Optimal Cache Allocation under Network-Wide Capacity Constraint

Van Sy Mai, Stratis Ioannidis, Davide Pesavento, and Lotfi Benmohamed

***Abstract*— Network cache allocation and management are important aspects of an Information-Centric Network (ICN) design, such as one based on Named Data Networking (NDN). We address the problem of optimal cache size allocation and content placement in an ICN in order to maximize the caching gain resulting from routing cost savings. While prior art assumes a given cache size at each network node and focuses on content placement, we study the problem when a global, network-wide cache storage budget is given and we solve for the optimal per-node cache allocation. This problem arises in cloud-based network settings where each network node is virtualized and housed within a cloud data center node with associated dynamic storage resources acquired from the cloud node as needed. As the offline centralized version of the optimal cache allocation problem is NP-hard, we develop a distributed adaptive algorithm that provides an approximate solution within a constant factor from the optimal. Performance evaluation of the algorithm is carried out through extensive simulations over multiple network topologies, demonstrating that our proposal significantly outperforms existing cache allocation algorithms.**

I. Introduction

This paper involves modeling, analysis, and implementation of caching in cloud-based information centric networks subject to a limited network-wide cache budget. In these networks, a subset of nodes act as the designated sources for content (data producers) while any node can be a data consumer that generates requests for data items, which get forwarded toward the designated producers. These requests may not reach the ultimate producer as ICN forwarding ends when reaching a node along the path that has cached the requested item in its Content Store (CS). When such a cache hit takes place, the requested item is served from the CS and sent back to the requesting node along the reverse path.

Literature on ICN caching is extensive [1]–[9]. With an ICN being a network of caches where each network node is equipped with a content store, designing a good caching solution involves the aspects of determining the size of each CS, deciding which data objects should be cached (placement strategy), and which ones should be evicted when needed (replacement strategy). An efficient caching solution brings many benefits as it (a) reduces the data producer load since consumer's requests would rarely be satisfied by the producer but rather by caches, (b) significantly reduces the amount of network traffic and avoids bottlenecks caused by publishing data at a limited set of locations, and (c) offers users a faster content retrieval for an enhanced user experience. In other words, the investment in caching is expected to be of benefit to users, network operators, as well as content providers when it enables performance similar to content distribution networks (CDNs) by dynamically storing content in regions of high demand. While the problem of assigning items to caches under given fixed cache sizes has already been studied, cache capacity design subject to a global network-wide cache budget has not, and it is the focus of this paper.

Our goal is to achieve an optimal caching solution that maximizes the caching gain by minimizing the aggregate routing costs due to content retrieval across the network. The network load made up of each user demand, which is determined by the rate of requests and the paths they follow, is typically dynamic and not known in advance. As a result it is desirable to have adaptive caching solutions that can achieve optimal placement of data items in network caches without prior knowledge of the demands and be adaptive to any potential demand changes. In addition, caching needs to be distributed as well, since centralized solutions are not expected to be feasible when multiple administrative domains are involved. The network is expected to be more scalable when implementing distributed algorithms with caching decisions that rely only on locally available information.

Path replication, also known as Leave Copy Everywhere (LCE), is a popular caching strategy that is dynamic and distributed, and is often discussed in the literature [10]–[12]. When a data item is forwarded on the reverse path towards the consumer that requested it, it is cached at each intermediate node along the path. When a node's cache is full, a replacement takes place by evicting an already cached item. Despite its popularity, LCE has no performance guarantees and can be shown to be arbitrarily suboptimal [8].

Our contributions: Our main contributions are the following: (1) While previous work only deals with object placement in fixed size caches, we address a more general problem where no assumption of fixed cache sizes is made but rather uses a global network-wide cache budget constraint, and design the optimal per-node cache capacity; (2) We design an adaptive distributed algorithm for this problem by making use of a game theory framework in combination with a distributed gradient estimation approach; (3) We show that our game-based algorithm can provide suboptimal solutions within a factor $(1-1/e-\epsilon)$ of optimum for any given small $\epsilon > 0$ and without prior knowledge of the network demand; (4) We present results from extensive simulations over a number of network topologies that show how our algorithm outperforms those based on fixed size caches.

The remainder of this paper is organized as follows. In Section II, we briefly review related work. We introduce the

V. S. Mai, D. Pesavento and L. Benmohamed are with the Information Technology Laboratory, NIST, USA. Emails: {vansy.mai, davide.pesavento, lotfi.benmohamed}@nist.gov. Mention of commercial products does not imply NIST's endorsement. S. Ioannidis is with the Northeastern University and is supported by NSF grants NeTS-1718355 and CCF-1750539. Email: ioannidis@ece.neu.edu.

system model and formally state the problem in Section III. Our main results on a distributed algorithm are discussed in Section IV along with a discussion on implementation issues. Numerical results are presented in Section V. All the proofs are omitted and can be found in our technical report [13].

