Jointly Optimal Routing and Caching for Arbitrary Network Topologies

Stratis Ioannidis *Member, IEEE,* and Edmund Yeh *Senior Member, IEEE*

***Abstract*—We study a problem of minimizing routing costs by jointly optimizing caching and routing decisions over an arbitrary network topology. We cast this as an equivalent caching gain maximization problem, and consider both source routing and hop-by-hop routing settings. The respective offline problems are NP-hard. Nevertheless, we show that there exist polynomial time approximation algorithms producing solutions within a constant approximation from the optimal. We also produce distributed, adaptive algorithms with the same approximation guarantees. We simulate our adaptive algorithms over a broad array of different topologies. Our algorithms reduce routing costs by several orders of magnitude compared to prior art, including algorithms optimizing caching under fixed routing.**

***Index Terms*—Caching, forwarding, routing, distributed optimization, convex relaxation, pipage rounding.**

I. Introduction

Storing content in a network to serve download requests is a problem as old as the Internet itself, arising in information-centric networks (ICNs) [2], [3], content-delivery networks (CDNs) [4], [5], web-cache design [6]–[8], wireless/femtocell networks [9]–[11], and peer-to-peer networks [12], [13]. Motivated by these applications, we study a *caching network*, i.e., a network of nodes augmented with additional caching capabilities. In such a network, some nodes act as designated content servers, permanently storing content and serving as "caches of last resort". Other nodes route requests towards these designated servers. If an intermediate node in the route towards a server caches the requested content, the request is satisfied early, and a copy of the content follows the reverse path towards the request's source.

This abstract setting naturally captures the above applications. For example, in the case of ICNs, designated servers correspond to traditional web servers permanently storing content, while nodes generating requests correspond to customer-facing gateways. Intermediate, cache-enabled nodes correspond to storage-augmented routers in the Internet's backbone: such routers forward requests but, departing from traditional network-layer protocols, immediately serve requests for content they store. An extensive body of research, both theoretical [7], [14]–[20] and experimental [2], [6]–[8], [21], has focused on modeling and analyzing caching networks in which *routing is fixed*: a request follows a predetermined route, e.g., the shortest path to the nearest designated server. Given routes to be followed and the demand for items, the above works determine (theoretically or empirically) the behavior of different caching algorithms deployed over intermediate nodes.

It is not a priori clear whether fixed routing and, more specifically, routing towards the nearest server is a justified design choice. This is of special interest in the context of ICNs, where delegating routing decisions to another protocol amounts to an "incremental" ICN deployment. For example, in such a deployment, requests can be routed towards the nearest designated server according to existing routing protocols such as OSPF or BGP [22]. An alternative is to *jointly* optimize both routing *and* caching decisions simultaneously, redesigning both caching *and* routing protocols from scratch. This poses a significant challenge as joint optimization is inherently combinatorial: indeed, jointly optimizing routing and caching decisions with the objective of, e.g., minimizing routing costs, is an NP-hard problem, and constructing a distributed approximation algorithm is far from trivial [9], [20], [23], [24].

This state of affairs gives rise to the following questions. First, is it possible to design *distributed, adaptive, and tractable algorithms jointly optimizing both routing and caching decisions over arbitrary cache network topologies, with provable performance guarantees*? Second, presuming such algorithms exist, *do they yield significant performance improvements over fixed routing protocols*? Answering this question in the affirmative may justify the potential increase in protocol complexity due to joint optimization. It can also inform future ICN design, settling whether an incremental approach (in which routing and caching are separate) suffices.

Our goal is to provide rigorous, comprehensive answers to these two questions. We make the following contributions:

- By constructing a counterexample, we show that fixed routing (and, in particular, routing towards the nearest server) can be *arbitrarily suboptimal* compared to jointly optimizing caching and routing decisions. Intuitively, joint optimization affects routing costs drastically because *exploiting path diversity increases caching opportunities*.
- We propose a formal mathematical framework for joint routing and caching optimization. We consider both *source routing* and *hop-by-hop* routing strategies, the two predominant classes of routing protocols over the Internet [22].
- We study the offline version of the joint routing and caching optimization problem, which is NP-hard, and construct a polynomial-time $1 - 1/e$ approximation algorithm. Our proposed algorithms first relaxes the integral

Manuscript received December 10, 2017; revised April 8th, 2018.

This work was supported in part by the National Science Foundation under Grant CNS-1718355, Grant CNS-1423250, and in part by a Cisco Systems Research Grant. This is an extended version of a paper that appeared in the 4th ACM Conference on Information-Centric Networking (ICN 2017) [1]. *(Corresponding author: Stratis Ioannidis.)*

The authors are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, 02115 USA (e-mail: ioannidis@ece.neu.edu, eyeh@ece.neu.edu).

problem to a convex optimization problem. The resulting solution is subsequently rounded to produce an integral solution.
- We provide a *distributed*, *adaptive* algorithm that converges to joint routing and caching strategies that are, globally, within a $1 - 1/e$ approximation ratio from the optimal. Our distributed implementation amounts to a projected gradient ascent (PGA) over the convex relaxation used in our offline algorithm, coupled with a randomized rounding technique.
- We evaluate our distributed algorithm over 9 synthetic and 3 real-life network topologies, and show that it significantly outperforms the state of the art: it reduces routing costs by a factor between 10 and 1000, for a broad array of competitors, including both fixed and dynamic routing protocols.

The remainder of this paper is organized as follows. We review related work in Section II, and present our mathematical model of a caching network in Section III. The suboptimality of fixed routing is shown in Section IV, while our offline and online algorithms are presented in Sections V and VI, respectively, under source routing. Extensions to hop-by-hop routing are discussed in Section VII. A numerical evaluation of our algorithms over several topologies is presented in Section VIII, and we conclude in Section IX.

II. Related Work

There is a vast literature on *individual caches*, serving as fast secondary memory devices, and the topic is classic (see, e.g., [25]–[27]). Nevertheless, the study of *networks of caches* still poses significant challenges. A central problem is that, even if arriving traffic in a cache is, e.g., Poisson, the *outgoing* traffic is often hard to describe analytically, even when caches implement simple eviction policies. This is true for several traditional eviction policies, like Least Recently Used (LRU), Least Frequently Used (LFU), First In First Out (FIFO), and Random Replacement (RR). The *Che approximation* [7], [14], a significant breaktrough, approximates the hit rate under several eviction policies by assuming constant occupancy times. This approximation is accurate in practice [14], and its success has motivated extensive research in so-called *time-to-live* (TTL) caches. A series of recent works have focused on identifying how to set TTLs to (a) approximate the behavior of known eviction policies, (b) describe hit-rates in closed-form formulas [7], [15]–[17], [28]. Despite these advances, none of the above works address issues of routing cost minimization over multiple hops, which is our goal.

A simple, elegant, and ubiquitous algorithm for populating caches under fixed routing is *path replication* [12], sometimes also referred to as "leave-copy-everywhere" (LCE) [6]: once a cache hit occurs, every downstream node receiving the response caches the content, while eviction happen via LRU, LFU, FIFO, and other traditional policies. Several variants exist: in "leave-copy-down" (LCD), a copy is placed *only* in the node immediately preceding the cache where the hit occurred [6], [29], while "move-copy-down" (MCD) also removes the present upstream copy. Probabilistic variants have also been proposed [30]. Several works [6], [21], [30]–[32] experimentally study these variants over a broad array of topologies. Despite the simplicity and elegance inherent in path replication, when targeting an optimization objective such as, e.g., minimizing total routing costs, all of the above variants, combined any of the traditional eviction policies, are known to be arbitrarily suboptimal [20].

In their seminal paper [12] introducing path replication, Cohen and Shenker also introduced the abstract problem of finding a content placement that minimizes routing costs. The authors show that path replication combined with a constant rate of evictions leads to an allocation that is optimal, in equilibrium, when nodes are visited through uniform sampling. Unfortunately, optimality breaks down when uniform sampling is replaced by routing over graph topologies [20]. Several papers have studied the offline cost minimization under restricted topologies [9], [23], [24], [33]–[35]. With the exception of [9], these works model the network as a bipartite graph: nodes generating requests connect directly to caches in a single hop, and algorithms do not readily generalize to arbitrary topologies. In general, the *pipage rounding* technique of Ageev and Sviridenko [36] yields again a constant approximation algorithm in the bipartite setting, while approximation algorithms are also known for several variants of this problem [23], [24], [33], [34]. Excluding [24], all these works focus only on centralized solutions of the offline caching problem; none considers jointly optimizing caching *and* routing decisions.

Joint caching and routing has been studied in restricted settings. The benefit of routing towards nearest replicas, rather than towards nearest designated servers, has been observed empirically [37]–[39]. Deghan et al. [5], Abedini and Shakkotai [40], and Xie et al. [41] all study joint routing and content placement schemes in a bipartite, single-hop setting. In all cases, minimizing the single-hop routing cost reduces to solving a linear program; Naveen et al. [10] extend this to other, non-linear (but still convex) objectives of the hit rate, still under single-hop, bipartite routing constraints. None of these approaches generalize to a multi-hop setting, which leads to non-convex formulations (see Section III-F); addressing this lack of convexity is one of our technical contributions. A multi-hop, multi-path setting is formally analyzed by Carofiglio et al. [39] under restricted arrival rates, assuming that requests by different users follow non-overlapping paths. Our approach addresses the problem in its full generality, for arbitrary topologies, arrival rates, and overlapping paths.

When routes are fixed, and only caching decisions are optimized, maximizing the caching gain amounts to maximizing a submodular function subject to matroid constraints [9], [20]. Problems with structure appear in many important applications related to combinatorial optimization [42]–[46]; for an overview of the topic, see Krause and Golovin [47]. Though generic submodular maximization subject to matroid constraints is NP-hard, several known approximation algorithms exist in the so-called value oracle model (i.e., assuming access to a poly-time oracle that evaluates the submodular objective). Nemhauser et al. [43] show that the greedy algorithm produces a solution within 1/2 of the optimal. Vondrák [44] and Calinescu et al. [45], [46] show that the so-

called continuous-greedy algorithm produces a solution within $(1 - 1/e)$ of the optimal in polynomial time, which cannot be further improved [48].

Under the value oracle model, the continuous-greedy algorithm requires random sampling to estimate the gradient of the so-called multilinear relaxation of the objective. For the specific objective of maximizing the caching gain under fixed routing, the concave relaxation technique of Ageev and Sviridenko [36] attains the $1 - 1/e$ approximation ratio while eschewing sampling; this is shown in [9] for homogeneous caches and a specific class of topologies, and generalized to heterogeneous caches and arbitrary topologies in [20].

Jointly optimizing routing and caching decisions is *not* a submodular maximization problem subject to matroid constraints. Nevertheless, we show that that a variant the technique by Ageev and Sviridenko [36] can be used to obtain a poly-time approximation algorithm, that also lends itself to a distributed, adaptive implementation. We show this by extending [20] to incorporate routing decisions, both through source and hop-by-hop routing. Crucially, our evaluations in Section VIII show that jointly optimizing caching *and* routing significantly improves performance compared to fixed routing, reducing the routing costs of [20] by as much as three orders of magnitude.

III. Model

We begin by presenting our formal model, extending [20] to account for both caching *and* routing decisions. Our analysis applies to two routing variants: (a) *source routing* and (b) *hop-by-hop routing*. In both cases, we study two types of strategies: *deterministic* and *randomized*. For example, in source routing, requests for an item originating from the same source may be forwarded over several possible paths, given as input. In deterministic source routing, *only one* is selected and used *for all subsequent requests* with this origin. In contrast, a randomized strategy samples a new path to follow independently with each new request. We also use similar deterministic and randomized analogues both for caching strategies as well as for hop-by-hop routing strategies.

Randomized strategies subsume deterministic ones, and are arguably more flexible and general. This begs the question: why study both? There are three reasons. First, optimizing deterministic strategies naturally relates to combinatorial techniques such as [36], which we can leverage to solve the offline probem. Second, the online, distributed algorithms we propose to construct randomized strategies in fact *mirror our solution to the offline, deterministic problem*: they leverage the same convex relaxation. Finally, and most importantly: deterministic strategies turn out to be equivalent to randomized strategies! As we show in Thm. 3, the smallest routing cost attained by randomized strategies is exactly the same as the one attained by deterministic strategies.

A. Network Model and Content Requests

Consider a network represented as a directed, symmetric[1] graph $G(V, E)$. Content items (e.g., files, or file chunks) of equal size are to be distributed across network nodes. Each node is associated with a cache that can store a finite number of items. We denote by C the set of possible content items, i.e., the *catalog*, and by $c_v \in \mathbb{N}$ the cache capacity at node $v \in V$: exactly c_v content items can be stored in v. The network serves content requests routed over the graph G. A request (i, s) is determined by (a) the item $i \in C$ requested, and (b) the source $s \in V$ of the request. We denote by $\mathcal{R} \subseteq C \times V$ the set of all requests. Requests of different types $(i, s) \in \mathcal{R}$ arrive according to independent Poisson processes with arrival rates $\lambda_{(i,s)} > 0$, $(i, s) \in \mathcal{R}$.

For each item $i \in C$ there is a fixed set of *designated server* nodes $\mathcal{S}_i \subseteq V$, that always store i. A node $v \in \mathcal{S}_i$ permanently stores i in *excess memory outside its cache*. Thus, the placement of items to designated servers is fixed and outside the network's design. A request (i, s) is routed over a path in G towards a designated server. However, forwarding terminates upon reaching *any intermediate cache* that stores i. At that point, a response carrying i is sent over the reverse path, i.e., from the node where the cache hit occurred, back to source node s. Both caching *and* routing decisions are network design parameters, which we define formally below.

[1] A directed graph is symmetric when $(i, j) \in E$ implies that $(j, i) \in E$.

