A TTL-based Approach for Content Placement in Edge Networks

Nitish K. Panigrahy[1(✉)], Jian Li[2], Faheem Zafari[3], Don Towsley[1], and Paul Yu[4]

[1] University of Massachusetts, Amherst, MA 01003, USA
{nitish,towsley}@cs.umass.edu
[2] Binghamton University, SUNY, Binghamton, NY 13902, USA
lij@binghamton.edu
[3] Imperial College London, London SW72BT, UK
faheem16@imperial.ac.uk
[4] U.S. Army Research Laboratory, Adelphi, MD 20783, USA
paul.l.yu.civ@mail.mil

Abstract. Edge networks are promising to provide better services to users by provisioning computing and storage resources at the edge of networks. However, due to the uncertainty and diversity of user interests, content popularity, distributed network structure, cache sizes, it is challenging to decide where to place the content, and how long it should be cached. In this paper, we study the utility optimization of content placement at edge networks through timer-based (TTL) policies. We propose provably optimal distributed algorithms that operate at each network cache to maximize the overall network utility. Our TTL-based optimization model provides theoretical answers to how long each content must be cached, and where it should be placed in the edge network. Extensive evaluations show that our algorithm outperforms path replication with conventional caching algorithms over some network topologies.

Keywords: TTL cache · Utility maximization · Edge network

1 Introduction

Content distribution has become a dominant application in today's Internet. Much of these contents are delivered by Content Distribution Networks (CDNs), which are provided by Akamai, Amazon, etc. [18]. There usually exists a stringent requirement on the latency between the service provider and end users for these applications. CDNs use a large network of caches to deliver content from a location close to the end users. This aligns with the trend of edge networks,

N. K. Panigrahy and J. Li—Authors with equal contribution.

Q. Zhao and L. Xia (Eds.): VALUETOOLS 2021, LNICST 404, pp. 1–21, 2021.
https://doi.org/10.1007/978-3-030-92511-6_1

where computing and storage resources are provisioned at the edge of networks. If a user's request is served by a nearby edge cache, the user experiences a faster response time than if it was served by the backend server. It also reduces bandwidth requirements at the central content repository.

With the aggressive increase in Internet traffic over past years [6], CDNs need to host content from thousands of regions belonging to web sites of thousands of content providers. Furthermore, each content provider may host a large variety of content, including videos, music, and images. Such an increasing diversity in content services requires CDNs to provide different quality of service to varying content classes and applications with different access characteristics and performance requirements. Significant economic benefits and important technical gains have been observed with the deployment of service differentiation [11]. While a rich literature has studied the design of fair and efficient caching algorithms for content distribution, little work has paid attention to the provision of multi-level services in edge networks.

Managing edge networks requires policies to route end-user requests to the local distributed caches, as well as caching algorithms to ensure availability of requested content at the cache. In general, there are two classes of policies for studying the performance of caching algorithms: *timer-based, i.e., Time-To-Live (TTL)* [5,10,13] and *non-timer-based* caching algorithms, e.g., Least-Recently-Used (LRU), Least-Frequently-Used (LFU), First In First Out (FIFO), and RANDOM [7]. Since the cache size is usually much smaller than the total amount of content, some contents need to be evicted if the requested content is not in the cache. Exact analysis of LRU, LFU, FIFO and RANDOM has proven to be difficult, even under the simple *Independence Reference Model* (IRM) [7], where requests are independent of each other. The strongly coupled nature of these eviction algorithms makes implementation of differential services challenging. In contrast, a TTL cache associates each content with a timer upon request and the content is evicted from the cache on timer expiry, independent of other contents. Analysis of these policies is simple since the eviction of contents are decoupled from each other.

Most studies have focused on the analysis of a single edge cache. When an edge network is considered, independence across different caches is usually assumed [25]. Again, it is hard to analyze most conventional caching algorithms, such as LRU, FIFO and RANDOM, but some accurate results for TTL caches are available [3,13].