II. Related Work

The problem we study amounts to maximizing a submodular function subject to matroid constraints. Such problems are ubiquitous and appear in many domains (see [14] for a detailed overview). Though NP-hard, there exist known approximation algorithms: Nemhauser et al. [15] show that the greedy algorithm produces a solution within $1/2$ of the optimal. Vondrák [16] and Calinescu et al. [17] show that the so-called continuous-greedy algorithm produces a solution within $(1-1/e)$ of the optimal in polynomial time.

In the context of caching gain maximization, a restricted version of our problem is considered in [8], [18], where cache sizes are given and only object placements are optimized. Under a global constraint, however, the projected gradient ascent method in [8] becomes inadequate as a distributed adaptive algorithm. Our problem also resembles network resource allocation or utility maximization problems, where various decomposition techniques allows distributed implementations [19]. However, our (relaxed) global cost function is coupled in a such way that renders decomposition and decoupling approaches communication-expensive and complicated, let alone distributed adaptive implementation. This calls for a different approach. Specifically, we propose to employ the game theory-based framework in [20]–[22] for designing a distributed algorithm, where the global cost is embedded in the potential of a game. In contrast to [20]–[22], however, we do not assume separability of the potential function, nor do we employ any decoupling technique; this is achieved by making use of the distributed gradient estimation scheme in [8].

Finally, our work is also related to virtual machine (VM) allocation in cloud computing; see, e.g., [23], [24]. Heterogeneity of host resources and VM requirements lead to multiple knapsack-like constraints per host. With simpler storage constraints, we can provide distributed algorithms with provable approximation guarantees.

III. Preliminaries and Problem Formulation

A. Notation and System Model

Let $\mathbb{R}$ and $\mathbb{N}$ denote the sets of real and natural numbers, respectively. If $\mathcal{A}$ is a finite set, $|\mathcal{A}|$ denotes its cardinality. Let $[\cdot]_{\mathcal{X}}$ denote the projection onto $\mathcal{X}$; $[\cdot]_+ = \max(0,\cdot)$.

Consider a *connected* network $\mathcal{G}=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$ the set of links. Nodes are equipped with caches (CSes), whose capacity can be adjusted as part of an optimized design. As discussed above, the nodal cache size can be adjusted as needed by acquiring or relinquishing units of storage at the local cloud node (data center), part of the operator's deployed cloud. The local cache is used to store content items from a catalog made up of a set $\mathcal{C}$, and subsequently serve requests for these items. We denote by M the total cache capacity that the network operator is willing to deploy, it reflects a limit on the operator's budget.

Let $x_{v,i} \in \{0,1\}$ for $v \in \mathcal{V}$, $i \in \mathcal{C}$ be the variable indicating if node v stores item i. Node v's capacity is thus $\sum_{i\in\mathcal{C}} x_{vi}$, which must be less than its maximum cache capacity, denoted by $\bar{c}_v$. Moreover, the total $\sum_{v\in\mathcal{V},i\in\mathcal{C}} x_{vi}$ must be within given budget M. We assume that, for each item i, there exists a set $\mathcal{S}_i \subset \mathcal{V}$ that serve as designated servers (producers) for that item, i.e., $x_{vi}=1, \forall v\in\mathcal{S}_i$. Requests arrive in $\mathcal{G}$ and traverse predetermined paths towards producers. Formally, a request for item $i\in\mathcal{C}$ through path $p=\{p_1,\dots,p_K\}\subset\mathcal{V}$ is denoted by (i,p). Let $\mathcal{R}$ denote the set of all such requests. We assume that all $(i,p)\in\mathcal{R}$ are *well-routed*, i.e., p has no loops and terminates at producers in $\mathcal{S}_i$. Moreover, requests for each element in $\mathcal{R}$ arrive according to independent Poisson processes with rates $\lambda_{(i,p)}>0$, which is standard for modeling request arrivals (e.g., [1], [3]–[8]). A request (i,p) is routed along p until it reaches a cache that has item i. Then, a response message carrying item i is generated and sent over p in the reverse direction to the first node in p. We assume that the cost of routing an item over a link $(i,j)\in\mathcal{E}$ is $w_{ij}\in\mathbb{R}_+$, while the cost of forwarding requests/control messages is negligible.