TABLE I
Notation Summary

Common Notation	
$G(V, E)$	Network graph, with nodes V and edges E
C	Item catalog
c_v	Cache capacity at node $v \in V$
w_{uv}	Weight of edge $(u, v) \in E$
$\mathcal{R}$	Set of requests (i, s), with $i \in C$ and source $s \in V$
$\lambda_{(i,s)}$	Arrival rate of requests $(i, s) \in \mathcal{R}$
$\mathcal{S}_i$	Set of designated servers of $i \in C$
x_{vi}	Variable indicating whether $v \in V$ stores $i \in C$
ξ_{vi}	Marginal probability that v stores i
X	Global caching strategy of x_{vi}s, in $\{0, 1\}^{\lvert V\rvert \times \lvert C\rvert}$
Ξ	Expectation of caching strategy matrix X
T	Duration of a timeslot in online setting
w_{uv}	weight/cost of edge (u, v)
$\texttt{supp}(\cdot)$	Support of a probability distribution
$\texttt{conv}(\cdot)$	Convex hull of a set
Source Routing	
$\mathcal{P}_{(i,s)}$	Set of paths request $(i, s) \in \mathcal{R}$ can follow
$P_{\texttt{SR}}$	Total number of paths
p	A simple path of G
$k_p(v)$	The position of node $v \in p$ in path p.
$r_{(i,s),p}$	Variable indicating whether $(i, s) \in \mathcal{R}$ is forwarded over $p \in \mathcal{P}_{(i,s)}$
$\rho_{(i,s),p}$	Marginal probability that s routes request for i over p
r	Routing strategy of $r_{(i,s),p}$s, in $\{0, 1\}^{\sum_{(i,s)\in\mathcal{R}} \lvert\mathcal{P}_{(i,s)}\rvert}$.
ρ	Expectation of routing strategy vector r
$\mathcal{D}_{\texttt{SR}}$	Feasible strategies (r, X) of MaxCG-S
RNS	Route to nearest server
RNR	Route to nearest replica
Hop-by-Hop Routing	
$G^{(i)}$	DAG with sinks in $\mathcal{S}_i$
$E^{(i)}$	Edges in DAG $G^{(i)}$
$G^{(i,s)}$	Subgraph of $G^{(i)}$ including only nodes reachable from s
$\mathcal{P}^{u}_{(i,s)}$	Set of paths in $G^{(i,s)}$ from s to u.
$P_{\texttt{HH}}$	Total number of paths
$r^{(i)}_{uv}$	Variable indicating whether u forwards a request for i to v
$\rho^{(i)}_{uv}$	Marginal probability that u forwards a request for i to v
r	Routing strategy of $r^{i}_{u,v}$s, in $\{0, 1\}^{\sum_{i\in C} \lvert E^{(i)}\rvert}$.
ρ	Expectation of routing strategy vector r
$\mathcal{D}_{\texttt{HH}}$	Feasible strategies (r, X) of MaxCG-HH

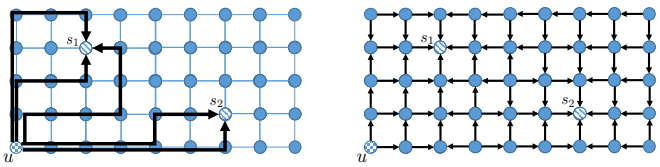


Fig. 1. Source Routing vs. Hop-by-Hop routing. In source routing, shown left, source node u on the bottom left can choose among 5 possible paths to route a request to one of the designated servers storing i (s_1, s_2). In hop-by-hop routing, each intermediate node selects the next hop among one of its neighbors in a DAG whose sinks are the designated servers.

B. Caching Strategies

We study both *deterministic* and *randomized* caches.

Deterministic caches. For each node $v \in V$, we define v*'s caching strategy* as a vector $x_v \in \{0,1\}^{|C|}$, where $x_{vi} \in \{0,1\}$, for $i \in C$, is the binary variable indicating whether v stores content item i. As v can store no more than c_v items:

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

The *global caching strategy* is the matrix $X = [x_{vi}]_{v \in V, i \in C} \in \{0,1\}^{|V| \times |C|}$, whose rows comprise the caching strategies of each node.

Randomized caches. In randomized caches, the caching strategies x_v, $v \in V$, are random variables. We denote by:

$$\xi_{vi} \equiv \mathbf{P}[x_{vi} = 1] = \mathbb{E}[x_{v,i}] \in [0,1], \text{ for } i \in C, \tag{2}$$

the marginal probability that node v caches item i, and by

$$\Xi = [\xi_{vi}]_{v \in V, i \in C} = \mathbb{E}[X] \in [0,1]^{|V| \times |C|}, \tag{3}$$

the corresponding expectation of the global caching strategy.

C. Source Routing Strategies

Recall that requests are routed towards designated server nodes. In source routing, for every request $(i,s) \in C \times V$, there exists a set $\mathcal{P}_{(i,s)}$ of *paths* that the request can follow towards a designated server in $\mathcal{S}_i$. A source node s can forward a request among any of these paths, but we assume each response follows the same path as its corresponding request.

Formally, a path p of length $|p| = K$ is a sequence $\{p_1, p_2, \ldots, p_K\}$ of nodes $p_k \in V$ such that $(p_k, p_{k+1}) \in E$, for every $k \in \{1, \ldots, |p| - 1\}$. We make the following natural assumptions on the set of paths $\mathcal{P}_{(i,s)}$. For every $p \in \mathcal{P}_{(i,s)}$:

(a) p starts at s, i.e., $p_1 = s$;
(b) p is simple, i.e., it contains no loops;
(c) the last node in p is a designated server for item i, i.e., if $|p| = K$, $p_K \in \mathcal{S}_i$; and
(d) no other node in p is a designated server for i, i.e., if $|p| = K$, $p_k \notin \mathcal{S}_i$, for $k = 1, \ldots, K-1$.

These properties imply that a request routed over a path $p \in \mathcal{P}_{(i,s)}$ is always satisfied as, in the worst case, an item is retrieved at the terminal designated server. Given a path p and a $v \in p$, we denote by $k_p(v)$ the position of v in p; i.e., $k_p(v)$ equals to $k \in \{1, \ldots, |p|\}$ such that $p_k = v$.

Deterministic Routing. Given sets $\mathcal{P}_{(i,s)}$, $(i,s) \in \mathcal{R}$, the *routing strategy* of a source $s \in V$ w.r.t. request $(i,s) \in \mathcal{R}$ is a vector $r_{(i,s)} \in \{0,1\}^{|\mathcal{P}_{(i,s)}|}$, where $r_{(i,s),p} \in \{0,1\}$ is a binary variable indicating whether s selected path $p \in \mathcal{P}_{(i,s)}$. These satisfy:

$$\sum_{p \in \mathcal{P}_{(i,s)}} r_{(i,s),p} = 1, \text{ for all } (i,s) \in \mathcal{R}, \tag{4}$$

indicating that exactly one path is selected. Let

$$P_{\text{SR}} = \sum_{(i,s) \in \mathcal{R}} |\mathcal{P}_{(i,s)}| \tag{5}$$

be the total number of paths. We refer to the vector

$$r = [r_{(i,s),p}]_{(i,s) \in \mathcal{R}, p \in \mathcal{P}_{(i,s)}} \in \{0,1\}^{P_{\text{SR}}} \tag{6}$$

as the *global routing strategy*.

Randomized Routing. In randomized routing, variables $r_{(i,s)}$, $(i,s) \in \mathcal{R}$ are random. That is, we randomize routing by allowing requests to be routed over a random path in $\mathcal{P}_{(i,s)}$, selected *independently of all past routing decisions (at s or elsewhere)*. We denote by

$$\rho_{(i,s),p} \equiv \mathbf{P}[r_{(i,s),p} = 1] = \mathbb{E}[r_{(i,s),p}], \text{ for } p \in \mathcal{P}_{(i,s)}, \tag{7}$$

the probability that path p is selected by s, and by

$$\rho = [\rho_{(i,s),p}]_{(i,s) \in \mathcal{R}, p \in \mathcal{P}_{(i,s)}} = \mathbb{E}[r] \in [0,1]^{P_{\text{SR}}} \tag{8}$$

the expectation of the global routing strategy r.

Remark. We make no a priori assumptions on P_{SR}, the total number of paths used during source routing. The complexity of our offline algorithm, and the rate of convergence of our distributed, adaptive algorithm depend on P_{SR} (see Lemma 5). In practice, if the number of possible paths is, e.g., exponential in $|V|$, it makes sense to restrict each $\mathcal{P}_{(i,s)}$ to a small subset of possible paths, or to use hop-by-hop routing instead. As discussed below, the later restricts the maximum number of paths considered.

As we treat path sets $P_{(i,s)}$ as inputs, our algorithms and performance guarantees apply irrespectively of how these paths are constructed or selected. That said, there exist both centralized and distributed algorithms for constucting such multi-path alternatives, such as k-shortest path algorithms [49], [50], equal cost multipath routing (ECMP) [51], [52], and multipath distance vector routing [53], [54]. All of these include inherent caps on the multiplicity of alternative paths discovered, and can thus be used to construct a input instance to our problem whose P_{SR} is polynomial in the number of nodes.

D. Hop-by-Hop Routing Strategies

Under hop-by-hop routing, each node along the path makes an individual decision on where to route a request message. When a request for item i arrives at an intermediate node $v \in V$, node v determines how to forward the request to one of its neighbors. The decision depends on i but *not* on the request's source. This limits the paths a request may follow, making hop-by-hop routing less expressive than source routing. On the other hand, reducing the space of routing strategies reduces complexity. In adaptive algorithms, it also speeds up convergence, as routing decisions w.r.t. i are "learned" across requests by different sources.

To ensure loop-freedom, we must assume that forwarding decisions are restricted to a subset of possible neighbors in G. For each $i \in C$, we denote by $G^{(i)}(V, E^{(i)})$ a graph that has the following properties:

(a) $G^{(i)}$ is a subgraph of G, i.e., $E^{(i)} \subseteq E$;
(b) $G^{(i)}$ is a directed acyclic graph (DAG); and
(c) (c) a node v in $G^{(i)}$ is a sink if and only if it is a designated server for i, i.e., $v \in \mathcal{S}_i$.

We assume that every node $v \in V$ can forward a request for item i *only to a neighbor in* $G^{(i)}$. Then, the above properties of $G^{(i)}$ ensure both loop freedom and successful termination.

Deterministic Routing. For any node $s \in V$, let $G^{(i,s)}$ be the induced subgraph of $G^{(i)}$ which results from removing any nodes in $G^{(i)}$ not reachable from s. For any u in $G^{(i,s)}$, let $\mathcal{P}^u_{(i,s)}$ be the set of all paths in $G^{(i,s)}$ from s to u, and denote the total number of paths by

$$P_{\text{HH}} = \sum_{(i,s)\in C} \sum_{u \in V} |\mathcal{P}^u_{(i,s)}|. \tag{9}$$

We denote by $r^{(i)}_{uv} \in \{0, 1\}$, for $(u, v) \in E^{(i)}$, $i \in C$, the decision variable indicating whether u forwards a request for i to v. The *global routing strategy* is $r = [r^{(i)}_{uv}]_{i\in C,(u,v)\in E^{(i)}} \in \{0, 1\}^{\sum_{i\in C} |E^{(i)}|}$, and satisfies

$$\sum_{v:(u,v)\in E^{(i)}} r^{(i)}_{uv} = 1, \quad \text{for all } v \in V, i \in C. \tag{10}$$

Note that, in contrast to source routing strategies, that have length P_{SR}, hop-by-hop routing strategies have length at most $|C||E|$.

Randomized Routing. As in source routing, we also consider randomized hop-by-hop routing strategies, whereby each request is forwarded independently from previous routing decisions to one of the possible neighbors. We again denote by

$$\begin{aligned}\rho &= [\rho^{(i)}_{uv}]_{i\in C,(u,v)\in E^{(i)}} = [\mathbb{E}[r^{(i)}_{uv}]]_{i\in C,(u,v)\in E^{(i)}} \\ &= \left[\mathbf{P}[r^{(i)}_{uv} = 1]\right]_{i\in C,(u,v)\in E^{(i)}} \in [0, 1]^{\sum_{i\in C} |E^{(i)}|},\end{aligned} \tag{11}$$

the vector of corresponding (marginal) probabilities of routing decisions at each node v.

Remark. Given G and $\mathcal{S}_i$, $G^{(i)}$ can be constructed in polynomial time using, e.g., the Bellman-Ford algorithm [55]. Indeed, requiring that v forwards requests for $i \in C$ only towards neighbors with a smaller distance to a designated server in $\mathcal{S}_i$ results in such a DAG. A distance-vector protocol [22] can form this DAG in a distributed fashion. This need only be executed once, before any subsequent caching and routing optimization algorithms are executed. Other constructions are also possible. For example, one could determine a potential function for each node, where $\mathcal{S}_i$ has zero potential, and considering only edges that decrease the potential. Our proposed algorithms work for arbitrary DAGs, irrepectively of how they are produced (i.e., even if the potential function is not the distance to $\mathcal{S}_i$), though in practise it would be preferable to, e.g., take into account edge weights when computing distances, which we introduce below in Sec. III-F. That said, once DAGs $G^{(i)}$ have been constructed, our algorithms can be executed with these as inputs.

E. Offline vs. Online Setting

To reason about the caching networks we have proposed, we consider two settings: the *offline* and *online* setting. In the offline setting, all problem inputs (demands, network topology, cache capacities, etc.) are known apriori to, e.g., a system designer. At time $t = 0$, the system designer selects (a) a caching strategy X, and (b) a routing strategy r. Both can be either deterministic or randomized, but both are also static: they do not change as time progresses. In the case of caching, cache contents (selected deterministically or at random at $t = 0$) remain fixed for all $t \geq 0$. In the case of routing decisions, the distribution over paths (in source routing) or neighbors (in hop-by-hop routing) remains static, but each request is routed independently of previous requests.

In the online setting, no a priori knowledge of the demand, i.e., the rates of requests $\lambda_{(i,s)}$, $(i, s) \in \mathcal{R}$ is assumed. Both caching *and* routing strategies change through time via a *distributed, adaptive* algorithm. Time is slotted, and each slot has duration $T > 0$. During a timeslot, both caching and routing strategies remain fixed. Nodes have access only to local information: they are aware of their graph neighborhood and state information they maintain locally. They exchange messages, including both normal request and response traffic, as well as (possibly) control messages, and may adapt their state. At the conclusion of a time slot, each node changes its caching and routing strategies. Changes made by v depend only on its neighborhood, its current local state, as well as on messages that node v received in the previous timeslot. Both caching and routing strategies during a timeslot may be deterministic or randomized. Note that implementing a caching strategy at the conclusion of a timeslot involves changing cache contents, which incurs additional overhead; if T is large, however, this cost is negligible compared to the cost of transferring items during a timeslot.

F. Optimal Routing and Caching

We are now ready to formally pose the problem of jointly optimizing routing and caching. We pose here the offline problem in which problem inputs are given; nevertheless, we will devise distributed, adaptive algorithms that do not a priori know the demand in Section VI.

To capture costs (e.g., latency, money, etc.), we associate a *weight* $w_{uv} \geq 0$ with each edge $(u, v) \in E$, representing the cost of transferring an item across this edge. We assume that costs are solely due to response messages that carry an item, while request-forwarding costs are negligible. We do not assume that $w_{uv} = w_{vu}$. We describe the cost minimization objectives under source and hop-by-hop routing below.