In this paper, we consider a TTL-based edge network, where a set of network caches host a library of unique contents, and serve a set of users. Figure 1 illustrates an example of such a network, which is consistent with the YouTube video delivery system [23]. Each user can generate a request for a content, which is forwarded along a fixed *path* from the edge cache towards the server. Forwarding stops upon a cache hit, i.e., the requested content is found in a cache on the path. When such a cache hit occurs, the content is sent over the reverse path to the edge cache initializing the request. This raises the questions: *where to cache the requested content on the reverse path* and *what is the value of its timer?* Answer-

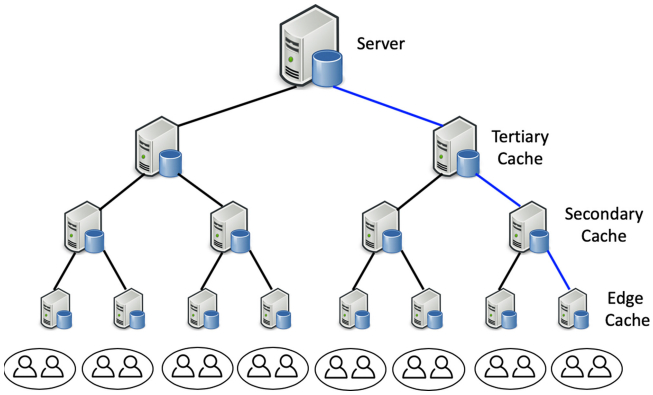

Fig. 1. An edge network with a server holding all contents and three layers of caches. Each edge cache serves a set of users with different requests. The blue line illustrates a unique path between the edge cache and the server. (Color figure online)

ing these questions can provide new insights in edge network design. However, it may also increase the complexity and hardness of the analysis.

Our goal is to provide thorough and rigorous answers to these questions. To that end, we consider moving the content one cache up (towards the user) if there is a cache hit on it and pushing the content one cache down (away from the user) once its timer expires in the cache hierarchy, since the recently evicted content may still be in demand. This policy is known as "Move Copy Down with Push" (MCDP) policy.

We first formulate a utility-driven caching framework for linear edge networks, where each content is associated with a utility and content is managed with a timer whose duration is set to maximize the aggregate utility for all contents over the edge network. Building on MCDP policy, we formulate the optimal TTL policy as a non-convex optimization problem in Sect. 3. One contribution of this paper is to show that this non-convex problem can be transformed into a convex one by change of variables. We further develop distributed algorithms for content management over linear edge networks.

Informed by our results for linear edge networks, we consider a general cache network where each edge cache serves distinct contents, i.e., there are no common contents among edge caches, in Sect. 4.2. We show that this edge network can be treated as a union of different linear edge networks between each edge cache and the central server.

We further consider a more general case where common content is requested among edge caches in Sect. 4.3. This introduces non-convex constraints, resulting in a non-convex utility maximization problem. We show that although the original problem is non-convex, the duality gap is zero. Based on this, we design an iterative primal-dual algorithm for content placement in edge network.

In Sect. 5, through numerical evaluations we show that our algorithm outperforms path replication with traditional caching algorithms over a broad array

of network topologies. We provide some discussions on extension of our utility maximization framework under MCDP to a general graph based cache networks in Sect. 5.4. Numerical results are given on how to optimize the performance. We discuss related works in Sect. 6 and conclude the paper in Sect. 7.

2 Preliminaries

We represent the edge cache network shown in Fig. 1 by a graph $G = (V, E)$. We use $\mathcal{D} = \{d_1, \cdots, d_n\}$ with $|\mathcal{D}| = n$ to denote the set of contents. Each network cache $v \in V$ can store up to B_v contents to serve requests from users. We assume that each user will first send a request for the content to its local network cache, which may then route the request to other caches for retrieving the content. Without loss of generality, we assume that there is a fixed and unique path from the local cache towards a terminal cache that is connected to a server that always contains the content.

To be more specific, a request (v, i, p) is determined by the local cache, v, that firstly received the user request, the requested content, i, and the path, p, over which the request is routed. We denote a path p of length $|p| = L$ as a sequence $\{v_{1p}, v_{2p}, \cdots, v_{Lp}\}$ of nodes $v_{lp} \in V$ such that $(v_{lp}, v_{(l+1)p}) \in E$ for $l \in \{1, \cdots, L\}$, where $v_{Lp} = v$. We assume that path p is loop-free and terminal cache v_{1p} is the only cache on path p that accesses the server for content i.