B. Problem Statement

Let C_0 denote the expected cost when there are no items cached except for designated servers, i.e.,

$$C_0 = \textstyle\sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}. \tag{1}$$

When contents are cached according to an allocation $X=\{x_{vi}\}_{v\in\mathcal{V},i\in\mathcal{C}}$, the cost of serving a request $(i,p)\in\mathcal{R}$ is

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

yielding a caching gain $F := C_0 - \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} C_{(i,p)}$, i.e.,

$$F(X) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{1\le k\le |p|-1} w_{p_{k+1}p_k} \Big[1 - \prod_{1\le l\le k}(1-x_{p_l i})\Big] \tag{3}$$

We seek a distributed adaptive algorithm for the following problem:

Given cache budget M for the whole network, design an allocation X to maximize the caching gain:

$$\begin{aligned}
\text{(MaxCG)}\quad \max_{X}\quad & F(X) \\
\text{s.t.}\quad & x_{vi}\in\{0,1\}, \quad \forall v\in\mathcal{V}, i\in\mathcal{C} && (4)\\
& x_{vi}=1, \qquad \forall v\in\mathcal{S}_i, \forall i\in\mathcal{C} && (5)\\
& \textstyle\sum_{i\in\mathcal{C}} x_{vi} \le \bar{c}_v, \qquad \forall v\in\mathcal{V} && (6)\\
& \textstyle\sum_{v\in\mathcal{V},i\in\mathcal{C}} x_{vi} \le M && (7)
\end{aligned}$$

Let $\mathcal{D}_1$ denote the feasible set of (MaxCG), i.e., $\mathcal{D}_1 = \{X\in\mathbb{R}^{|\mathcal{V}|\times|\mathcal{C}|} \mid (4)-(7) \text{ hold}\}$. Since (MaxCG) is NP-hard [8], we seek approximate solutions. Moreover, we aim to develop *distributed* and *adaptive* algorithms that enable the caches themselves to adapt and update their capacities to solve the underlying optimization problem. For *centralized*, *offline* algorithms employing full knowledge of the demand and system parameters, see [13].

IV. Distributed Algorithm and Convergence

We develop an algorithm for (MaxCG) based on a number of approximation steps. We first apply a convex relaxation, followed by a smooth approximation. We then embed the resulting problem in a game theory framework and show that the gradient play strategy, combined with a distributed gradient estimation scheme, allows us to adapt both cache sizes and content allocations in a distributed fashion.

A. Convex Relaxation and Smooth Approximation

First, we employ the approach in [8], which convexifies both $\mathcal{D}_1$ and F. Specifically, we relax the Boolean variables:

$$\max \quad \{F(Y) \mid Y \in \mathcal{D}_2\} \tag{8}$$

with $\mathcal{D}_2 = \{X \in [0,1]^{|\mathcal{V}|\times|\mathcal{C}|} \mid (5)-(7) \text{ hold}\}$. Since F is nonconcave on $\mathcal{D}_2$, we approximate it with concave function

$$L(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{1\le k\le |p|-1} w_{p_{k+1}p_k} \min\left\{1, \textstyle\sum_{l=1}^{k} y_{p_l i}\right\} \tag{9}$$

satisfying $L(X) = f(X), \forall X \in \mathcal{D}_1$ and $(1-e^{-1})L(Y) \le F(Y) \le L(Y), \forall Y \in \mathcal{D}_2$. Thus, the resulting problem

$$L^* := \quad \max \quad \{L(Y) \mid Y \in \mathcal{D}_2\} \tag{10}$$

is convex[1] and approximates (MaxCG) within $(1-1/e)$ ratio.

Second, since L in (9) is not differentiable, we consider

$$\tilde{L}(Y) = \sum_{(i,p)\in\mathcal{R}} \lambda_{(i,p)} \sum_{1\le k\le |p|-1} w_{p_{k+1}p_k} \mathrm{sat}_\alpha\left(\textstyle\sum_{l=1}^{k} y_{p_l i}\right) \tag{11}$$

where $\alpha \in (0,1)$ is a small number and $\mathrm{sat}_\alpha(x) = 1$ if $x \ge 1 + \frac{\alpha}{2}$; $\mathrm{sat}_\alpha(x) = x$ if $x < 1 - \frac{\alpha}{2}$; and $\mathrm{sat}_\alpha(x) = 1 - (1 + \frac{\alpha}{2} - x)^2/2\alpha$ otherwise. Clearly, $\mathrm{sat}_\alpha(x)$ is a lower bound of $\min\{1, x\}, \forall x \in \mathbb{R}_+$. Thus, $\tilde{L}$ is a concave lower bound of L and $\lim_{\alpha\to 0^+} \tilde{L} = L$. Indeed, $\tilde{L}(Y) \le L(Y) \le \tilde{L}(Y) + \frac{\alpha}{8} C_0, \forall Y \in \mathcal{D}_2$. Thus, the following problem

$$\max \quad \{\tilde{L}(Y) \mid Y \in \mathcal{D}_2\} \tag{12}$$

can be used in place of (10) by using a sufficiently small α.