Source Routing. The cost of serving a request $(i, s) \in \mathcal{R}$ under source routing is:

$$C^{(i,s)}_{\text{SR}}(r, X) = \sum_{p\in\mathcal{P}_{(i,s)}} r_{(i,s),p} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'}i}). \tag{12}$$

Intuitively, (12) states that $C^{(i,s)}_{\text{SR}}$ includes the cost of an edge (p_{k+1}, p_k) in the path p if (a) p is selected by the routing strategy, and (b) *no* cache preceding this edge in p stores i.

In the deterministic setting, we seek a global caching and routing strategy (r, X) minimizing the aggregate expected cost, defined as:

$$C_{\text{SR}}(r, X) = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} C_{\text{SR}}^{(i,s)}(r, X), \tag{13}$$

with $C_{\text{SR}}^{(i,s)}$ given by (12). That is, we wish to solve:

$$\begin{aligned} &\text{MinCost-SR} \\ \text{Minimize:} \quad & C_{\text{SR}}(r, X) && (14a)\\ \text{subj. to:} \quad & (r, X) \in \mathcal{D}_{\text{SR}} && (14b)\end{aligned}$$

where $\mathcal{D}_{\text{SR}} \subset \mathbb{R}^{P_{\text{SR}}} \times \mathbb{R}^{|V|\times|C|}$ is the set of (r, X) satisfying the routing, capacity, and integrality constraints, i.e.:

$$\begin{aligned} &\textstyle\sum_{i\in C} x_{vi} = c_v, && \forall v \in V, && (15a)\\ &\textstyle\sum_{p\in\mathcal{P}_{(i,s)}} r_{(i,s),p} = 1, && \forall (i,s) \in \mathcal{R}, && (15b)\\ &x_{vi} \in \{0, 1\}, && \forall v \in V, i \in C, && (15c)\\ &r_{(i,s),p} \in \{0, 1\}, && \forall p \in \mathcal{P}_{(i,s)}, (i,s) \in \mathcal{R}. && (15d)\end{aligned}$$

This problem is NP-hard, even in the case where routing is fixed: see Shanmugam et al. [9] for a reduction from the 2-Disjoint Set Cover Problem.

Hop-By-Hop Routing. Similarly to (12), under hop-by-hop routing, the cost of serving (i, s) can be written as:

$$\begin{aligned} C_{\text{HH}}^{(i,s)}(r, X) = &\sum_{(u,v)\in G^{(i,s)}} w_{vu} \cdot r_{uv}^{(i)}(1 - x_{ui}) \cdot \ldots \\ &\sum_{p\in\mathcal{P}_{(i,s)}^{u}} \prod_{k'=1}^{|p|-1} r_{p_{k'}p_{k'+1}}^{(i)}(1 - x_{p_{k'}i}). \end{aligned} \tag{16}$$

We wish to solve:

$$\begin{aligned} &\text{MinCost-HH} \\ \text{Minimize:} \quad & C_{\text{HH}}(r, X) && (17a)\\ \text{subj. to:} \quad & (r, X) \in \mathcal{D}_{\text{HH}} && (17b)\end{aligned}$$

where $C_{\text{HH}}(r, X) = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} C_{\text{HH}}^{(i,s)}(r, X)$ is the expected routing cost, and $\mathcal{D}_{\text{HH}}$ is the set of $(r, X) \in \mathbb{R}^{\sum_{i\in C}|E^{(i)}|} \times \mathbb{R}^{|V|\times|C|}$ satisfying the constraints:

$$\begin{aligned} &\textstyle\sum_{i\in C} x_{vi} = c_v, && \forall v \in V, && (18a)\\ &\textstyle\sum_{v:(u,v)\in E^{(i)}} r_{uv}^{(i)} = 1 && \forall v \in V, i \in C, && (18b)\\ &x_{vi} \in \{0, 1\}, && \forall v \in V, i \in C, && (18c)\\ &r_{uv}^{(i)} \in \{0, 1\}, && \forall (u,v) \in E^{(i)}, i \in C. && (18d)\end{aligned}$$

Randomization. The above routing cost minimization problems can also be stated in the context of randomized caching and routing strategies. For example, in the case of source routing, assuming (a) independent caching strategies across nodes selected at time $t = 0$, with marginal probabilities given by Ξ, and (b) independent routing strategies at each source, with marginals given by ρ (also independent from caching strategies), all terms in C_{SR} contain products of independent random variables; this implies that:

$$\mathbb{E}[C_{\text{SR}}(r, X)] = C_{\text{SR}}[\mathbb{E}[r], \mathbb{E}[X]] = C_{\text{SR}}(\rho, \Xi), \tag{19}$$

where the expectation is taken over the randomness of both caching and routing strategies. The expected routing cost thus depends on the routing and caching strategies only through the expectations ρ and Ξ. This has the following consequence:

Lemma 1: Under randomized routing and caching strategies, problem MinCost-SR becomes

$$\min_{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{SR}})} C_{\text{SR}}(\rho, \Xi), \tag{20}$$

while problem MinCost-HH becomes

$$\min_{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{HH}})} C_{\text{HH}}(\rho, \Xi),, \tag{21}$$

where $\texttt{conv}(\mathcal{D}_{\text{SR}})$, $\texttt{conv}(\mathcal{D}_{\text{HH}})$ are the convex hulls of $\mathcal{D}_{\text{SR}}$, $\mathcal{D}_{\text{HH}}$, respectively.

Proof: We prove this for source-routing strategies; the proof for hop-by-hop strategies follows a similar argument. Consider a randomized routing strategy r and a randomized caching strategy X, such that $(r, X) \in \mathcal{D}_{\text{SR}}$. Let $\rho = \mathbb{E}[r]$ and $\Xi = \mathbb{E}[X]$. Then $(r, X) \in \mathcal{D}_{\text{SR}}$ readily implies that $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$; moreover, by (19), its expected routing cost would be exactly given by $C_{\text{SR}}(\rho, \Xi)$, so a randomized solution to MinCost-SR immediately yields a solution to the relaxed problem. To complete the proof, we need to show that any feasible solution $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$, we can constuct a MinCost feasible pair of randomized strategies $(r, X) \in \mathcal{D}_{\text{SR}}$ whose expectations are precisely (ρ, Ξ); then, by (19), it must be that $\mathbb{E}[C_{\text{SR}}(r, X)] = C_{\text{SR}}(\rho, X)$. Note that this construction is trivial for routing strategies: given $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$, we can construct a randomized strategy r by setting $r_{(i,s)}$ for each $(i, s) \in \mathcal{R}$ to be an independent categorical variable over $\mathcal{P}_{(i,s)}$ with $\mathbf{P}[r_{(i,s),p} = 1] = \rho_{(i,s),p}$, for $p \in \mathcal{P}_{(i,s)}$. It is less obvious how to do so for caching strategies; nevertheless, the technique by [20], [56] discussed in Section VI-D achieves precisely the desired property: given a feasible Ξ, it produces a feasible randomized integral X, independent across nodes that (a) satisfies capacity constraints exactly, and (b) has marginals given by Ξ. ■

The objective functions C_{SR}, C_{HH} are *not convex* and, therefore, the corresponding relaxed problems are *not* convex optimization problems. This is in stark contrast to single-hop settings, that often can naturally be expressed as linear programs [5], [10], [40].

G. Fixed Routing

When the global routing strategy r is fixed, (14) reduces to

$$\begin{aligned} \text{Minimize:} \quad & C_{\text{SR}}(r, X) && (22a)\\ \text{subj. to:} \quad & X \text{ satisfies (15a) and (15c).} && (22b)\end{aligned}$$

MinCost-HH can be similarly restricted to caching only. We studied this restricted optimization in earlier work [20]. In particular, under given global routing strategy r, we cast (22) as a maximization problem as follows. Let $C_0^r = C_{\text{SR}}(r, 0) = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{p\in\mathcal{P}_{(i,s)}} r_{(i,s),p} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}$ be the cost when all caches are empty (i.e., X is the zero matrix). Note that this is a constant that does not depend on X. Consider the following maximization problem:

$$\begin{aligned} \text{Maximize:} \quad & F_{\text{SR}}^r(X) = C_0^r - C_{\text{SR}}(r, X) && (23a)\\ \text{subj. to:} \quad & X \text{ satisfies (15a) and (15c).} && (23b)\end{aligned}$$

This problem is equivalent to (22), in that a feasible solution to (23) is optimal if and only if it also optimal for (22). The objective $F^r_{\text{SR}}(X)$, referred to as the *caching gain* in [20], is monotone, non-negative, and submodular, while the set of constraints on X is a set of matroid constraints. As a result, for any r, there exist standard approaches for constructing a polynomial time approximation algorithm solving the corresponding maximization problem (23) within a $1 - 1/e$ factor from its optimal solution [9], [20], [45]. In addition, we show in [20] that an approximation algorithm based on a technique known as *pipage rounding* [36] can be converted into a distributed, adaptive version with the same approximation ratio.

As discussed in Section V, we also approach the joint routing and caching problem by casting it as an equivalent caching gain maximization problem. In contrast to the fixed routing setting, the objective function $C_{\text{SR}}(r, X)$, expressed as a function of both caching *and* routing strategies, is neither monotone nor supermodular, and there is no constant C such that the function $C - C_{\text{SR}}(r, X)$ is monotone and submodular. In addition, constraints (15) do *not form a matroid*. One of our main contributions is to show that, in spite of these issues, it is still possible to construct a constant approximation algorithm for the maximization of an appropriately defined caching gain; moreover, the intuition behind this algorithm leads again to a distributed, adaptive implementation, as in [20].

H. Greedy Routing Strategies

In the case of source routing, we identify two "greedy" deterministic routing strategies, that are often used in practice, and play a role in our analysis. We say that a global routing strategy r is a *route-to-nearest-server* (RNS) strategy if all paths it selects are least-cost paths to designated servers, irrespectively of cache contents. Formally, r is RNS if for all $(i, s) \in \mathcal{R}$, $r_{(i,s),p^*} = 1$ for some

$$p^* \in \arg\min_{p \in \mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_{k+1},p_k}, \tag{24}$$

while $r_{(i,s),p} = 0$ for all other $p \in \mathcal{P}_{(i,s)}$ s.t. $p \neq p^*$. Similarly, given a caching strategy X, we say that a global routing strategy r is *route-to-nearest-replica* (RNR) strategy if, for all $(i, s) \in \mathcal{R}$, $r_{(i,s),p^*} = 1$ for some

$$p^* \in \arg\min_{p \in \mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_{k+1},p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'},i}), \tag{25}$$

while $r_{(i,s),p} = 0$ for all other $p \in \mathcal{P}_{(i,s)}$ s.t. $p \neq p^*$. In contrast to RNS strategies, RNR strategies depend on the caching strategy X. Note that RNS and RNR strategies can be defined similarly in the context of hop-by-hop routing.

IV. Routing to Nearest Server Is Suboptimal

A simple approach, followed by most works that optimize caching separately from routing, is to always route requests to the nearest designated server storing an item (i.e., use an RNS strategy). It is therefore interesting to ask how this simple heuristic performs compared to a solution that attempts to solve (14) by *jointly* optimizing caching and routing. It is easy to see that RNS and, more generally, routing that ignores caching strategies, can lead to arbitrarily suboptimal solutions. In other words, routing to the nearest server can incur a cost that arbitrarily larger than the cost of a strategy (r, X) that is jointly optimized:

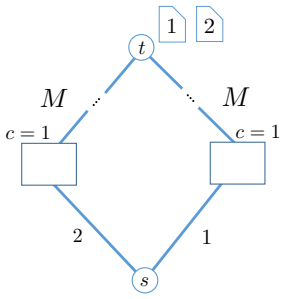


Fig. 2. A simple diamond network illustrating the benefits of path diversity. A source node s generates requests for items 1 and 2, permanently stored on designated server t. Intermediate nodes on the are two alternative paths towards t have capacity 1. Numbers above edges indicate costs. Under RNS, requests for both items are forwarded over the same path towards t, leading to a $\Theta(M)$ routing cost irrespective of the caching strategy. In contrast, the jointly optimal solution uses different paths per item, leading to an $O(1)$ cost.

Theorem 1: For any $M > 0$, there exists a caching network for which the route-to-nearest-server strategy r' satisfies

$$\min_{X:(r',X)\in\mathcal{D}_{\text{SR}}} C_{\text{SR}}(r', X) / \min_{(r,X)\in\mathcal{D}_{\text{SR}}} C_{\text{SR}}(r, X) \geq \frac{M+1}{2}. \tag{26}$$

Proof: Consider the simple diamond network shown in Fig. 2. A source node s generates requests for items 1 and 2 (i.e., $\mathcal{R} = \{(1, s), (2, s)\}$), that are permanently stored on designated server t, requesting each with equal rate $\lambda_{(1,s)} = \lambda_{(2,s)} = 1\text{sec}^{-1}$. The path sets $\mathcal{P}_{(i,s)}$, $i = 1, 2$, are identical, and consist of the two alternative paths towards t, each passing through an intermediate node with cache capacity 1 (i.e., able to store only one item). The two paths have routing costs $M+1$ and $M+2$, respectively. Under the route-to-nearest server strategy r', requests for both items are forwarded over the path of length $M+1$ towards t; fixing routes this way leads to a cost $M+1$ for at least one of the items. This happens irrespectively of which item is cached in the intermediate node. On the other hand, if routing and caching decisions are jointly optimized, requests for the two items can be forwarded to different paths, allowing both items to be cached in the nearby caches, and reducing the cost for both requests to at most 2. ■

The example in Fig. 2 illustrates that joint optimization of caching *and* routing decisions benefits the system by *increasing path diversity*. In turn, increasing path diversity can increase caching opportunities, thereby leading to reductions in caching costs. This is consistent with our experimental results in Section VIII.

V. Offline Source Routing

Motivated by the negative result of Thm. 1, we turn our attention to solving the offline problem MinCost-SR. As in the fixed-routing setting described in Section III-G, we first cast this as a maximization problem. Let C_0 be the constant:

$$C^0_{\text{SR}} = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{p\in\mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}. \tag{27}$$

Then, given a pair of strategies (r, X), we define the *expected caching gain* $F_{\text{SR}}(r, X)$ as follows:

$$F_{\text{SR}}(r, X) = C^0_{\text{SR}} - C_{\text{SR}}(r, X), \tag{28}$$

where C_{SR} is the aggregate routing cost given by (13). Note that $F_{\text{SR}}(r, X) \geq 0$. We seek to solve the following problem, equivalent to MINCOST-SR:

$$\text{MAXCG-S}$$
$$\text{Maximize:} \quad F_{\text{SR}}(r, X) \tag{29a}$$
$$\text{subj. to:} \quad (r, X) \in \mathcal{D}_{\text{SR}}. \tag{29b}$$

The selection of the constant C^0_{SR} is not arbitrary: this is precisely the value that allows us to approximate F_{SR} via the concave relaxation L_{SR} below (c.f. Lemma 2). Moreover, in Sec. VIII we show that, in spite of attaining approximation guarantees w.r.t. F_{SR} rather than C_{SR}, the resulting approximation algorithm has excellent performance in practice in terms of minimizing routing costs. In particular, we can reduce routing costs by a factor as high as 10^3 compared to fixed routing policies, including the one described in [20].