We assume that the request processes for distinct contents are described by independent Poisson processes with arrival rate λ_i for content $i \in \mathcal{D}$. Denote $\Lambda = \sum_{i=1}^{n} \lambda_i$. Then the popularity (request probability) of content i satisfies [2]

$$\rho_i = \frac{\lambda_i}{\Lambda}, \quad i = 1, \cdots, n. \tag{1}$$

We consider TTL caches in this paper. Each content i is associated with a timer T_{ij} at cache j. Suppose content i is requested and routed along path p. There are two cases: (i) content i is not in any cache along path p, in which case content i is fetched from the server and inserted into the terminal cache (denoted by cache 1)[1]. Its timer is set to T_{i1}; (ii) if content i is in cache l along path p, content i is moved to cache $l+1$ preceding cache l in which i is found, and the timer at cache $l+1$ is set to $T_{i(l+1)}$. Content i is pushed one cache back to cache $l-1$ and the timer is set to $T_{i(l-1)}$, once the timer expires. We call it Move Copy Down with Push (MCDP) [24]. Denote the *hit probability* of content i as h_i, then the corresponding *hit rate* is $\lambda_i h_i$.

Denote by $\mathcal{P}$ the set of all requests, and $\mathcal{P}^i$ the set of requests for content i. Suppose a network cache v serves two requests (v_1, i_1, p_1) and (v_2, i_2, p_2), then there are two cases: (i) non-common requested content, i.e., $i_1 \neq i_2$; and (ii) common requested content, i.e., $i_1 = i_2$. In the following, we will focus on how to design optimal TTL policies for content placement in an edge cache network under these two cases.

[1] Since we consider path p, for simplicity, we move the dependency on p and v, denote it as nodes $1, \cdots, L$ directly.

While classical cache eviction policies such as LRU provide good performance and are easy to implement, Garetto et al. [14] showed that K-LRU[2] can significant improve LRU even for very small K. Furthermore, Ramadan et al. [23] proposed K-LRU with big cache abstraction (K-LRU(B)) to effectively utilize resources in a hierarchical network of cache servers. Thus, in the rest of the paper, we compare the performance of MCDP to K-LRU and K-LRU(B).

3 Linear Edge Network

We begin with a linear edge network, i.e., there is a single path between the user and the server, composed of $|p| = L$ caches labeled $1, \cdots, L$. A content enters the edge network via cache 1, and is promoted to a higher index cache whenever a cache hit occurs. In the following, we consider the MCDP replication strategy when each cache operates with a TTL policy.

3.1 Stationary Behavior

Requests for content i arrive according to a Poisson process with rate λ_i. Under TTL, content i spends a deterministic time in a cache if it is not requested, independent of all other contents. We denote the timer as T_{il} for content i in cache l on the path p, where $l \in \{1, \cdots, |p|\}$.

Denote by t_k^i the k-th time that content i is either requested or the timer expires. For simplicity, we assume that content is in cache 0 (i.e., server) when it is not in the cache network. We then define a discrete time Markov chain (DTMC) $\{X_k^i\}_{k\geq 0}$ with $|p|+1$ states, where X_k^i is the index of the cache that content i is in at time t_k^i. The event that the time between two requests for content i exceeds T_{il} occurs with probability $e^{-\lambda_i T_{il}}$; consequently we obtain the transition probability matrix of $\{X_k^i\}_{k\geq 0}$ and compute the stationary distribution. The timer-average probability that content i is in cache $l \in \{1, \cdots, |p|\}$ is

$$h_{i1} = \frac{e^{\lambda_i T_{i1}} - 1}{1 + \sum_{j=1}^{|p|} (e^{\lambda_i T_{i1}} - 1) \cdots (e^{\lambda_i T_{ij}} - 1)}, \tag{2a}$$

$$h_{il} = h_{i(l-1)}(e^{\lambda_i T_{il}} - 1),\ l = 2, \cdots, |p|, \tag{2b}$$

where h_{il} is also the hit probability for content i at cache l.

Remark 1. The stationary analysis of MCDP is similar to a different caching policy LRU($\boldsymbol{m}$) considered in [15]. We relegate its explicit expression to Appendix A.1, and also refer interested readers to [15] for more detail.

[2] K-LRU adds $K-1$ meta-caches ahead of the real cache. Only "popular" contents (requested at least $K-1$ times) are stored in real cache.

