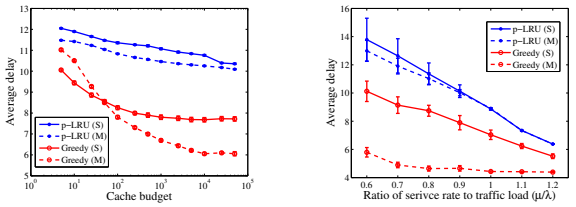
D. Trace-driven Simulation

Here, we present trace-driven evaluation results where we use traces for web accesses collected from an industrial research lab. The trace consists of approximately 9 million requests generated from 142,000 distinct IP addresses for more than 3 million distinct files. We only consider Greedy, the greedy algorithm presented in Algorithm 2, since it performs close to Algorithm 1, and has lower complexity.



(a) (b)

Fig. 6: Numerical evaluation of the two greedy approximations. Aggregate user request rate is $\lambda = 5$. (a) Service rate of the back-end server equals 2.5. (b) Cache budget is set to 125.



(a) (b)

Fig. 7: Trace-driven evaluation of the Greedy and p-LRU for the single-cache (S) and multi-cache (M) network setups. (a) The service rate is set to be 0.8 times the aggregate traffic rate. (b) Cache budget is 10,000.

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 distinct IP addresses. For every two consecutive segments, we use the first one as learning dataset from which we compute the file popularities and determine the optimal value of p for the p-LRU scheme as well as content placement and routing based on the Greedy algorithm. We use the second segment to compute the average delays under the p-LRU and Greedy algorithms.

Figure 7a compares the average delays for different cache budgets for the p-LRU and the Greedy algorithm for the single-cache (S) and multi-cache (M) networks. Significant reductions in average delay of up to 50% are observed for both 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 7b 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 7a, Greedy significantly reduces the average content access delay. Again, the cache architecture makes little difference for p-LRU but significant effect to 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

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 context of Internet, hybrid networks such as those considered in this paper, and sensor networks [3], [5], [6], [18]–[21]. Baev *et al.* [19] prove that the problem

of content placement with the objective of minimizing the access delay is NP-complete, and present approximate algorithms. The problem of efficient routing in cache networks has also been explored separately in the literature [22]–[24]. Rosensweig *et al.* [22] propose Breadcrumbs – a simple, best-effort routing policy for locating cached content. Cache-aware routing schemes that calculate paths with minimum transportation costs based on given caching policies and request demands have been proposed in [25].

The joint caching and routing problem, with the objective of minimizing content access delay, has recently been studied in [5], [6], where the authors consider a hybrid network consisting of multiple femto-cell 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 back-end server via the cellular network. They assume that the delays between users and caches are homogeneous and independent of the request rate.

Our work differs from the previous research 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 [5], [6] 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 identify 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 "hardness" of the general problem.

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, single-hop (uncached) path to a back-end content server; we considered cases in which this uncached path was modeled as a congestion-insensitive, constant-delay path, and a congestion-sensitive path modeled as an M/M/1 queue. 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 two-cache 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 a distributed algorithm for content placement and routing, and on developing solutions for the case of time-varying content popularity.

References

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

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

[3] E. Nygren, R. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet application," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, 2010.

[4] B. Azimdoost, C. Westphal, and H. Sadjadpour, "On the throughput capacity of information-centric networks," in *International Teletraffic Congress (ITC)*, September 2013, pp. 1–9.

[5] 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.

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

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

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

[9] M. Dehghan, A. Seetharamz, 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.

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

[11] D. B. West, *Introduction to graph theory (2nd ed.)*. Chapter 3, Prentice Hall, 2001.

[12] 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.

[13] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003, vol. 24.

[14] M. Dehghan, A. Seetharamz, 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. [Online]. Available: http://arxiv.org/pdf/1501.00216v1.pdf

[15] 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.

[16] 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.

[17] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *INFOCOM*, 2001.

[18] 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.

[19] 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.

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

[21] 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.

[22] E. Rosensweig and J. Kurose, "Breadcrumbs: efficient, best-effort content location in cache networks," in *IEEE INFOCOM*, 2009.

[23] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *IFIP Networking*, 2012.

[24] 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.

[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.