A. Offline Approximation Algorithm

Its equivalence to MINCOST-SR implies that MAXCG-S is also NP-hard. Nevertheless, we show that there exists a polynomial time approximation algorithm for MAXCG-S. Following [36], the technique for producing an approximation algorithm to solve MAXCG-S is to: (a) relax the combinatorial joint routing and caching problem to a convex optimization problem, (b) solve this convex relaxation, and (c) round the (possibly fractional) solution to obtain an integral solution to the original problem. To that end, consider the concave function $L_{\text{SR}} : \texttt{conv}(\mathcal{D}_{\text{SR}}) \to \mathbb{R}_+$, defined as:

$$L_{\text{SR}}(\rho, \Xi) = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{p\in\mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \cdot \ldots \min\{1, 1 - \rho_{(i,s),p} + \textstyle\sum_{k'=1}^{k} \xi_{p_{k'}i}\}. \tag{30}$$

Then, L_{SR} closely approximates F_{SR}:

Lemma 2: For all $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$,

$$(1 - 1/e)\, L_{\text{SR}}(\rho, \Xi) \leq F_{\text{SR}}(\rho, \Xi) \leq L_{\text{SR}}(\rho, \Xi).$$

Proof: This follows from the Goemans-Williamson inequality [20], [57], which states that, for any sequence of $y_i \in [0, 1]$, $i \in \{1, \ldots, n\}$:

$$(1-\frac{1}{e}) \min\{1, \sum_{i}^{n} y_i\} \leq 1 - \prod_{i=1}^{n}(1 - y_i) \leq \min\{1, \sum_{i=1}^{n} y_i\}. \tag{31}$$

The lower bound was proved by Goemans and Williamson (see Lemma 3.1 in [57], and Eq. (16) of Ageev and Sviridenko [36] for a shorter derivation). The upper bound follows easily from the concavity of of the min operator (see Thm. 2 of Ioannidis and Yeh [20]). To see this, let $t_i \in \{0, 1\}$, $i \in \{1, \ldots, n\}$, be independent Bernoulli random variables with expectations $\mathbb{E}[t_i] = y_i$. Then:

$$1 - \prod_{i=1}^{n}(1 - y_i) = \mathbf{P}[\sum_{i=1}^{n} t_i > 0] = \mathbb{E}[\min\{1, \sum_{i=1}^{n} t_i\}] \leq \min\{1, \sum_{i=1}^{n} \mathbb{E}[t_i]\} = \min\{1, \sum_{i=1}^{n} y_i\}$$

Algorithm 1 OFFLINE APPROXIMATION ALGORITHM

1: Find $(\rho^*, \Xi^*) \in \arg\max_{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{SR}})} L_{\text{SR}}(\rho, \Xi)$
2: Fix ρ^*, and round Ξ^* as in Lemma 3 to obtain integral, feasible X' s.t. $F_{\text{SR}}(\rho^*, X') \geq F_{\text{SR}}(\rho^*, \Xi^*)$
3: Fix X', and round ρ^* as in Lemma 4 to obtain integral, feasible r' s.t. . $F_{\text{SR}}(r', X') \geq F_{\text{SR}}(\rho^*, X')$
4: **return** (r', X')

where the inequality holds by Jensen's inequality and the fact that $\min\{1, \cdot\}$ is concave. The lemma therefore follows by applying inequality (31) to every term in the summation making up F_{SR}, to all variables ξ_{vi}, $v \in V$, $i \in C$, and $1 - \rho_{(i,s),p}$, $(i, s) \in \mathcal{R}$, $p \in \mathcal{P}_{(i,s)}$. ∎

Constructing a constant-approximation algorithm for MAXCG-S amounts to the following steps. *First,* obtain

$$(\rho^*, \Xi^*) \in \underset{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{SR}})}{\arg\max} L_{\text{SR}}(\rho, \Xi). \tag{32}$$

As L_{SR} is concave function and $\texttt{conv}(\mathcal{D}_{\text{SR}})$ is convex, the above maximization is a convex optimization problem. In fact, it can be reduced to a linear program [20], so it can be solved in polynomial time [58]. *Second,* round the (possibly fractional) solution $(\rho^*, \Xi^*) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ to an integral solution $(r, X) \in \mathcal{D}_{\text{SR}}$ such that $F_{\text{SR}}(r, X) \geq F_{\text{SR}}(\rho^*, \Xi^*)$. This rounding is deterministic and takes place in polynomial time. The above steps are summarized in Algorithm 1, for which the following theorem holds:

Theorem 2: Algorithm 1 terminates within a number of steps that is polynomial in $|V|$, $|C|$, and P_{SR}, and produces a strategy $(r', X') \in \mathcal{D}_{\text{SR}}$ such that

$$F_{\text{SR}}(r', X') \geq (1 - 1/e) \max_{(r,X)\in\mathcal{D}_{\text{SR}}} F_{\text{SR}}(r, X).$$

Proof: We first prove the following two auxiliary lemmas. First, a feasible fractional solution can be converted–in polynomial time–to a feasible solution in which only ρ is fractional, while increasing F_{SR}.

Lemma 3 ([9]): Given any $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$, an integral X such that $(\rho, X) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ and $F_{\text{SR}}(\rho, X) \geq F_{\text{SR}}(\rho, \Xi)$ can be constructed in $O(|V|^2|C|P_{\text{SR}})$ time.

Proof: This is proved in [9] for fixed routing strategies; for completeness, we repeat the proof here. Given a fractional solution $(\rho, \Xi) \in \mathcal{D}_{\text{SR}}$, there must exist a $v \in V$ that contains two fractional values $\xi_{vi}, \xi_{vi'}$, as capacities are integral. Restricted to these two variables, function F_{SR} is an affine function of $\xi_{vi}, \xi_{vi'}$. That is,

$$F_{\text{SR}}(\rho, \Xi) = A\xi_{vi} + B\xi_{vi'} + C \equiv F^{vi,vi'}_{\text{SR}}(\xi_{vi}, \xi_{vi'}),$$

where constants A, B, C depend on ρ and $\Xi_{-(vi,vi)}$ (the values of Ξ excluding $\xi_{vi}, \xi_{vi'}$), but not on $\xi_{vi}, \xi_{vi'}$. Hence, $F^{vi,vi'}_{\text{SR}}(\xi_{vi}, \xi_{vi'})$ is maximized at the extrema of the polytope in $\mathbb{R}^2$ implied by the capacity and $[0, 1]$ constraints involving variables $\xi_{vi}, \xi_{vi'}$ alone. Formally, consider the polytope:

$$\mathcal{D}^{(vi,vi')}_{\text{SR}}(\Xi_{-(vi,vi)}) = \{(\xi_{vi}, \xi_{vi'}) \in [0, 1]^2 : \sum_{j\in C} \xi_{vj} = c_v\} \subset \mathbb{R}^2.$$

Then, the optimization:

$$\max_{(\xi_{vi},\xi_{vi'})\in\mathcal{D}^{(vi,vi')}_{\text{SR}}(\Xi_{-(vi,vi)})} F^{vi,vi'}_{\text{SR}}(\xi_{vi}, \xi_{vi'})$$

has a solution that is an extremum of $\mathcal{D}_{\text{SR}}^{(vi,vi')}(\Xi_{-(vi,vi)})$. That is, there exists an optimal solution to this problem where either ξ_{vi} or $\xi_{vi'}$ is integral (either 0 or 1). Finding this solution amounts to testing two cases and seeing which of the two maximizes $F_{\text{SR}}^{vi,vi'}$ (and thereby also F_{SR}): (a) the case where a value $\delta = \min\{\xi_{vi}, 1-\xi_{vi'}\}$ is subtracted from ξ_{vi} and added to $\xi_{vi'}$, and (b) the case where $\delta' = \min\{1-\xi_{vi}, \xi_{vi'}\}$ is subtracted from $\xi_{vi'}$ and added to ξ_{vi}.

The above imply that there exists a way to transfer equal mass from one of the two fractional variables $\xi_{vi}, \xi_{vi'}$ to the other so that (a) one of them becomes integral (either 0 or 1), (b) the resulting Ξ' remains feasible, and (c) F_{SR} does not decrease.[2] Performing this transfer of mass reduces the number of fractional variables in Ξ by one, while maintaining feasibility and, crucially, either increasing F_{SR} or keeping it constant. This rounding can be repeated so long as Ξ remains fractional: this eliminates all fractional variables in at most $O(|V||C|)$ steps. Each step requires at most two evaluations of F_{SR} for each of the two cases, which can be done in $O(|V|P_{\text{SR}})$ time. Note that the pair of fractional variables selected each time is arbitrary: the order of elimination (i.e., the order with which pairs of fractional variables are rounded) leads to a different rounding, but all such roundings are (a) feasible and, (b) either increase F_{SR} or keep it constant. ∎

The routing strategy ρ can also be rounded in polynomial time, while keeping the caching strategy X fixed:

Lemma 4: Given any $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$, an integral r s.t. $(r, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ and $F_{\text{SR}}(r, \Xi) \geq F_{\text{SR}}(\rho, \Xi)$ can be constructed in $O(|V|P_{\text{SR}})$ time. Moreover, if Ξ is integral, then the resulting r is a route-to-nearest-replica (RNR) strategy.

Proof: Given $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$, notice that, for fixed Ξ, F_{SR} is an affine function of the routing strategy ρ. All coefficients involving variables $\rho_{(i,s),p}$, $p \in \mathcal{P}_{(i,s)}$, are non-negative, and the set of constraints on ρ is separable across requests $(i,s) \in \mathcal{R}$. Hence, given Ξ, maximizing F_{SR} w.r.t. ρ can be done by selecting the path $p^* \in \mathcal{P}_{(i,s)}$ with the highest coefficient of F_{SR}, for every $(i,s) \in \mathcal{P}$; this is precisely the lowest cost path, i.e., $p^*_{(i,s)} \in \mathcal{P}_{(i,s)}$ is such that

$$p^*_{(i,s)} = \operatorname*{arg\,min}_{p \in \mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_k p_k} \prod_{k'=1}^{k} (1 - \xi_{p_{k'} i}). \tag{33}$$

Hence, given Ξ, setting $\rho_{(i,s),p^*} = 1$, and $\rho_{(i,s),p} = 0$ for all remaining paths $p \in \mathcal{P}_{(i.s.)}$ s.t. $p \neq p^*$ can only increase F_{SR}. Each p^* can be computed in $O(|\mathcal{P}_{(i,s)}||V|)$ time and there is most $O(\mathcal{R})$ such paths. This results in an integral, feasible strategy r, and the resulting F_{SR} either increases or stays constant, i.e., $(r, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ and $F_{\text{SR}}(r, \Xi) \geq F_{\text{SR}}(\rho, \Xi)$. Finally, if $\Xi = X$ for some integral X, then the selection of each strategy p^* through (33) yields precisely a route-to-nearest-replica routing for (i,s). ∎

To conclude the proof of Theorem 2, note that the complexity statement is a consequence of Lemmas 3 and 4. By construction, the output of the algorithm (r', X') is such that: $F_{\text{SR}}(r', X') \geq F_{\text{SR}}(\rho^*, \Xi^*)$. Let

$$(r^*, X^*) \in \operatorname*{arg\,max}_{(r,X) \in \mathcal{D}_{\text{SR}}} F_{\text{SR}}(r, X)$$

be an optimal solution to MAXCG-S. Then, by Lemma 2 and the optimality of (ρ^*, X^*) in $\texttt{conv}(\mathcal{D}_{\text{SR}})$:

$$F_{\text{SR}}(r^*, X^*) \leq L_{\text{SR}}(r^*, X^*) \leq L_{\text{SR}}(\rho^*, \Xi^*) \leq \frac{e}{e-1} F_{\text{SR}}(\rho^*, \Xi^*).$$

Together, these imply that the constructed (r', X') is such that $F_{\text{SR}}(r', X') \geq (1 - 1/e) F_{\text{SR}}(r^*, X^*)$. ∎

B. Implications: RNS and an Equivalence Theorem

Lemma 4 has the following immediate implication:

Corollary 1: There exists an optimal solution (r^*, X^*) to MAXCG-S (and hence, to MINCOST-SR) in which r^* is an route-to-nearest-replica (RNR) strategy w.r.t. X^*.

Proof: Let (r^*, X^*) be an optimal solution to MAXCG-S in which r^* is not a RNR strategy. Then, by Lemma 4, we can construct an r' that is an RNR strategy w.r.t. X such that (a) $F_{\text{SR}}(r', X^*) \geq F_{\text{SR}}(r^*, X^*)$ and (b) $(r', X^*) \in \mathcal{D}_{\text{SR}}$. As (r^*, X^*) is optimal, so is (r', X^*). ∎

Although, in light of Thm. 1, Cor. 1 suggests an advantage of RNR over RNS strategies, its proof is non-constructive, not providing an algorithm to find an optimal solution, RNR or otherwise.

We can also show the following result regarding *randomized* strategies. For μ a probability distribution over $\mathcal{D}_{\text{SR}}$, let $\mathbb{E}_\mu[C_{\text{SR}}(r, X)]$ be the expected routing cost under μ. Then, the following equivalence theorem holds:

Theorem 3: The deterministic and randomized versions of MINCOST-SR attain the same optimal routing cost, i.e.:

$$\begin{aligned} \min_{(r,X) \in \mathcal{D}_{\text{SR}}} C_{\text{SR}}(r, X) &= \min_{(\rho,\Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})} C_{\text{SR}}(\rho, \Xi) \\ &= \min_{\mu : \text{supp}(\mu) = \mathcal{D}_{\text{SR}}} \mathbb{E}_\mu[C_{\text{SR}}(r, X)] \end{aligned} \tag{34}$$

Proof: Clearly,

$$\min_{(r,X) \in \mathcal{D}_{\text{SR}}} C_{\text{SR}}(r, X) \geq \min_{(\rho,\Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})} C_{\text{SR}}(\rho, \Xi)$$

as $\mathcal{D}_{\text{SR}} \subset \texttt{conv}(\mathcal{D}_{\text{SR}})$. Let

$$(\rho^*, \Xi^*) \in \operatorname*{arg\,min}_{(\rho,\Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})} C_{\text{SR}}(\rho, \Xi) = \operatorname*{arg\,max}_{(\rho,\Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})} F_{\text{SR}}(\rho, \Xi).$$

Then, Lemmas 3 and 4 imply that we can construct an integral $(r'', X'') \in \mathcal{D}_{\text{SR}}$ s.t.

$$F_{\text{SR}}(r'', X'') \geq F_{\text{SR}}(\rho^*, \Xi^*). \tag{35}$$

Hence,

$$\min_{(r,X) \in \mathcal{D}_{\text{SR}}} C_{\text{SR}}(r, X) \leq C_{\text{SR}}(r'', X'') \overset{(35)}{\leq} C_{\text{SR}}(\rho^*, \Xi^*),$$

and the first equality holds.