3.2 From Timer to Hit Probability

We consider a TTL cache network where requests for different contents are independent of each other and each content i is associated with a timer T_{il} at each cache $l \in \{1, \cdots, |p|\}$ on the path. Denote $\boldsymbol{T}_i = (T_{i1}, \cdots, T_{i|p|})$ and $\boldsymbol{T} = (\boldsymbol{T}_1, \cdots, \boldsymbol{T}_n)$. From (2), the overall utility on path p is given as

$$\sum_{i \in \mathcal{D}} \sum_{l=1}^{|p|} \psi^{|p|-l} U_i(\lambda_i h_{il}(\boldsymbol{T})), \tag{3}$$

where the utility function $U_i : [0, \infty) \to \mathbb{R}$ is assumed to be increasing, continuously differentiable, and strictly concave of content hit rate, and $0 < \psi \leq 1$ is a discount factor capturing the utility degradation along the request's routing direction. Since each cache is finite in size, we have the capacity constraint

$$\sum_{i \in \mathcal{D}} h_{il}(\boldsymbol{T}) \leq B_l, \quad l \in \{1, \cdots, |p|\}. \tag{4}$$

Therefore, the optimal TTL policy for content placement on path p is the solution of the following optimization problem

$$\begin{aligned} \max_{\boldsymbol{T}} \quad & \sum_{i \in \mathcal{D}} \sum_{l=1}^{|p|} \psi^{|p|-l} U_i(\lambda_i h_{il}(\boldsymbol{T})) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{D}} h_{il}(\boldsymbol{T}) \leq B_l, \quad l \in \{1, \cdots, |p|\}, \\ & T_{il} \geq 0, \quad \forall i \in \mathcal{D}, \quad l = 1, \cdots, |p|, \end{aligned} \tag{5}$$

where $h_{il}(\boldsymbol{T})$ is given in (2). However, (5) is a non-convex optimization with a non-linear constraint. Our objective is to characterize the optimal timers for different contents on path p.. To that end, it is helpful to express (5) in terms of hit probabilities. In the following, we discuss how to change the variables from timer to hit probability.

Since $0 \leq T_{il} \leq \infty$, it is easy to check that $0 \leq h_{il} \leq 1$ for $l \in \{1, \cdots, |p|\}$ from (2a) and (2b). Furthermore, it is clear that there exists a mapping between $(h_{i1}, \cdots, h_{i|p|})$ and $(T_{i1}, \cdots, T_{i|p|})$. By simple algebra, we obtain

$$T_{i1} = \frac{1}{\lambda_i} \log \left(1 + \frac{h_{i1}}{1 - \left(h_{i1} + h_{i2} + \cdots + h_{i|p|} \right)} \right), \tag{6a}$$

$$T_{il} = \frac{1}{\lambda_i} \log \left(1 + \frac{h_{il}}{h_{i(l-1)}} \right), \quad l = 2, \cdots, |p|. \tag{6b}$$

Note that

$$h_{i1} + h_{i2} + \ldots + h_{i|p|} \leq 1, \tag{7}$$

must hold during the operation, which is always true for our caching policies.

3.3 Maximizing Aggregate Utility

With the change of variables discussed above, we can reformulate (5) as follows

$$\max \quad \sum_{i\in\mathcal{D}}\sum_{l=1}^{|p|}\psi^{|p|-l}U_i(\lambda_i h_{il}) \tag{8a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{D}} h_{il} \le B_l, \quad l=1,\cdots,|p|, \tag{8b}$$

$$\sum_{l=1}^{|p|} h_{il} \le 1, \quad \forall i \in \mathcal{D}, \tag{8c}$$

$$0 \le h_{il} \le 1, \quad \forall i \in \mathcal{D}, \quad l=1,\cdots,|p| \tag{8d}$$

where (8b) is the cache capacity constraint and (8c) is due to the variable exchanges under MCDP as discussed above. Problem (8) under MCDP has a unique global optimum.

3.4 Upper Bound (UB) on optimal Aggregate Utility

Constraint (8c) in (8) is enforced due to variable exchanges under MCDP as discussed above. Here we can define an upper bound on optimal aggregate utility by removing (8c) and solving the following optimization problem

$$\max \sum_{i\in\mathcal{D}}\sum_{l=1}^{|p|}\psi^{|p|-l}U_i(\lambda_i h_{il}), \quad \text{s.t., constraints (8b) and (8d).} \tag{9}$$

Note that the UB optimization problem is now independent of any timer driven caching policy and can be used as a performance benchmark for comparing various caching policies. Furthermore, it is easier to solve UB optimization problem (9) since it involves a smaller number of constraints compared to MCDP based optimization problem (8).