B. Potential Game-Based Design

First, we restate (12) as follows:

$$\max_{\{\mathbf{y}_v \in \Omega_v\}} \quad \{\tilde{L}(Y) \mid \textstyle\sum_{v\in\mathcal{V}} (\mathbf{y}_v^\mathsf{T}\mathbf{1} - c_v^0) \le 0\} \tag{13}$$

where $\mathbf{y}_v^\mathsf{T}$ is the v-th row of Y, $\mathbf{1}$ is a column vector of all ones, c_v^0 are constants such that $\sum_{v\in\mathcal{V}} c_v^0 = M$, and

$$\Omega_v = \{\mathbf{y} \in [0,1]^{|\mathcal{C}|} \mid \textstyle\sum_{i\in\mathcal{C}} y_i \le \bar{c}_v, y_i = 1 \text{ if } v \in \mathcal{S}_i\} \tag{14}$$

Suppose that each node v knows c_v^0 (e.g., $c_v^0 = M/|\mathcal{V}|$).

We now adapt the game theory framework in [20]–[22] for (13). In particular, we will design a state based potential game between the nodes so that they will converge to a pure Nash equilibrium that can be made arbitrarily close to an optimal solution of (13). The crucial differences between our design and that in [20]–[22] are the nodal cost functions and the implementation of the learning algorithm. In particular, we *do not assume that cost functions are separable across nodes* (indeed, the terms of the objective (13) are coupled).

[1] in fact, (10) can be converted into a linear program (e.g., [8]).

1) Game model: We now construct a game model for node caches; the evolution of which via appropriate dynamics eventually leads to a solution of (13) in a distributed fashion.

1. *State space:* Let $Z = (Y, \mathbf{e})$ denote the state of the game, where $\mathbf{e} = \{e_v\}_{v\in\mathcal{V}}$ and e_v is an error term of node v representing an estimation of $(\mathbf{y}_v^\mathsf{T}\mathbf{1} - c_v^0)$.
2. *Actions:* Each node v has a state-dependent action set $\mathcal{A}_v(Z)$, where an action $\mathbf{a}_v$ is a tuple $\mathbf{a}_v = (\hat{\mathbf{y}}_v, \{\hat{e}_{v\to u}\}_{u\in\mathcal{N}_v})$. Here, $\hat{e}_{v\to u}$ is the estimate error that node v sends to a direct neighbor u, and $\mathcal{N}_v$ denotes the set of node v's neighbors.
3. *State dynamics:* For any state $Z = (Y, \mathbf{e})$ and action $\{\mathbf{a}_v\}$, the next state $\tilde{Z} = (\tilde{Y}, \tilde{\mathbf{e}})$ is given by $\tilde{\mathbf{y}}_v = \mathbf{y}_v + \hat{\mathbf{y}}_v$ and $\tilde{e}_v = e_v + \hat{\mathbf{y}}_v^\mathsf{T}\mathbf{1} + \sum_{u\in\mathcal{N}_v} \left(\hat{e}_{u\to v} - \hat{e}_{v\to u}\right)$ where the admissible action set of node v is $\mathcal{A}_v(Z) = \mathcal{A}_v(\mathbf{y}_v) := \{\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{C}|} \mid \mathbf{y}_v + \hat{\mathbf{y}} \in \Omega_v\}$. Note that $\sum_{v\in\mathcal{V}} (\tilde{e}_v - \tilde{\mathbf{y}}_v^\mathsf{T}\mathbf{1}) = \sum_{v\in\mathcal{V}} (e_v - \mathbf{y}_v^\mathsf{T}\mathbf{1})$.
4. *Nodal cost function:* For a state Z and admissible action profile $\{\mathbf{a}_v \in \mathcal{A}_v(\mathbf{y}_v)\}_{v\in\mathcal{V}}$, let

$$J_v(Z, \mathbf{a}) = -\tilde{L}(\tilde{Y}) + \tfrac{\mu}{2} \textstyle\sum_{u\in\mathcal{N}_v} [\tilde{e}_u]_+^2, \tag{15}$$

be node v's cost function, where $(\tilde{Y}, \tilde{\mathbf{e}})$ is the next state and $\mu > 0$ is a parameter. Here, J_v involves the global (approximated) caching gain function $\tilde{L}$, but as we will show later, each node does not need to evaluate J_v.