Note that for $\mu^* \in \arg\min_{\mu : \text{supp}(\mu) = \mathcal{D}_{\text{SR}}} \mathbb{E}_\mu[C_{\text{SR}}(r, X)]$, and $(r^*, X^*) = \arg\min_{(r,X) \in \mathcal{D}_{\text{SR}}} C_{\text{SR}}(r, X)$, we have that

$$\begin{aligned} \mathbb{E}_{\mu^*}[C_{\text{SR}}(r, X)] &= \min_{\mu : \text{supp}(\mu) = \mathcal{D}_{\text{SR}}} \mathbb{E}_\mu[C_{\text{SR}}(r, X)] \\ &\leq \min_{(r,X) \in \mathcal{D}_{\text{SR}}} C_{\text{SR}}(r, X) \\ &= C_{\text{SR}}(r^*, X^*), \end{aligned}$$

[2] This property is called ϵ-convexity by Ageev and Sviridenko [36].

as deterministic strategies are a subset of randomized strategies. On the other hand,

$$\begin{aligned}\mathbb{E}_{\mu^*}[C_{\text{SR}}(r,X)] &= \sum_{(r,X)\in\mathcal{D}_{\text{SR}}} \mu\left((r,X)\right) C_{\text{SR}}(r,X)\\ &\geq C_{\text{SR}}(r^*,X^*) \sum_{(r,X)\in\mathcal{D}_{\text{SR}}} \mu\left((r,X)\right)\\ &= C_{\text{SR}}(r^*,X^*).\end{aligned}$$

and the second equality also follows. ∎

The first equality of the theorem implies that, surprisingly, there is *no inherent advantage in randomization*: although randomized strategies constitute a superset of deterministic strategies, the optimal attainable routing cost (or, equivalently, caching gain) is the same for both classes. The second equality implies that assuming independent caching and routing strategies is as powerful as sampling routing and caching strategies from an arbitrary joint distribution. Thm. 3 generalizes Thm. 5 of [20], which pertains to optimizing caching alone.

VI. Online Source Routing

The algorithm in Thm. 2 is offline and centralized: it assumes full knowledge of the input, including demands and arrival rates, which are rarely available in practice. To that end, we turn our attention to solving MaxCG-S in the online setting, in the absence of any a priori knowledge of the demand. Our main contribution is to show that an expected caching gain within a constant approximation of the optimal solution to the offline problem MaxCG-S can be attained in steady state by a distributed, adaptive algorithm:

Theorem 4: There exists a distributed, adaptive algorithm constructing randomized strategies $(r^{(k)}, X^{(k)}) \in \mathcal{D}_{\text{SR}}$ at the k-th slot that satisfy

$$\lim_{k\to\infty} \mathbb{E}[F_{\text{SR}}(r^{(k)},X^{(k)})] \geq (1-1/e) \max_{(r,X)\in\mathcal{D}_{\text{SR}}} F_{\text{SR}}(r,X). \quad (36)$$

Note that, despite the fact that the algorithm has no prior knowledge of the demands, the guarantee provided is w.r.t. an optimal solution of the *offline* problem (29). Our algorithm naturally generalizes [20]: when the path sets $\mathcal{P}_{(i,s)}$ are singletons, and routing is fixed, our algorithm coincides with the cache-only optimization algorithm in [20]. Interestingly, the algorithm casts routing and caching *in the same control plane*: the same quantities are communicated through control messages to adapt both the caching and routing strategies.

A. Algorithm Overview

Before proving Thm. 4, we first give a brief overview of the distributed, adaptive algorithm that attains the approximation ratio of the theorem, and state its convergence guarantee precisely. Intuitively, the algorithm that attains the guarantees of Thm. 4 solves the problem:

$$\max_{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{SR}})} L_{\text{SR}}(\rho,\Xi), \quad (37)$$

where function $L_{\text{SR}} : \texttt{conv}(\mathcal{D}_{\text{SR}}) \to \mathbb{R}_+$ is the approximation of the caching gain F_{SR} given by (30). Recall that, in contrast to (20), (37) is a convex optimization problem by the concavity of L_{SR}. Our distributed adaptive algorithm effectively performs a *projected gradient ascent* to solve the convex relaxation (37) in a distributed, adaptive fashion. The concavity of L_{SR} ensures convergence, while Lemma 2 ensures that the caching gain attained in steady state is within an $1-\frac{1}{e}$ factor from the optimal.

In more detail, recall from III-E that, in the online setting, time is partitioned into slots of equal length $T > 0$. Caching and routing strategies are randomized as described in Sec. III: at the beginning of a timeslot, nodes place a random set of contents in their cache, independently of each other. During a timeslot, new requests are routed upon arrival over random paths, selected independently of (a) all past routes followed, and (b) of past and present caching decisions.

Nodes in the network maintain the following state information. Each node $v \in G$ maintains locally a vector $\xi_v \in [0,1]^{|C|}$, determining its randomized caching strategy. Moreover, for each request $(i,s) \in \mathcal{R}$, source node s maintains a vector $\rho_{(i,s)} \in [0,1]^{|\mathcal{P}_{(i,s)}|}$, determining its randomized routing strategy. Together, these variables represent the global state of the network, denoted by $(\rho,\Xi) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$. When the timeslot ends, each node performs the following four tasks:

1) Subgradient Estimation. Each node uses measurements collected during the duration of a timeslot to construct estimates of the gradient of L_{SR} w.r.t. its own local state variables. As L_{SR} is not everywhere differentiable, an estimate of a *subgradient* of L_{SR} is computed instead.

2) State Adaptation. Nodes adapt their local caching and routing state variables ξ_v, $v \in V$, and $\rho_{(i,s)}$, $(i,s) \in \mathcal{R}$, pushing them towards a direction that increases L_{SR}, as determined by the estimated subgradients, while maintaining feasibility in $\texttt{conv}(\mathcal{D}_{\text{SR}})$.

3) State Smoothening. Nodes compute "smoothened" versions $\bar{\xi}_v$, $v \in V$, and $\bar{\rho}_{(i,s)}$, $(i,s) \in \mathcal{R}$, interpolated between present and past states. This is needed on account of the non-differentiability of L_{SR}.

4) Randomized Caching and Routing. After smoothening, each node v reshuffles the contents of its cache using the smoothened caching marginals $\bar{\xi}_v$, producing a random placement (i.e., caching strategy x_v) to be used throughout the next slot. Moreover, each node $s \in V$ routes requests $(i,s) \in \mathcal{R}$ received during next timeslot over random paths (i.e., routing strategies $r_{(i,s)}$) sampled in an i.i.d. fashion from the smoothened marginals $\bar{\rho}_{(i,s)}$.

Pseudocode summarizing these steps of the algorithm is provided in Alg. 2.

Convergence Guarantees. Together, the four tasks above ensure that, in steady state, the expected caching gain of the jointly constructed routing and caching strategies is within a constant approximation of the optimal solution to the offline problem MaxCG-S. The proof of the convergence of the algorithm relies on the following key lemma, proved in Section VI-E:

Lemma 5: Let $(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)}) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ be the smoothened state variables at the k-th slot of Algorithm 2, and

$$(\rho^*,\Xi^*) \in \operatorname*{arg\,max}_{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{SR}})} L_{\text{SR}}(\rho,\Xi).$$

Then, for γ_k the step-size used in projected gradient ascent,

$$\varepsilon_k \equiv \mathbb{E}[L_{\text{SR}}(\rho^*, \Xi^*) - L_{\text{SR}}(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)})] \leq \frac{D^2 + M^2 \sum_{\ell=\lfloor k/2 \rfloor}^{k} \gamma_\ell^2}{2 \sum_{\ell=\lfloor k/2 \rfloor}^{k} \gamma_\ell},$$

where

$$D = \sqrt{2|V| \max_{v \in V} c_v + 2|\mathcal{R}|}, \text{ and}$$

$$M = W|V|\Lambda \sqrt{(|V||C|P_{\text{SR}}^2 + |\mathcal{R}|P_{\text{SR}})(1 + \frac{1}{\Lambda T})}.$$

In particular, if $\gamma_k = 1/\sqrt{k}$, then $\varepsilon_k = O(1/\sqrt{k})$.
Lemma 5 establishes that Algorithm 2 converges arbitrarily close to an optimizer of L_{SR}. As, by Lemma 2, this is a close approximation of F_{SR}, the limit points of the algorithm are with the $1 - 1/e$ from the optimal. Crucially, Lemma 5 can be used to determine the rate of convergence of the algorithm, by determining the number of steps required for ε_k to reach a desired threshold δ. Moreover, through quantity M, Lemma 5 establishes a tradeoff w.r.t. T: increasing T decreases the error in the estimated subgradient, thereby reducing the total number of steps till convergence, but also increases the time taken by each step.

The convergence guarantee in Lemma 5 holds under the assumption that (a) although unknown, demands are stationary, and (b) γ_k converges to zero. In practice, we would prefer that caches adapt to demand fluctuations. To achieve this, one would fix γ to a constant positive value, ensuring that Algorithm 2 tracks demand changes. Though convergence to a minimizer is not guaranteed in this case, the algorithm is nonetheless guaranteed to reach states concentrated around an optimal allocation (see, e.g., Chapter 8 of Kushner & Yin [59]).

In the remainder of this section we describe in detail the constituent four steps of the algorithm (namely, subgradient estimation, state adaptation, smoothening, and random sampling). These are presented in Sections VI-B to VI-D, respectively. We present proofs of Lemma 5 and of Thm. 4 in Sections VI-E and VI-F, respectively. Finally, we propose a modification that reduces overhead due to control messages in Section VI-G.

B. Subgradient Estimation

We now describe how to estimate the subgradients of L_{SR} through measurements collected during a timeslot. These estimates are computed in a distributed fashion at each node, using only information available from control messages traversing the node. Let $(\rho^{(k)}, \Xi^{(k)}) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ be the pair of global states at the k-th measurement period. At the conclusion of a timeslot, each $v \in V$ produces a random vector $z_v = z_v(\rho^{(k)}, \Xi^{(k)}) \in \mathbb{R}_+^{|C|}$ that is an unbiased estimator of a subgradient of L_{SR} w.r.t. to ξ_v. Similarly, for every $(i, s) \in \mathcal{R}$, source node s produces a random vector $q_{(i,s)} = q_{(i,s)}(\rho^{(k)}, \Xi^{(k)}) \in \mathbb{R}^{|\mathcal{P}_{(i,s)}|}$ that is an unbiased estimator of a subgradient of L_{SR} with respect to (w.r.t.) $\rho_{(i,s)}$. Formally,

$$\mathbb{E}[z_v(\rho^{(k)}, \Xi^{(k)})] \in \partial_{\xi_v} L_{\text{SR}}(\rho^{(k)}, \Xi^{(k)}), \quad (38)$$

$$\mathbb{E}[q_{(i,s)}(\rho^{(k)}, \Xi^{(k)})] \in \partial_{\rho_{(i,s)}} L_{\text{SR}}(\rho^{(k)}, \Xi^{(k)}), \quad (39)$$

where $\partial_{\xi_v} L_{\text{SR}}(\rho, \Xi)$, $\partial_{\rho_{(i,s)}} L_{\text{SR}}$ are the sets of subgradients of L_{SR} w.r.t. ξ_v and $\rho_{(i,s)}$, respectively. To produce these estimates, nodes measure the upstream cost incurred at paths passing through it using control messages, exchanged among nodes as follows:

1) Every time a node s generates a new request (i, s), it also generates additional control messages, one per path $p \in \mathcal{P}_{(i,s)}$. The message corresponding to path p is to be propagated over p, and contains a counter initialized to $1 - \rho_{(i,s),p} + \xi_{si}$.
2) When following path p, the message is forwarded until a node $u \in p$ s.t. $1 - \rho_{(i,s),p} + \sum_{\ell=1}^{k_p(u)} \xi_{p_\ell i} > 1$ is found, or the end of the path is reached. To keep track of this, every $v \in p$ traversed adds its state variable ξ_{vi} to the message counter.
3) Upon reaching either such a node u or the end of the path, the control message is sent down in the reverse direction. Initializing its counter to zero, every time it traverses an edge in this reverse direction, it adds the weight of this edge into a weight counter.
4) Every node on the reverse path "sniffs" the weight counter of the control message, learning the sum of weights of all edges further upstream towards u; that is, recalling that $k_p(v)$ is the position of visited node $v \in p$, v learns the quantity:
$$t_{vi} = \sum_{k'=k_v(p)}^{|p|-1} w_{p_{k'+1}p_{k'}} \mathbb{1}(1 - \rho_{(i,s),p} + \sum_{\ell=1}^{k'} \xi_{p_\ell i} \leq 1), \quad (40)$$
where $\mathbb{1}(E)$ is 1 if and only if E is true and 0 o.w.
5) In addition, the source s of the request, upon receiving the message sent over the reverse path, "sniffs" the quantity:
$$t_{(i,s),p} = -\sum_{k'=1}^{|p|-1} w_{p_{k'+1}p_{k'}} \mathbb{1}(1 - \rho_{(i,s),p} + \sum_{\ell=1}^{k'} \xi_{p_\ell i} \leq 1). \quad (41)$$
This is the (negative of) the sum of weights accumulated by the control message returning to the source s.

An example illustrating the above five steps can be found in Fig. 3. Let $\mathcal{T}_{vi}$ be the set of quantities collected in this way at node v regarding item $i \in C$ during a measurement period of duration T. At the end of the timeslot, each node $v \in V$ produces z_v as follows:

$$z_{vi} = \textstyle\sum_{t \in \mathcal{T}_{vi}} t/T, \quad i \in C. \quad (42)$$

Similarly, let $\mathcal{T}_{(i,s),p}$ be the set of quantities collected in this way at source node s regarding path $p \in \mathcal{P}_{(i,s)}$ during a measurement period of duration T. At the end of the measurement period, s produces the estimate $q_{(i,s)}$:

$$q_{(i,s),p} = \textstyle\sum_{t \in \mathcal{T}_{(i,s),p}} t/T, \quad i \in C. \quad (43)$$

We show that the resulting z_v, $q_{(i,s)}$ satisfy (38) and (39), respectively, in Lemma 6.