3.5 Distributed Algorithm

In Sect. 3.3, we formulated convex utility maximization problems with a fixed cache size. However, system parameters can change over time, so it is not feasible to solve the optimization offline and implement the optimal strategy. Thus, we need to design distributed algorithms to implement the optimal strategy and adapt to the changes in the presence of limited information.

Primal Algorithm: We aim to design an algorithm based on the optimization problem in (8), which is the primal formulation. Denote $\boldsymbol{h}_i = (h_{i1},\cdots,h_{i|p|})$ and

$\boldsymbol{h} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_n)$. We first define the following objective function.

$$Z(\boldsymbol{h}) = \sum_{i\in\mathcal{D}}\sum_{l=1}^{|p|} \psi^{|p|-l} U_i(\lambda_i h_{il}) - \sum_{l=1}^{|p|} C_l\left(\sum_{i\in\mathcal{D}} h_{il} - B_l\right) - \sum_{i\in\mathcal{D}} \tilde{C}_i\left(\sum_{l=1}^{|p|} h_{il} - 1\right) - \sum_{i\in\mathcal{D}}\sum_{l=1}^{|p|} \hat{C}_{il}(-h_{il}), \tag{10}$$

where $C_l(\cdot), \tilde{C}_i(\cdot)$ and $\hat{C}_{il}(\cdot)$ are convex and non-decreasing penalty functions denoting the cost for violating constraints (8b) and (8c).

Note that constraint (8c) ensures $h_{il} \leq 1\ \forall i \in \mathcal{D},\ l = 1, \cdots, |p|$, provided $h_{il} \geq 0$. One can assume that $h_{il} \geq 0$ holds in writing down (10). This would be true, for example, if the utility function is a β-fair utility function with $\beta > 0$ (Sect. 2.5 [26]). For other utility functions, it is challenging to incorporate constraint (8d) since it introduces $n|p|$ additional price functions. For all cases evaluated across various system parameters we found $h_{il} \geq 0$ to hold true. Hence we ignore constraint (8d) in the primal formulation and define the following objective function

$$Z(\boldsymbol{h}) = \sum_{i\in\mathcal{D}}\sum_{l=1}^{|p|} \psi^{|p|-l} U_i(\lambda_i h_{il}) - \sum_{l=1}^{|p|} C_l\left(\sum_{i\in\mathcal{D}} h_{il} - B_l\right) - \sum_{i\in\mathcal{D}} \tilde{C}_i\left(\sum_{l=1}^{|p|} h_{il} - 1\right). \tag{11}$$

It is clear that $Z(\cdot)$ is strictly concave. Hence, a natural way to obtain the maximal value of (11) is to use the standard *gradient ascent algorithm* to move the variable h_{il} for $i \in \mathcal{D}$ and $l \in \{1, \cdots, |p|\}$ in the direction of gradient,

$$\frac{\partial Z(\boldsymbol{h})}{\partial h_{il}} = \lambda_i \psi^{|p|-l} U_i'(\lambda_i h_{il}) - C_l'\left(\sum_{j\in\mathcal{D}} h_{jl} - B_l\right) - \tilde{C}_i'\left(\sum_{m=1}^{|p|} h_{im} - 1\right), \tag{12}$$

where $U_i'(\cdot)$, $C_l'(\cdot)$, $\tilde{C}_i'(\cdot)$ denote partial derivatives w.r.t. h_{il}.

Since h_{il} indicates the probability that content i is in cache l, $\sum_{j\in\mathcal{D}} h_{jl}$ is the expected number of contents currently in cache l, denoted by $B_{\text{curr},l}$.

Therefore, the primal algorithm for MCDP is given by

$$T_{il}[k] \leftarrow \begin{cases} \frac{1}{\lambda_i}\log\left(1 + \frac{h_{il}[k]}{1-\left(h_{i1}[k]+h_{i2}[k]+\cdots+h_{i|p|}[k]\right)}\right), & l = 1; \\ \frac{1}{\lambda_i}\log\left(1 + \frac{h_{il}[k]}{h_{i(l-1)}[k]}\right), & l = 2, \cdots, |p|, \end{cases} \tag{13a}$$

$$h_{il}[k+1] \leftarrow \max\left\{0, h_{il}[k] + \zeta_{il}\left[\lambda_i \psi^{|p|-l} U_i'(\lambda_i h_{il}[k]) - C_l'\left(B_{\text{curr},l} - B_l\right) - \tilde{C}_i'\left(\sum_{m=1}^{|p|} h_{im}[k] - 1\right)\right]\right\}, \tag{13b}$$

where $\zeta_{il} > 0$ is the step-size parameter, and k is the iteration number incremented upon each request arrival.