The game model we just described admits a potential function $\Phi_\mu(Z, \mathbf{a}) = -\tilde{L}(\tilde{Y}) + \frac{\mu}{2} \sum_{v\in\mathcal{V}} [\tilde{e}_v]_+^2$ with bounded level sets. Thus, a Nash equilibrium always exists and can be reached by the *gradient play* strategy (see, e.g., [22]). The implementation of this strategy is given next.

2) Algorithm description: Suppose that time is partitioned into periods of equal length T. Each node updates its state $(\mathbf{y}_v, e_v)$ as follows (see [13] for the detailed derivation):

- *At period $t = 0$, each node v initializes $\mathbf{y}_v(0) \in \Omega_v$ and $e_v(0) \leftarrow (\mathbf{1}^\mathsf{T}\mathbf{y}_v(0) - c_v^0)$ such that $\sum_{v\in\mathcal{V}} e_v(0) \le 0$.*
- *At $t > 0$, node v exchanges $e_v(t)$ with neighbors, computes $\hat{e}_{v\to u}(t) = \gamma\mu\left([e_v(t)]_+ - [e_u(t)]_+\right)$ with step size $\gamma > 0$, and updates its state as follows:*

$$\mathbf{y}_v(t+1) = \left[\mathbf{y}_v(t) + \gamma\left(\nabla_{\mathbf{y}_v}\tilde{L}(Y(t)) - \mu\mathbf{1}[e_v(t)]_+\right)\right]_{\Omega_v} \tag{16}$$

$$e_v(t+1) = e_v(t) + \mathbf{1}^\mathsf{T}(\mathbf{y}_v(t+1) - \mathbf{y}_v(t)) + \textstyle\sum_{u\in\mathcal{N}_v} \hat{e}_{u\to v}(t) - \hat{e}_{v\to u}(t), \tag{17}$$

where $\nabla_{\mathbf{y}_v}\tilde{L}(Y(t))$ can be estimated in a distributed fashion as shown in Section IV-D.1 below.

In [20]–[22], a potential game-based algorithm is provided for solving a (more general) constrained optimization problem, the design of which, if applied to (12), would yield an exponentially large state space. Specifically, to decompose Φ_μ, each node v would need to maintain and update a local estimate Y_v of the state Y through exchanging information with its neighbors. This would incur much more expensive communication and computational costs compared to our algorithm outlined above. Our advantage is gained by employing a distributed scheme for each node to estimate partial gradients of $\tilde{L}$. Such an algorithm (given in Section IV-D.1 below) requires only a simple message exchange protocol.

C. Convergence

Note that any Nash equilibrium is optimal to $\Phi_\mu^* = \min_{\{\tilde{\mathbf{y}}_v \in \Omega_v\}} \Phi_\mu(Z, \mathbf{a})$. Therefore, we have the following.

Theorem 1: For a fixed μ, if $\{Z, \mathbf{a}\} = \{(Y, \mathbf{e}), (\hat{Y}, \hat{E})\}$ is a stationary state Nash equilibrium, Y is also optimal for

$$\max_{\{\mathbf{y}_v \in \Omega_v\}} \tilde{L}(Y) - \frac{\mu}{2|\mathcal{V}|}\big[\sum_{v\in\mathcal{V}}(\mathbf{y}_v^\mathsf{T}\mathbf{1} - c_v^0)\big]_+^2. \quad (18)$$

As a result, an approximate (with arbitrary given accuracy) solution to (13) can be obtained with sufficiently large μ.

Corollary 1: As $\mu \to \infty$, the equilibria of the game constitute solutions of (13)*.*

We now summarize approximation steps introduced so far in dealing with the original problem (MaxCG). First, we relax the binary constraints (4) and approximate the objective function F by L in (9), thereby obtaining (10), a convex problem on the relaxed feasible set. Second, since L is nondifferentiable, we then replace it with $\tilde{L}$ in (11). Third, by resorting to the potential game theory, we effectively remove the global constraint (7) by adding a penalizing term to $\tilde{L}$, resulting (18). In summary, (MaxCG) $\approx$ (10) $\approx$ (13) $\approx$ (18). Moreover, the overall approximation ratio (in terms of caching gains) is within $(1-\epsilon-1/e)$ for any given small $\epsilon > 0$ by selecting μ sufficiently large and α sufficiently small [13].