In the above construction, control messages are sent over all paths in $\mathcal{P}_{(i,s)}$. It is important to note however that when sent over paths p such that $\rho_{(i,s),p} \approx 0$ control messages *do not travel far*: the termination condition (the sum exceeding

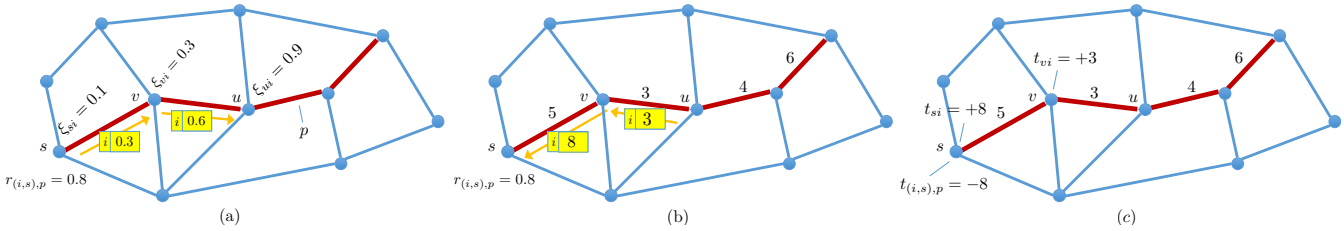


Fig. 3. Example of control message trajectory over a path. When node s generates a request $(i, s) \in \mathcal{R}$, it also generates a control message for every path $p \in \mathcal{P}_{(i,s)}$, indicated by thick red edges. In (a), the control message counter is initialized to $(1 - r_{(i,s),p}) + \xi_{si} = 1 - 0.8 + 0.1 = 0.3$ by s. It is is forwarded upstream on p to node v, that adds its own caching state variable w.r.t. item i, namely $\xi_{ui} = 0.3$, to the counter. As the sum is below 1.0, the message is forwarded upstream, until it reaches node u with $\xi_{ui} = 0.9$. As the total sum is now $1.5 > 1.0$, the propagation over p terminates, and a response is sent downstream by u. The response is shown in (b), accumulating the weights of edges it traverses. Nodes in its path, namely v and s, sniff this information, as shown in (c), and collect measurements t_{vi}, t_{si} to be added to the averages estimating $\partial_{\xi_{ui}} L_{\text{SR}}$ and $\partial_{\xi_{si}} L_{\text{SR}}$, respectively. The source s also collects measurement $t_{(i,s),p} = -t_{si}$, to be used in the average estimating $\partial_{\rho_{(i,s),p}} L$.

1) is satisfied early on. Messages sent over unlikely paths are thus pruned early, and "deep" propagation only happens in very likely paths. Nevertheless, to reduce control traffic, in Section VI-G we modify the algorithm to propagate only *a single control message over a single path*.

C. State Adaptation and Smoothening

Having estimates $Z = [z_v]_{v \in V}, q = [q_{(i,s)}]_{(i,s) \in \mathcal{R}}$, the global state is adapted as follows: at the conclusion of the k-th period, the new state $(\rho^{(k+1)}, \Xi^{(k+1)})$ is computed as:

$$\mathcal{P}_{\texttt{conv}(\mathcal{D}_{\text{SR}})}(\rho^{(k)} + \gamma_k q(\rho^{(k)}, \Xi^{(k)}), \Xi^{(k)} + \gamma_k Z(\rho^{(k)}, \Xi^{(k)})), \quad (44)$$

where $\gamma_k = 1/\sqrt{k}$ is a gain factor and $\mathcal{P}_{\texttt{conv}(\mathcal{D}_{\text{SR}})}$ is the orthogonal projection onto the convex set $\texttt{conv}(\mathcal{D}_{\text{SR}})$. Note that this additive adaptation and corresponding projection is separable across nodes and can be performed in a distributed fashion: each node $v \in V$ adapts its own relaxed caching strategy, each source s adapts its routing strategy, and all nodes project these strategies to their respective local constraints implied by (15b),(15a), and the $[0, 1]$ constraints. Note that these involve projections onto the rescaled simplex, for which well-known linear algorithms exist [60]. Upon performing the state adaptation (44), each node $v \in V$ and each source s, for $(i, s) \in \mathcal{R}$, compute the following "sliding averages" of current and past states:

$$\bar{\xi}_v^{(k)} = \textstyle\sum_{\ell=\lfloor \frac{k}{2} \rfloor}^{k} \gamma_\ell \xi_v^{(\ell)} / [\sum_{\ell=\lfloor \frac{k}{2} \rfloor}^{k} \gamma_\ell]. \quad (45)$$

$$\bar{\rho}_s^{(k)} = \textstyle\sum_{\ell=\lfloor \frac{k}{2} \rfloor}^{k} \gamma_\ell \rho_v^{(\ell)} / [\sum_{\ell=\lfloor \frac{k}{2} \rfloor}^{k} \gamma_\ell]. \quad (46)$$

This is necessary because of the non-differentiability of L_{SR} [61]. Note that $(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)}) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$, as a convex combination of elements of $\texttt{conv}(\mathcal{D}_{\text{SR}})$.

D. Randomized Caching and Routing.

The resulting $(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)})$ determine the randomized routing and caching strategies at each node during a timeslot. First, given $\bar{\rho}^{(k)}$, each time a request (i, s) is generated, path $p \in \mathcal{P}_{(i,s)}$ is used to route the request with probability $\bar{\rho}_{(i,s),p}$, independently of past routing and caching decisions. Second, given $\bar{\xi}_v^{(k)}$, each node $v \in V$ reshuffles its contents, placing items in its cache independently of all other nodes: that is, node v selects a random strategy $x_v^{(k)} \in \{0, 1\}^{|C|}$ sampled independently of any other node in V.

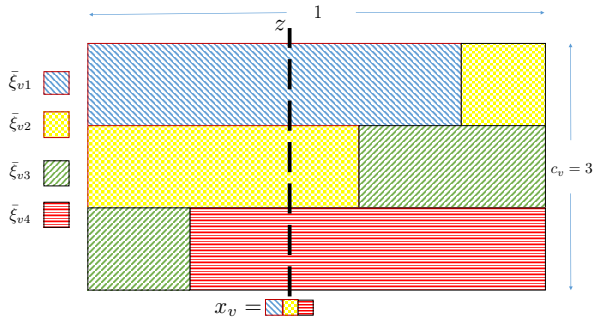


Fig. 4. Construction of a feasible randomized caching strategy x_v that satisfies marginals $\mathbf{P}[x_{vi} = 1] = \bar{\xi}_{vi}$, where $\sum_{i \in C} \bar{\xi}_{vi} = c_v$. In this example, $c_v = 3$, and $C = \{1, 2, 3, 4\}$. Given $\bar{\xi}_v$, 4 rectangles of height 1 each are constructed, such that the i-th rectangle has length $\bar{\xi}_{vi} \in [0, 1]$, and the total length is c_v. After placing the 4 rectangles in a 3×1 box, cutting the box at z selected u.a.r. from $[0, 1]$, and constructing a triplet of items from the rectangles it intersects, leads to an integral caching strategy with the desired marginals.

The random strategy $x_v^{(k)}$ satisfies the following two properties: (a) it is a *feasible* strategy, i.e., satisfies the capacity and integrality constraints (15a) and (15c), and (b) it is *consistent* with the marginals $\bar{\xi}_v^{(k)}$, i.e., for all $i \in C$, $\mathbb{E}[x_{vi}^{(k)} \mid \bar{\xi}_v^{(k)}] = \bar{\xi}_{vi}^{(k)}$. We note that there can be many random caching strategies whose distributions satisfy the above two properties. An efficient algorithm generating such a distribution is provided in [20] and, independently, in [56]. Given $\bar{\xi}_v^{(k)}$, a distribution over (deterministic) caching strategies can be computed in $O(c_v |C| \log |C|)$ time, and has $O(|C|)$ support; for the sake of completeness, we briefly outline this below. We follow the high-level description of [56] here; a detailed, formal description of the algorithm, a proof of its correctness, and a computational complexity analysis, can be found in [20].

The input to the algorithm are the marginal probabilities $\bar{\xi}_{vi} \in [0, 1]$, $i \in C$ s.t. $\sum_{i \in C} \bar{\xi}_{vi} = c_v$, where $c_v \in \mathbb{N}$ is the capacity of cache v. To construct a randomized caching strategy with the desired marginal distribution, consider a rectangle box of area $c_v \times 1$, as illustrated in Fig. 4. For each $i \in C$,

Algorithm 2 PROJECTED GRADIENT ASCENT

Execute the following for each $v \in V$ and each $(i,s) \in \mathcal{R}$:
Pick arbitrary state $(\rho^{(0)}, \Xi^{(0)}) \in \texttt{conv}(\mathcal{D}_{\texttt{SR}})$.
for each timeslot $k \geq 1$ **do**
  for each $v \in V$ **do**
    Compute the sliding average $\bar{\xi}_v^{(k)}$ through (45).
    Sample a feasible $x_v^{(k)}$ from a distribution with marginals $\bar{\xi}_v^{(k)}$.
    Place items $x_v^{(k)}$ in cache.
    Collect measurements and, at the end of the timeslot, compute estimate z_v f $\partial_{\xi_v} L_{\texttt{SR}}(\rho^k, \Xi^{(k)})$ through (42).
    Adapt $\xi_v^{(k)}$ through (44) to new state $\xi_v^{(k+1)}$ in the direction of the gradient with step-size γ_k, projecting back to $\texttt{conv}(\mathcal{D}_{\texttt{SR}})$.
  end for
  for each $(i,s) \in \mathcal{R}$ **do**
    Compute the sliding average $\bar{\rho}_{(i,s)}^{(k)}$ through (46).
    Whenever a new request arrives, sample $p \in \mathcal{P}_{(i,s)}$ from distribution $\bar{\rho}_{(i,s)}^{(k)}$.
    Collect measurements and, at the end of the timeslot, compute estimate $q_{(i,s)}$ of $\partial_{\rho_{(i,s)}} L_{\texttt{SR}}(\rho^k, \Xi^{(k)})$ through (43).
    Adapt $\rho_{(i,s)}^{(k)}$ through (44). to new state $\rho_{(i,s)}^{(k+1)}$ in the direction of the gradient with step-size γ_k, projecting back to $\texttt{conv}(\mathcal{D}_{\texttt{SR}})$.
  end for
end for

place a rectangle of length $\bar{\xi}_{vi}$ and height 1 inside the box, starting from the top left corner. If a rectangle does not fit in a row, cut it, and place the remainder in the row immediately below, starting again from the left. As $\sum_{i \in C} \bar{\xi}_{vi} = c_v$, this space-filling method tessellates the $c_v \times 1$ box. The randomized placement then is constructed as follows: select a value in $z \in [0,1]$ uniformly at random, and "cut" the box at position z. The value will intersect exactly c_v distinct rectangles: as $\bar{\xi}_{vi} \leq 1$, no rectangle "overlaps' with itself. The algorithm then produces as output the caching strategy $x_v \in \{0,1\}^{|C|}$ where:

$$x_{vi} = \begin{cases} 1, & \text{if the line intersects rectangle } i, \\ 0, & \text{o.w.} \end{cases}$$

As the line intersects c_v distinct rectangles, $\sum_{i \in C} x_{vi} = c_v$, so the caching strategy is indeed feasible. On the other hand, by construction, the probability that $x_{vi} = 1$ is exactly equal to the length of the i-th rectangle, so the marginal probability that i is placed in the cache is indeed $\mathbf{P}[x_{vi} = 1] = \bar{\xi}_{vi}$, and the randomized cache strategy x_v has the desired marginals.

E. Proof of Lemma 5

We first show that (42) and (43) are unbiased estimators of the subgradient:

Lemma 6: The vectors z_v, $v \in V$, and $q_{(i,s)}$, $(i,s) \in \mathcal{R}$ constructed through coordinates (42) and (43), satisfy:

$$\mathbb{E}[z_v] \in \partial_{\xi_v} L_{\texttt{SR}}(\rho, \Xi), \quad \text{and} \quad \mathbb{E}[q_{(i,s)}] \in \partial_{\xi_v} L_{\texttt{SR}}(\rho, \Xi).$$

Moreover, $\mathbb{E}[\|z_v\|_2^2] < C_1$, and $\mathbb{E}[\|q_{(i,s)}\|_2^2] < C_2$, where

$$C_1 = W^2 \bar{P}^2 |V|^2 |C| (\Lambda^2 + \frac{\Lambda}{T}),$$
$$C_2 = W^2 |V|^2 P (\Lambda^2 + \frac{\Lambda}{T}),$$

and constants $W, \bar{P}$, and Λ are given by:

$$W = \max_{(i,j) \in E} w_{ij},$$
$$\bar{P} = \max_{(i,s) \in \mathcal{R}} |\mathcal{P}_{(i,s)}|, \text{ and}$$
$$\Lambda = \sum_{(i,s) \in \mathcal{R}} \lambda_{(i,s)}.$$

Proof: A vector $\zeta \in \mathbb{R}^{|C|}$ belongs to $\partial_{\xi_v} L_{\texttt{SR}}(\rho, \Xi)$ if and only if $\zeta_i \in [\underline{\partial_{\xi_{vi}} L}_{\texttt{SR}}(\rho, \Xi), \overline{\partial_{\xi_{vi}} L}_{\texttt{SR}}(\rho, \Xi)]$, where:

$$\overline{\partial_{\xi_{vi}} L}_{\texttt{SR}}(\rho, \Xi) = \sum_{(i,s) \in \mathcal{R}} \lambda_{(i,s)} \sum_{p \in \mathcal{P}_{(i,s)}} \mathbb{1}_{v \in p} \cdot \sum_{k'=k_p(v)}^{|p|-1} w_{p_{k'+1} p_{k'}} \mathbb{1}_{1 - \rho_{(i,s)} + \sum_{\ell=1}^{k'} \xi_{p_\ell i} \leq 1},$$

$$\underline{\partial_{\xi_{vi}} L}_{\texttt{SR}}(\rho, \Xi) = \sum_{(i,s) \in \mathcal{R}} \lambda_{(i,s)} \sum_{p \in \mathcal{P}_{(i,s)}} \mathbb{1}_{v \in p} \cdot \sum_{k'=k_p(v)}^{|p|-1} w_{p_{k'+1} p_{k'}} \mathbb{1}_{1 - \rho_{(i,s)} + \sum_{\ell=1}^{k'} \xi_{p_\ell i} < 1}.$$