Remark 2. Note that the primal formulation in (13) can be implemented distributively with respect to (w.r.t.) different contents and caches by some amount of book-keeping. For example in (13b), $\sum_{m=1}^{|p|} h_{im}[k]$ at cache l can be computed by first storing the value of $\sum_{m=1}^{|p|} h_{im}[k]$ at the edge cache in previous iteration and updating it during delivery of content i (from cache l) to the user.

4 General Edge Networks

In Sect. 3, we consider linear edge networks and characterize the optimal TTL policy for content when coupled with MCDP. Inspired by these results, we consider general edge networks in this section.

4.1 Contents, Servers and Requests

We consider the general edge network described in Sect. 2. Denote by $\mathcal{P}$ the set of all requests, and $\mathcal{P}^i$ the set of requests for content i. Suppose a cache in node v serves two requests (v_1, i_1, p_1) and (v_2, i_2, p_2), then there are two cases: (i) non-common requested content, i.e., $i_1 \neq i_2$; and (ii) common requested content, i.e., $i_1 = i_2$.

4.2 Non-common Requested Content

In this section, we consider the case that each network cache serves requests for different contents from each request (v, i, p) passing through it. Since there is no coupling between different requests (v, i, p), we can directly generalize the results for a particular path p in Sect. 3 to a tree network. Hence, given the utility maximization formulation in (8), we can directly formulate the optimization problem for MCDP as

$$\max \quad \sum_{i\in\mathcal{D}} \sum_{p\in\mathcal{P}^i} \sum_{l=1}^{|p|} \psi^{|p|-l} U_{ip}(\lambda_{ip} h_{il}^{(p)}) \tag{14a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{D}} \sum_{p:l\in\{1,\cdots,|p|\}} h_{il}^{(p)} \leq B_l, \quad \forall l \in V, \tag{14b}$$

$$\sum_{l=1}^{|p|} h_{il}^{(p)} \leq 1, \quad \forall i \in \mathcal{D}, p \in \mathcal{P}^i, \tag{14c}$$

$$0 \leq h_{il}^{(p)} \leq 1, \quad \forall i \in \mathcal{D}, l \in \{1, \cdots, |p|\}, p \in \mathcal{P}^i. \tag{14d}$$

The optimization problem defined in (14) under MCDP has a unique global optimum.

4.3 Common Requested Contents

Now consider the case where different users share the same content, e.g., there are two requests (v_1, i, p_1) and (v_2, i, p_2). Suppose that cache l is on both paths p_1 and p_2, where v_1 and v_2 request the same content i. If we cache separate copies on each path, results from the previous section apply. However, maintaining redundant copies in the same cache decreases efficiency. A simple way to deal with that is to only cache one copy of content i at l to serve both requests from v_1 and v_2. Though this reduces redundancy, it complicates the optimization problem.

In the following, we formulate a utility maximization problem for MCDP with TTL caches, where all users share the same requested contents $\mathcal{D}$.

$$\max \quad \sum_{i\in\mathcal{D}} \sum_{p\in\mathcal{P}^i} \sum_{l=1}^{|p|} \psi^{|p|-l} U_{ip}(\lambda_{ip} h_{il}^{(p)}) \tag{15a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{D}} \left(1 - \prod_{p:j\in\{1,\cdots,|p|\}} (1 - h_{ij}^{(p)})\right) \leq B_j, \quad \forall j \in V, \tag{15b}$$

$$\sum_{j\in\{1,\cdots,|p|\}} h_{ij}^{(p)} \leq 1, \quad \forall i \in \mathcal{D}, p \in \mathcal{P}^i, \tag{15c}$$

$$0 \leq h_{il}^{(p)} \leq 1, \quad \forall i \in \mathcal{D}, j \in \{1, \cdots, |p|\}, p \in \mathcal{P}^i, \tag{15d}$$

where (15b) ensures that only one copy of content $i \in \mathcal{D}$ is cached at node j for all paths p that pass through node j. This is because the term $1 - \prod_{p:j\in\{1,\cdots,|p|\}}(1 - h_{ij}^{(p)})$ is the overall hit probability of content i at node j over all paths. (15c) is the cache capacity constraint and (15d) is the constraint from MCDP TTL cache policy as discussed in Sect. 3.2. (15) under MCDP is a non-convex optimization problem.