The convergence of our algorithm is given next.

Theorem 2: For Algorithm (16)–(17) *with $\gamma < \bar{\gamma}^0 := \frac{2}{(\alpha^{-1}C_0 + 2\mu)}$, we have $\lim_{t\to\infty} \Phi_\mu(Z(t), \mathbf{a}(t)) = \Phi_\mu^*$. Moreover, any limit point Y^* of $\{Y(t)\}$ is an optimizer of* (18)*.*

Note that $\bar{\gamma}^0$ is a theoretical bound for the gradient method, while step sizes larger than $\bar{\gamma}^0$ often still work in practice; the larger the step sizes are, the closer to instability.

Remark 1: Given a fixed μ, the global capacity constraint in (13) is likely to be violated due to the penalizing term in (18). To reduce such violation, we can initialize $\sum_{v\in\mathcal{V}} c_v^0 = (M - \epsilon)$ for some small $\epsilon \in (0, 1)$ and select $\mu = \Theta(\tilde{L}^*) = \Theta(C_0)$ (note that $L(Y) \le C_0, \forall Y$), where C_0 (or an upper bound $\bar{C}_0$) can be estimated in a centralized fashion from history data or in a distributed manner; see [13] for details.

D. Implementation considerations

This subsection details on how each node in the network can obtain online estimations of $\partial_{y_{vi}}\tilde{L}$ and update cache contents. See [13] for distributed estimation of step size bound $\bar{\gamma}_0$ in Theorem 2 for ensuring convergence.

1) Distributed gradient estimation: We adopt the mechanism used in [8], [9], namely, additional control messages are attached to the request and response traffic to gather needed information. This enables each node v to estimate partials $\partial_{\mathbf{y}_v}\tilde{L}$ in a distributed fashion by using information in the messages passing by during each time interval T. In particular:

- Every time a node generates a new request $(i, p) \in \mathcal{R}$, it creates an additional control message m_s attached to the request. At node p_1, $m_s(p_1) = y_{p_1 i}$. At node p_l,

$$m_s(p_l) = m_s(p_{l-1}) + y_{p_l i} \quad (19)$$

  until a node $u \in p$ with $m_s(u) > 1 + \frac{\alpha}{2}$ is found or the end of the path is reached (in which case $u = p_{|p|}$). Each visited node p_l keeps a local copy of $m_s(p_l)$.

TABLE I
GRAPH TOPOLOGIES AND PARAMETERS

Graph	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{Q}\rvert$	$\lvert\mathcal{R}\rvert$	M
grid_2d (G2)	100	180	100	20	1K	300
expander (Ex)	100	340	100	50	2K	400
barabasi_albert (BA)	100	384	100	50	2K	400
small_world (SW)	100	240	100	50	2K	400
watts_strogatz (WS)	100	200	100	50	2K	400
erdos_renyi (ER)	100	521	100	50	2K	400
geant (Ge)	22	33	100	20	1K	144
abilene (Ab)	9	13	10	9	100	28
dtelekom (Dt)	68	273	100	20	1K	304

- Node u then creates a control message m_r to send back in the reverse direction with $m_r(u) = 0$. At any p_l,

$$m_r(p_l) = m_r(p_{l+1}) + w_{p_{l+1}p_l}\mathrm{sat}'_\alpha(m_s(p_l)), \quad (20)$$

  where $\mathrm{sat}'_\alpha(x) = 0$ if $x \ge 1 + \frac{\alpha}{2}$; $\mathrm{sat}'_\alpha(x) = 1$ if $x < 1 - \frac{\alpha}{2}$; and $\mathrm{sat}'_\alpha(x) = (1 + \frac{\alpha}{2} - x)/\alpha$ otherwise.
- For each item i and each node v, let $t_{vi} := m_r(v)$ as computed above and let $\mathcal{T}_{vi}$ denote the set of t_{vi} collected by node v regarding item i during each time slot. It can be seen that $t_{vi} = \frac{\partial}{\partial y_{vi}} \sum_{k=k_p(v)}^{|p|-1} w_{p_{k+1}p_k}\mathrm{sat}_\alpha\big(\sum_{l=1}^{k} y_{p_l i}\big)$, where $k_p(v)$ denotes the position of v in path p. Then it can be shown [8, Lem. 1] that $z_{vi} := \sum_{t\in\mathcal{T}_{vi}} t/T$ is an unbiased estimate of $\partial_{y_{vi}}\tilde{L}$ and thus can be used in (16).