If $L_{\texttt{SR}}$ is differentiable at (ρ, Ξ) w.r.t ξ_{vi}, the two limits coincide and are equal to $\frac{\partial L_{\texttt{SR}}}{\partial_{vi}}$. It immediately follows from the fact that requests are Poisson that $\mathbb{E}[z_{vi}(\rho, \Xi)] = \overline{\partial_{\xi_{vi}} L}_{\texttt{SR}}(\rho, \Xi)$, so indeed $\mathbb{E}[z_v(Y)] \in \partial_{\xi_v} L_{\texttt{SR}}(\rho, \Xi)$. To prove the bound on the second moment, note that, for T_{vi} the number of requests generated for i that pass through v during the slot, $\mathbb{E}[z_{vi}^2] = \frac{1}{T^2} \mathbb{E}[(\sum_{t \in \mathcal{T}_{vi}} t)^2] \leq \frac{W^2 \bar{P}^2 |V|^2}{T^2} \mathbb{E}\left[T_{vi}^2\right]$, as $t \leq W \bar{P} |V|$. On the other hand, T_{vi} is Poisson distributed with expectation

$$\sum_{(i,s) \in \mathcal{R}} \mathbb{1}_{\exists p \in \mathcal{P}_{(i,s)} \text{ s.t. } v \in p} \lambda_{(i,s)} T,$$

and the upper bound follows. The statement for $q_{(i,s)}$ follows similarly. ∎

We now establish the convergence of the smoothened marginals to a global maximizer of L. Under (44), (45) and (46), from Theorem 14.1.1, page 215 of Nemirofski [61], we have that

$$\varepsilon_k \leq \frac{D^2 + M^2 \sum_{\ell=\lfloor k/2 \rfloor}^{k} \gamma_\ell^2}{2 \sum_{\ell=\lfloor k/2 \rfloor}^{k} \gamma_\ell},$$

where $\gamma_k = \frac{1}{\sqrt{k}}$,

$$D \equiv \max_{x,y \in \texttt{conv}(\mathcal{D}_{\texttt{SR}})} \|x - y\|_2 = \sqrt{|V| \max_v 2c_v + 2|\mathcal{R}|},$$

and

$$M \equiv \sup_{(\rho,\Xi)} \sqrt{\mathbb{E}[\|Z(\rho,\Xi)\|_2^2] + \mathbb{E}[\|q(\rho,\Xi)\|_2^2]}.$$

From Lem. 6, $M \leq \sqrt{|V| C_1 + |\mathcal{R}| C_2}$, and Lemma 5 follows. ∎

F. Proof of Theorem 4

By construction, conditioned on $(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)})$, the $|V| + |\mathcal{R}|$ variables x_v, $v \in V$, and $r_{(i,s)}$, (i,s), are independent. Hence, conditioned on $(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)})$, all monomial terms of F_{SR} involve independent random variables. Hence,

$$\mathbb{E}[F_{\text{SR}}(r^{(k)}, X^{(k)}) \mid \bar{\rho}^{(k)}, \bar{\Xi}^{(k)}] = F_{\text{SR}}(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)}),$$

and, in turn,

$$\lim_{k\to\infty} \mathbb{E}[F_{\text{SR}}(r^{(k)}, X^{(k)})] = \lim_{k\to\infty} \mathbb{E}[F_{\text{SR}}(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)})].$$

Lemma 5 implies that, for $\nu^{(k)}$ the distribution of $(\bar{\rho}^{(k)}, \bar{\Xi}^{(k)})$, and Ω the set of $(\rho^*, \Xi^*) \in \texttt{conv}(\mathcal{D}_{\text{SR}})$ that are maximizers of L_{SR},

$$\lim_{k\to\infty} \nu^{(k)}(\texttt{conv}(\mathcal{D}_{\text{SR}}) \setminus \Omega) = 0.$$

By Lemma 2, $F_{\text{SR}}(\rho^*, \Xi^*) \geq (1-1/e) \max_{(r,X)\in\mathcal{D}_{\text{SR}}} F_{\text{SR}}(r, X)$, for any $(\rho^*, \Xi^*) \in \Omega$. The theorem follows from the above observations, and the fact that F_{SR} is bounded in $\texttt{conv}(\mathcal{D}_{\text{SR}}) \setminus \Omega$. ∎

G. Reducing Control Traffic.

Control messages generated by the protocol can be reduced by modifying the algorithm to propagate only a single control message over a single path with each request. The path is selected uniformly at random over paths in the support of $\rho_{(i,s)}$. That is, when a request $(i,s) \in \mathcal{R}$ arrives at s, a single control message is propagated over p selected uniformly at random from $\texttt{supp}(\rho_{(i,s)}) = \{p \in \mathcal{P}_{(i,s)} : \rho_{(i,s),p} > 0\}$. This reduces the number of control messages generated by s by at least a $c = |\texttt{supp}(\rho_{(i,s)})|$ factor. To ensure that (38) and (39) hold, it suffices to rescale measured upstream costs by c. To do this, the (single) control message contains an additional field storing c. When extracting weight counters from downwards packets, nodes on the path compute $t'_{vi} = c \cdot t_{vi}$, and $t'_{(i,s),p} = c \cdot t_{(i,s)p}$, where t_{vi}, $t_{(i,s),p}$ are as in (40) and (41), respectively. This randomization reduces control traffic, but increases the *variance* of subgradient estimates, also by a factor of c. This, in turn, slows down the algorithm convergence; this tradeoff can be quantified through, e.g., the constants in Lemma 5.

VII. Hop-by-Hop Routing

The proofs for the hop-by-hop setting are similar, *mutatis-mutandis*, as the proofs of the source routing setting. As such, in our exposition below, we focus on the main technical differences between the algorithms for the two settings.

Offline Setting. Define the constant:

$$C^0_{\text{HH}} = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{(u,v)\in G^{(i,s)}} w_{vu} |\mathcal{P}^u_{(i,s)}|.$$

Using this constant, we define the caching gain maximization problem to be:

MaxCG-HH

$$\text{Maximize:} \quad F_{\text{HH}}(r, X) \tag{47a}$$

$$\text{subj. to:} \quad (r, X) \in \mathcal{D}_{\text{HH}} \tag{47b}$$

where $F_{\text{HH}}(r, X) = C^0_{\text{HH}} - \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} C^{(i,s)}_{\text{HH}}(r, X)$ is the expected caching gain. This is again an NP-hard problem, equivalent to (17). We can again construct a constant approximation algorithm for MaxCG-HH:

Theorem 5: There exists an algorithm that terminates within a number of steps that is polynomial in $|V|$, $|C|$, and P_{HH}, and produces a strategy $(r', X') \in \mathcal{D}_{\text{HH}}$ such that

$$F_{\text{HH}}(r', X') \geq (1-1/e) \max_{(r,X)\in\mathcal{D}_{\text{HH}}} F_{\text{HH}}(r, X).$$

Proof: Consider the function

$$L_{\text{HH}}(\rho, \Xi) = \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{(u,v)\in G^{(i,s)}} \sum_{p\in\mathcal{P}^u_{(i,s)}} w_{vu} \cdot \min\{1, 1-\rho^{(i)}_{uv}+\xi_{ui}+\textstyle\sum_{k'=1}^{|p|-1}(1-\rho^{(i)}_{p_{k'}p_{k'+1}}+\xi_{p_{k'}i})\}.$$

As in Lemma 2, we can show that this concave function approximates F_{HH}, in that for all $(\rho, \Xi) \in \texttt{conv}(\mathcal{D}_{\text{HH}})$:

$$(1-1/e) L_{\text{HH}}(\rho, \Xi) \leq F_{\text{HH}}(\rho, \Xi) \leq L_{\text{HH}}(\rho, \Xi).$$

To construct a constant approximation solution, first, a fractional solution

$$(\rho^*, \Xi^*) = \operatorname*{arg\,max}_{(\rho,\Xi)\in\texttt{conv}(\mathcal{D}_{\text{HH}})} L_{\text{HH}}(\rho, \Xi),$$

can be obtained. This again involves a convex optimization, which can be reduced to a linear program. Subsequently, the solution can be rounded to obtain an integral solution $(r, X) \in \mathcal{D}_{\text{SR}}$ such that $F_{\text{HH}}(r, X) \geq F_{\text{HH}}(\rho^*, \Xi^*)$. Rounding Ξ^* follows the same steps as for source routing. To round ρ^*, one first rounds each node's strategy individually, i.e., for every $v \in V$ and every $i \in C$, we would pick the neighbor that maximizes the objective. This again follows from the fact that, given a caching strategy Ξ, and given the routing strategies of all other nodes, F_{HH} is an affine function of $\{r^{(i)}_{uv}\}_{v:(uv)\in E^{(i)}}$, for all $u \in V$, with positive coefficients. Hence, keeping everything else fixed, if each node chooses a cost minimizing decision, this will round its strategy, and all nodes in V can do this sequentially. The DAG property ensures that all requests eventually reach a designated server, irrespectively of the routing strategies resulting from the rounding decisions. ∎

Online Setting. Finally, as in the case of source routing, we can provide a distributed, adaptive algorithm for hop-by-hop routing as well.

Theorem 6: There exists a distributed, adaptive algorithm under which the randomized strategies sampled during the k-th slot $(r^{(k)}, X^{(k)}) \in \mathcal{D}_{\text{HH}}$ satisfy

$$\lim_{k\to\infty} \mathbb{E}[F_{\text{HH}}(r^{(k)}, X^{(k)})] \geq (1-1/e) \max_{(r,X)\in\mathcal{D}_{\text{SR}}} F_{\text{HH}}(r, X).$$

Proof: A distributed algorithm can be constructed by performing projected gradient ascent over L_{HH}. Beyond the same caching state variables ξ_v stored at each node $v \in V$, each node $v \in V$ maintains routing state variables

$$\rho^{(i)}_u = [\rho^{(i)}_{uv}]_{v:(u,v)\in E^{(i)}} \in [0,1]^{|E^{(i)}|},$$

for each $i \in C$, containing the marginal probabilities $\rho^{(i)}_{uv}$ that u routes request message for item i towards $v \in E^{(i)}$. Time is slotted, and nodes perform subgradient estimation, state adaptation, state smoothening, and randomized sampling

of caching and routing strategies. As the last three steps are nearly identical to source routing, we focus below on how to estimate subgradients, which is the key difference between the two algorithms.

Whenever a request $(i,s) \in \mathcal{R}$ is generated, a control message is propagated in all neighbors of s in $E^{(i)}$. These messages contain counters initialized to $1 - \rho^{(i)}_{sv} + \xi_{si}$. Each node $v \in V$ receiving such a message generates one copy for each of its neighbors in $E^{(i)}$. For each neighbor u, v adds $1-\rho^{(i)}_{vu}+\xi_{vi}$ to the counter, and forwards the message to u if the counter is below 1.0. Formally, a control message originating at s and reaching a node v after having followed path $p \in G^{(i,s)}$ is forwarded to u if the following condition is satisfied:

$$1 - \rho^{(i)}_{vu} + \xi_{ui} + \sum_{\ell=1}^{k_p(v)-1} (1 - \rho^{(i)}_{p_\ell p_{\ell+1}} + \xi_{p_\ell i}) \leq 1.$$

If this condition is met, v forwards a copy of the control message to u; the above process is repeated at each of each neighbors. If the condition fails *for all* neighbors, a response message is generated by v and propagated over the reverse path, accumulating the weights of edges it passes through. Moreover, descending control messages are merged as follows. Each node v waits for all responses from neighbors to which it has sent control messages; upon the last arrival, it adds their counters, and sends the "merged" message containing the accumulated counter reversely over path p.

As before, messages on the return path are again "sniffed" by nodes they pass through, extracting the upstream costs. Their averages are used as estimators of the subgradients w.r.t. both the local routing and caching states, in a manner similar to how this was performed in source routing. As each edge is traversed at most twice, the maximum number of control messages is $O(|E^{(i)}|)$. As in the case of source routing, however, messages on low-probability paths are pruned early. Moreover, as in Section VI-G, only a single message need be propagated to a neighbor selected uniformly at random; in this case, the message needs to also contain a field keeping track of the product of the size of neighborhoods of nodes it has passed through, and updated by each node by multiplying the entry by the size of its own neighborhood. As in source routing, this is used as an additional scaling factor for quantities $t^{(i)}_{vu}$, t_{vi}. ∎

We note again that the distributed, adaptive algorithm attains an expected caching gain within a constant approximation from the *offline* optimal.

VIII. Evaluation

We simulate Alg. 2 over both synthetic and real networks. We compare its performance to traditional caching policies, combined with both static and dynamic multi-path routing.

Experiment Setup. We consider the topologies in Table II. For each graph $G(V,E)$, we generate a catalog of size $|C|$, and assign to each node $v \in V$ a cache of capacity c_v. For every item $i \in C$, we designate a node selected u.a.r. from V as a designated server for this item; the item is stored outside the designate server's cache. We assign a weight to each edge in E selected u.a.r. from the interval $[1, 100]$. We also select a random set of Q nodes as the possible request sources, and generate a set of requests $\mathcal{R} \subseteq C \times V$ by sampling exactly $|\mathcal{R}|$ from the set $C \times Q$, uniformly at random. For each such request $(i,s) \in \mathcal{R}$, we select the request rate $\lambda_{(i,s)}$ according to a Zipf distribution with parameter 1.2; these are normalized so that average request rate over all $|Q|$ sources is 1 request per time unit. For each request $(i,s) \in \mathcal{R}$, we generate $|\mathcal{P}_{(i,s)}|$ paths from the source $s \in V$ to the designated server of item $i \in C$. This path set includes the shortest path to the designated server. We consider only paths with stretch at most 4.0; that is, the maximum cost of a path in $P_{(i,s)}$ is at most 4 times the cost of the shortest path to the designated source. The values of $|C|$, $|\mathcal{R}|$ $|Q|$, c_v, and $\mathcal{P}_{(i,s)}$ for each G are given in Table II.

Online Caching and Routing Algorithms. We compare the performance of our joint caching and routing projected gradient ascent algorithm (`PGA`) to several competitors. In terms of caching, we consider four traditional eviction policies for comparison: Least-Recently-Used (`LRU`), Least-Frequently-Used (`LFU`), First-In-First-Out (`FIFO`), and Random Replacement (`RR`). We combine these policies with path-replication [2], [12]: once a request for an item reaches a cache that stores the item, every cache in the reverse path on the way to the query source stores the item, evicting stale items using one of the above eviction policies. We combine the above caching policies with three different routing policies. In route-to-nearest-server (`-S`), only the shortest path to the nearest designated server is used to route the message. In uniform routing (`-U`), the source s routes each request (i,s) on a path selected uniformly at random among all paths in $\mathcal{P}_{(i,s)}$. We combine each of these (static) routing strategies with each of the above caching strategies use. For instance, `LRU-U` indicates `LRU` evictions combined with uniform routing. Note that `PGA-S`, i.e., our algorithm restricted to RNS routing, is exactly the single-path routing algorithm proposed in [20]. To move beyond static routing policies for `LRU`, `LFU`, `FIFO`, and `RR`, we also combine the above traditional caching strategies with an adaptive routing strategy, akin to our algorithm, with estimates of the expected routing cost at each path used to adapt routing strategies. During a slot, each source node s maintains an average of the routing cost incurred when routing a request over each path. At the end of the slot, the source decreases the probability $\rho_{(i,s),p}$ that it will follow the path p by an amount proportional to the average, and projects the new strategy to the simplex. For fixed caching strategies, this dynamic routing scheme converges to a route-to-nearest-replica (RNS) routing, which we expect by Cor. 1 to have good performance. We denote this routing scheme with the extension `-D`. Note that all algorithms we simulate are online.