2) Eviction policy: At the end of each iteration t, each node v determines its expected cache size $\sum_{i\in\mathcal{C}} y_{vi}(t)$. As this can be fractional, a local rounding scheme is needed; e.g., randomized rounding as in [8]. A much simpler heuristic would be the following: node v finds a positive integer $c_v(t)$ such that 1) if (7) is a hard constraint, then $c_v(t) = \min\{\lfloor \mathbf{1}^\mathsf{T}\mathbf{y}_v(t)\rfloor, \bar{c}_v\}$; 2) otherwise, $c_v(t)$ is the nearest to $\sum_{i\in\mathcal{C}} y_{vi}(t)$. Finally, node v then places/keeps at most $c_v(t)$ items according to largest elements of $\mathbf{y}_v(t)$ in its cache.

3) Efficient update and message exchange: Our algorithm requires each node to perform only few basic operations at each iteration to update its states (16)–(17) and traversing control messages (19)–(20) for estimating local gradients; the projection $[\cdot]_{\Omega_v}$ in (16) can be as simple as scaling. Moreover, the number of exchange messages (including $\hat{e}_{v\to u}$, m_s, and m_r) is also small. They can be encoded in very few bytes and can be piggybacked onto existing traffic of Interest and Data packets, incurring negligible overhead and storage.

V. NUMERICAL EXAMPLES

We simulate our algorithm in Matlab over the graphs in Table I; see [13] for detailed descriptions of these graphs.

Experiment setup: For each graph, we generate a catalog $\mathcal{C}$ and assign each item $i \in \mathcal{C}$ to a node selected uniformly at random (u.a.r.) from $\mathcal{V}$. We select the weight of each edge u.a.r. from $[0.01, 1]$ and a set of consumers $\mathcal{Q} \subset \mathcal{V}$ u.a.r. Each consumer $v \in \mathcal{Q}$ requests an item i selected from $\mathcal{C}$ according to a Zipf distribution with parameter 1.2. The request is routed over the shortest path p between v and the designated server for item i. We choose $\bar{c}_v = |\mathcal{C}|$ and update period $T = 1$. We let $\alpha = 0.2$ $\mu = C_0/4$, $\gamma = \bar{\gamma}^0 = \frac{4}{11C_0}$, $c_v^0 = (M - \epsilon)/|\mathcal{V}|$ with $\epsilon = 0.1$ (see Remark 1).

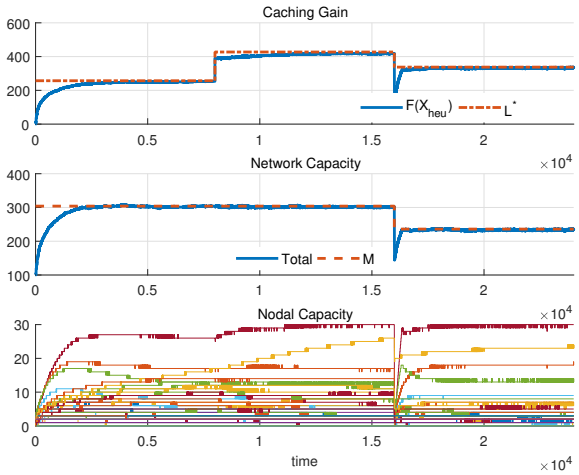

Fig. 1. (Color online) Simulation results for dtelekom network using cache allocation X_{heu} obtained from our heuristic placement in Sect. IV-D.2. L^* from (10) is obtained by a centralized algorithm and is an upper bound on the optimal caching gain. Bottom plot shows cache sizes $c_v(t), \forall v \in \mathcal{V}$.

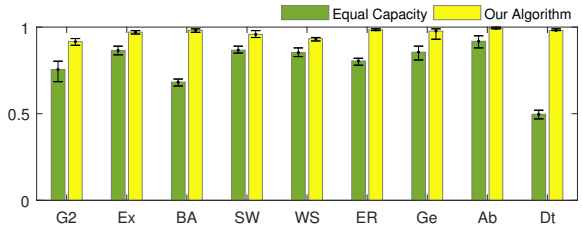

Fig. 2. Comparison of normalized caching gains for graphs in Table I.

Results: We simulate our algorithm on the dtelekom graph. During time interval $[0, 8000]$, $\lambda_{(i,p)}$ are selected u.a.r. from $(0.1, 1)$; after that $\lambda_{(i,p)} = 1, \forall (i,p) \in \mathcal{R}$. At $t = 16000$, we reduce the budget by $|\mathcal{V}|$ units. The simulation results are shown in Fig. 1, which clearly demonstrates optimality and adaptability of our algorithm. Specifically, $F(X_{heu})$ reaches near upper bound L^*, even under some network changes.

We also compare the performance, in terms of caching gains (normalized to L^*), of our algorithm with the centralized solution approach using the equal node-capacity allocation across all topologies in Table I. Specifically, the latter fixes $c_v(t) \equiv \bar{c}_v = \frac{M - |\mathcal{C}|}{|\mathcal{V}|} + |\{i : v \in \mathcal{S}_i\}|, \forall v \in \mathcal{V}$, i.e., (7) is redundant as $\sum_{v \in \mathcal{V}} \bar{c}_v = M$. Note that the optimal (relaxed) caching gain in equal node capacity, denoted by L^*_{EC} and obtained by solving (10) without global constraint (7), is not only an upper bound on caching gains of all suboptimal caching policies in the same setting, but also a lower bound of L^* in (10) with global constraint (7) and $\bar{c}_v = |\mathcal{C}|$. Significant gaps (ranging from 15% to 50%) between L^*_{EC} and other common caching strategies have been shown in [8] for a similar set of topologies. Here, we focus on showing improvement of $F(X_{heu})$ over L^*_{EC}. To this end, we run our algorithm for 10^4 time units with $\lambda_{(i,p)} = 1, \forall (i,p) \in \mathcal{R}$ and estimate the steady state caching gain by averaging the objective values $F(X_{heu})$ over the last 10^3 time units. Fig. 2 shows the average results of 10 runs, which clearly demonstrate that our algorithm yields (near) optimal caching gains and outperforms the best centralized solutions with equal capacity across all the topologies considered.

REFERENCES

[1] G. Carofiglio, L. Mekinda, and L. Muscariello, "LAC: Introducing latency-aware caching in information-centric networks," in *Proc. 40th Conf. Loc. Computer Netw.* IEEE, 2015, pp. 422–425.

[2] Y. Thomas, G. Xylomenos, C. Tsilopoulos, and G. C. Polyzos, "Object-oriented packet caching for ICN," in *Proc. 2nd ACM Conf. Info.-Centric Networking*. ACM, 2015, pp. 89–98.

[3] D. Nguyen, K. Sugiyama, and A. Tagami, "Congestion price for cache management in information-centric networking," in *Proc. IEEE Conf. Computer Commun. Wkshps*. IEEE, 2015, pp. 287–292.

[4] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *Proc. Int. Conf. Research Networking*. Springer, 2012, pp. 27–40.

[5] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. C. Tay, "A utility optimization approach to network cache design," in *Proc. 35th Annu. IEEE Int. Conf. Computer Commun.*, 2016, pp. 1–9.

[6] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networking," in *23rd Int. Conf. Computer Commun. Netw.*, 2014, pp. 1–8.

[7] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-aware caching and search in information-centric networks," in *Proc. 1st ACM Conf. Info.-Centric Network.*, 2014, pp. 37–46.

[8] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," in *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 44, no. 1, 2016, pp. 113–124.

[9] A. S. Gill, L. D'Acunto, K. Trichias, and R. van Brandenburg, "BidCache: Auction-based in-network caching in ICN," in *Globecom Wkshps.* IEEE, 2016, pp. 1–6.

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

[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerging Networking Experim. Tech.*, 2009, pp. 1–12.

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

[13] V. S. Mai, S. Ioannidis, D. Pesavento, and L. Benmohamed, "Optimal cache allocation for named data caching under network-wide capacity constraint," Tech. Rep., 2018. [Online]. Available: https://arxiv.org/pdf/1810.07229.pdf

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

[15] 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.

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

[17] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM J. Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

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

[19] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas. Commun.*, vol. 24, no. 8, pp. 1439–1451, 2006.

[20] N. Li and J. R. Marden, "Designing games for distributed optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 230–242, 2013.

[21] ——, "Decoupling coupled constraints through utility design," *IEEE Trans. Autom. Control*, vol. 59, no. 8, pp. 2289–2294, 2014.

[22] J. R. Marden and J. S. Shamma, "Game theory and distributed control," in *Handbook of game theory with economic applications*. Elsevier, 2015, vol. 4, pp. 861–899.

[23] W. Li, J. Tordsson, and E. Elmroth, "Virtual machine placement for predictable and time-constrained peak loads," in *Int. Wkshp Grid Econ. Business Models*. Springer, 2011, pp. 120–134.

[24] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. 2012 INFOCOM*. IEEE, 2012, pp. 2876–2880.