Experiments and Measurements. Each experiment consists of a simulation of the caching and routing policy, over a specific topology, for a total of 5000 time units. To leverage PASTA, we collect measurements during the duration of the execution at exponentially distributed intervals with mean 1.0 time unit. At each measurement epoch, we extract the current cache contents in the network and construct $X \in \{0,1\}^{|V|\times|C|}$. Similarly, we extract the current routing strategies $\rho_{(i,s)}$ for all requests $(i,s) \in \mathcal{R}$, and construct the global routing strategy $\rho \in [0,1]^{P_{\mathtt{SR}}}$. Then, we evaluate the *expected routing cost* $C_{\mathtt{SR}}(\rho, X)$. We report the average $\bar{C}_{\mathtt{SR}}$ of these values across

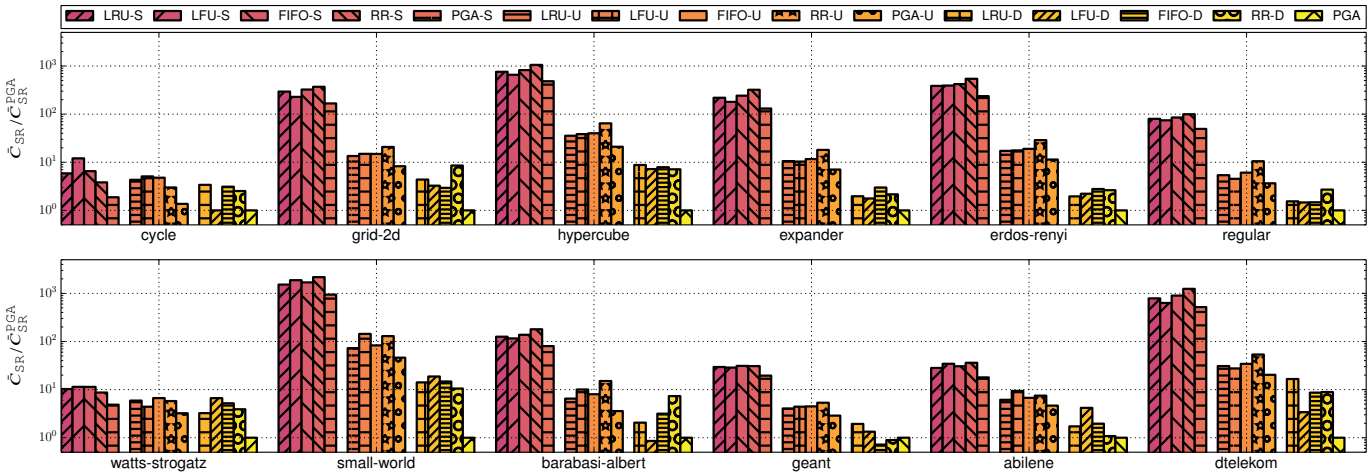


Fig. 5. Ratio of expected routing cost $\bar{C}_{\mathrm{SR}}$ to routing cost $\bar{C}_{\mathrm{SR}}^{\mathrm{PGA}}$ under our PGA policy, for different topologies and strategies. For each topology, each of the three groups of bars corresponds to a routing strategy, namely, RNS/shortest path routing (-S), uniform routing (-U), and dynamic routing (-D). The algorithm presented in [20] is PGA-S, while our algorithm (PGA), with ratio 1.0, is shown last for reference purposes; values of of $\bar{C}_{\mathrm{SR}}^{\mathrm{PGA}}$ are given in Table II.

TABLE II
GRAPH TOPOLOGIES, EXPERIMENT PARAMETERS, AND CONVERGENCE TIMES

Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert C\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert Q\rvert$	c_v	$\lvert\mathcal{P}_{(i,s)}\rvert$	$\bar{C}_{\mathrm{SR}}^{\mathrm{PGA}}$	LRU-S	PGA-S	LRU-U	PGA-U	LRU	PGA
cycle	30	60	10	100	10	2	2	20.17	0.47	865.29	0.47	436.14	6.62	148.20
grid-2d	100	360	300	1K	20	3	30	0.228	0.08	657.84	0.08	0.08	0.08	0.08
hypercube	128	896	300	1K	20	3	30	0.028	0.21	924.75	0.21	0.21	0.21	0.21
expander	100	716	300	1K	20	3	30	0.112	0.38	794.27	0.38	0.38	0.38	0.38
erdos-renyi	100	1042	300	1K	20	3	30	0.047	3.08	870.84	0.25	0.25	0.25	0.25
regular	100	300	300	1K	20	3	30	0.762	1.50	1183.97	0.05	8.52	0.05	11.49
watts-strogatz	100	400	300	1K	20	3	2	35.08	11.88	158.39	7.80	54.90	19.22	37.05
small-world	100	491	300	1K	20	3	30	0.029	0.30	955.48	0.30	0.30	0.30	0.30
barabasi-albert	100	768	300	1K	20	3	30	0.187	1.28	1126.24	1.28	6.86	1.28	7.58
geant	22	66	10	100	10	2	10	1.28	0.09	1312.96	1.85	12.71	0.09	14.41
abilene	9	26	10	90	9	2	10	0.911	3.44	802.66	3.44	23.08	5.75	14.36
dtelekom	68	546	300	1K	20	3	30	0.025	0.30	927.24	0.30	0.30	0.30	0.30

measurements collected after a warmup phase, during 1000 and 5000 time units of the simulation; that is, if t_i are the measurement times, then

$$\bar{C}_{\mathrm{SR}} = \frac{1}{t_{\mathrm{tot}} - t_{\mathrm{w}}} \sum_{t_i : \in [t_{\mathrm{w}}, t_{\mathrm{tot}}]} C_{\mathrm{SR}}(\rho(t_i), X(t_i)).$$

Performance w.r.t Routing Costs. The relative performance of the different strategies to our algorithm is shown in Figure 5. With the exception of cycle and watts-strogatz, where paths are scarce, we see several common trends across topologies. First, simply moving from RNS routing to uniform, multi-path routing, reduces the routing cost by a factor of 10. Even without optimizing routing or caching, simply increasing path options increases the available caching capacity. For all caching policies, optimizing routing through the dynamic routing policy (denoted by -D), reduces routing costs by another factor of 10. Finally, jointly optimizing routing and caching leads to a reduction by an additional factor between 2 and 10 times. In several cases, PGA outperforms RNS routing (including [20]) by 3 orders of magnitude.

Convergence. In Table II, we show the convergence time for different variants of LRU and PGA. We define the convergence time to be the time at which the time-average caching gain reaches 95% of the expected caching gain attained at steady state. LRU converges faster than PGA, though it converges to a sub-optimal stationary distribution. Interestingly, both -U and adaptive routing reduce convergence times for PGA, in some cases (like grid-2d and dtelekom) to the order of magnitude of LRU: this is because path diversification reduces contention: it assigns contents to non-overlapping caches, which are populated quickly with distinct contents.

IX. CONCLUSIONS

We have constructed joint caching and routing schemes with optimality guarantees for arbitrary network topologies. Identifying schemes that lead to improved approximation guarantees, especially on the routing cost directly rather than on the caching gain, is an important open question. Equally important is to incorporate queuing and congestion. In particular, accounting for queueing delays and identifying delay-minimizing strategies is open even under fixed routing. Such an analysis can also potentially be used to understand how different caching and routing schemes affect both delay optimality and throughput optimality. Finally, our adaptive algorithms proceed in a different timescale than content requests. Algorithms that mimic, e.g., path replication [12] would adapt

faster and reduce traffic. Providing such algorithms with guarantees is an open problem.

REFERENCES

[1] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," in *ACM ICN*, 2017.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*, 2009.

[3] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "VIP: A framework for joint dynamic forwarding and caching in named data networks," in *ICN*, 2014.

[4] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi, "Orchestrating massively distributed cdns," in *CoNEXT*, 2012.

[5] 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*, 2014.

[6] N. Laoutaris, S. Syntila, and I. Stavrakakis, "Meta algorithms for hierarchical web caches," in *ICPCC*, 2004.

[7] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.

[8] Y. Zhou, Z. Chen, and K. Li, "Second-level buffer cache management," *Parallel and Distributed Systems*, vol. 15, no. 6, pp. 505–519, 2004.

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

[10] K. Naveen, L. Massoulié, E. Baccelli, A. Carneiro Viana, and D. Towsley, "On the interaction between content caching and request assignment in cellular cache networks," in *ATC*, 2015.

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

[12] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *SIGCOMM*, 2002.

[13] S. Ioannidis and P. Marbach, "Absence of evidence as evidence of absence: A simple mechanism for scalable p2p search," in *INFOCOM*, 2009.

[14] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *ITC*, 2012.

[15] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *INFOCOM*, 2014.

[16] D. S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of TTL cache networks," *IFIP Performance*, 2014.

[17] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of TTL-based cache networks," in *VALUETOOLS*, 2012.

[18] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *INFOCOM*. IEEE, 2010, pp. 1–9.

[19] E. J. Rosensweig, D. S. Menasche, and J. Kurose, "On the steady-state of cache networks," in *INFOCOM*, 2013.

[20] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," in *Transactions on Networking*, 2018.

[21] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[22] J. F. Kurose and K. W. Ross, *Computer Networking: a Top-Down Approach*. Addison Wesley, 2007.

[23] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *SODA*, 2006.

[24] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *INFOCOM*, 2010.

[25] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[26] B. Nitzberg and V. Lo, "Distributed shared memory: A survey of issues and algorithms," *Computer*, vol. 24, no. 8, pp. 52–60, 1991.

[27] S. Albers, "Online algorithms: a survey," *Mathematical Programming*, vol. 97, no. 1-2, pp. 3–26, 2003.

[28] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. Tay, "A utility optimization approach to network cache design," in *INFOCOM*, 2015.

[29] N. Laoutaris, H. Che, and I. Stavrakakis, "The lcd interconnection of lru caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.

[30] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *ICN*. ACM.

[31] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2013, pp. 1–10.

[32] G. Rossini and D. Rossi, "Coupling caching and forwarding: Benefits, analysis, and implementation," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 127–136.

[33] 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.

[34] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive algorithms for distributed data management," *Journal of Computer and System Sciences*, vol. 51, no. 3, pp. 341–358, 1995.

[35] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *CoNext*, 2010.

[36] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.

[37] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: Hamlet-like doubts in ICN," in *ICN*, 2012.

[38] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 147–158.

[39] G. Carofiglio, L. Mekinda, and L. Muscariello, "Joint forwarding and caching with latency awareness in information-centric networking," *Computer Networks*, vol. 110, pp. 133–153, 2016.

[40] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 864–874, 2014.

[41] H. Xie, G. Shi, and P. Wang, "TECC: Towards collaborative in-network caching guided by traffic engineering," in *INFOCOM*, 2012.

[42] J. Edmonds, "Submodular functions, matroids, and certain polyhedra," *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, vol. 11, 1970.

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

[44] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *STOC*, 2008.

[45] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a submodular set function subject to a matroid constraint," in *Integer programming and combinatorial optimization*. Springer, 2007, pp. 182–196.

[46] ——, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[47] A. Krause and D. Golovin, "Submodular function maximization," *Tractability: Practical Approaches to Hard Problems*, vol. 3, no. 19, p. 8, 2012.

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

[49] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[50] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, no. 2, pp. 652–673, 1998.

[51] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000, iETF RFC2992.

[52] J. He and J. Rexford, "Toward internet-wide multipath routing," *IEEE network*, vol. 22, no. 2, 2008.

[53] S. Vutukury and J. J. Garcia-Luna-Aceves, "Mdva: A distance-vector multipath routing protocol," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2001, pp. 557–564.

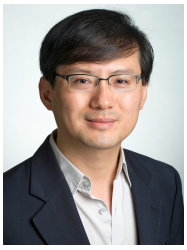
[54] M. K. Marina and S. R. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Network Protocols, 2001. Ninth International Conference on*. IEEE, 2001, pp. 14–23.

[55] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.

[56] B. Błaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *ICC*, 2015.

[57] M. X. Goemans and D. P. Williamson, "New 3/4-approximation algorithms for the maximum satisfiability problem," *SIAM Journal on Discrete Mathematics*, vol. 7, no. 4, pp. 656–666, 1994.
[58] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.
[59] H. J. Kushner and G. Yin, *Stochastic approximation and recursive algorithms and applications*. Springer Science & Business Media, 2003, vol. 35.
[60] C. Michelot, "A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$," *Journal of Optimization Theory and Applications*, vol. 50, no. 1, pp. 195–200, 1986.
[61] A. Nemirovski, *Efficient methods in convex programming*, 2005.

Stratis Ioannidis is an Assistant Professor in the Electrical and Computer Engineering department at Northeastern University, in Boston, MA, where he also holds a courtesy appointment with the College of Computer & Information Sciences. He received his B.Sc. (2002) in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and his M.Sc. (2004) and Ph.D. (2009) in Computer Science from the University of Toronto, Canada. Prior to joining Northeastern, he was a research scientist at the Technicolor research centers in Paris, France, and Palo Alto, CA, as well as at Yahoo Labs in Sunnyvale, CA. He is the recipient of an NSF CAREER award, a Google Faculty Research Award, and a Best Paper Award at the 2017 ACM Conference on Information-centric Networking (ICN).



Edmund Yeh is a Professor of Electrical and Computer Engineering at Northeastern University, Boston, USA. He received his B.S. in Electrical Engineering with Distinction and Phi Beta Kappa from Stanford University in 1994. He then studied at Cambridge University on the Winston Churchill Scholarship, obtaining his M.Phil in Engineering in 1995. He received his Ph.D. in Electrical Engineering and Computer Science from MIT under Professor Robert Gallager in 2001. He was previously Assistant and Associate Professor of Electrical Engineering, Computer Science, and Statistics at Yale University. He is the recipient of the Alexander von Humboldt Research Fellowship, the Army Research Office Young Investigator Award, and the Best Paper Award at the 2017 ACM Conference on Information-centric Networking (ICN) and at the 2015 IEEE International Conference on Communications (ICC) Communication Theory Symposium.