Example 1. Consider two requests (v_1, i, p_1) and (v_2, i, p_2) with paths p_1 and p_2 intersecting at j. Let the corresponding path perspective hit probability be $h_{ij}^{(p_1)}$ and $h_{ij}^{(p_2)}$. Then the term inside outer summation of (15b) is $1 - (1 - h_{ij}^{(p_1)})(1 - h_{ij}^{(p_2)})$, i.e., the hit probability of content i in j.

Remark 3. Note that we assume independence between different requests (v, i, p) in (15), e.g., in Example 1, if the insertion of content i in node j is caused by request (v_1, i, p_1), when request (v_2, i, p_2) comes, it is not counted as a cache hit from its perspective. Our framework still holds if we follow the logical TTL MCDP on a path. However, in that case, the utilities will be larger than the one we consider here.

In the following, we develop an optimization framework that handles the non-convexity issue in this optimization problem and provides a distributed solution.

To this end, we first introduce the Lagrangian function

$$L(\boldsymbol{h},\boldsymbol{\nu},\boldsymbol{\mu}) = \sum_{i\in\mathcal{D}}\sum_{p\in\mathcal{P}^i}\sum_{l=1}^{|p|}\psi^{|p|-l}U_{ip}(\lambda_{ip}h_{il}^{(p)}) - \sum_{i\in\mathcal{D}}\sum_{p\in\mathcal{P}^i}\mu_{ip}\left(\sum_{j\in\{1,\cdots,|p|\}}h_{ij}^{(p)} - 1\right)$$
$$-\sum_{j\in V}\nu_j\left(\sum_{i\in\mathcal{D}}\left[1-\prod_{p:j\in\{1,\cdots,|p|\}}(1-h_{ij}^{(p)})\right] - B_j\right), \tag{16}$$

where the Lagrangian multipliers (price vector and price matrix) are $\boldsymbol{\nu} = (\nu_j)_{j\in V}$, and $\boldsymbol{\mu} = (\mu_{ip})_{i\in\mathcal{D},p\in\mathcal{P}}$. Constraint (15d) is ignored in the Lagrangian function due to the same reason stated for the primal formulation in Sect. 3.5.

The dual function can be defined as

$$d(\boldsymbol{\nu},\boldsymbol{\mu}) = \sup_{\boldsymbol{h}} L(\boldsymbol{h},\boldsymbol{\nu},\boldsymbol{\mu}), \tag{17}$$

and the dual problem is given as

$$\min_{\boldsymbol{\nu},\boldsymbol{\mu}} \quad d(\boldsymbol{\nu},\boldsymbol{\mu}) = L(\boldsymbol{h}^*(\boldsymbol{\nu},\boldsymbol{\mu}),\boldsymbol{\nu},\boldsymbol{\mu}), \quad \text{s.t.} \quad \boldsymbol{\nu},\boldsymbol{\mu} \geq \mathbf{0}, \tag{18}$$

where the constraint is defined pointwise for $\boldsymbol{\nu},\boldsymbol{\mu}$, and $\boldsymbol{h}^*(\boldsymbol{\nu},\boldsymbol{\mu})$ is a function that maximizes the Lagrangian function for given $(\boldsymbol{\nu},\boldsymbol{\mu})$, i.e.,

$$\boldsymbol{h}^*(\boldsymbol{\nu},\boldsymbol{\mu}) = \arg\max_{\boldsymbol{h}} L(\boldsymbol{h},\boldsymbol{\nu},\boldsymbol{\mu}). \tag{19}$$

The dual function $d(\boldsymbol{\nu},\boldsymbol{\mu})$ is always convex in $(\boldsymbol{\nu},\boldsymbol{\mu})$ regardless of the convexity of the optimization problem (15) [4]. Therefore, it is always possible to iteratively solve the dual problem using

$$\nu_l[k+1] = \nu_l[k] - \gamma_l\frac{\partial L(\boldsymbol{\nu},\boldsymbol{\mu})}{\partial\nu_l}, \quad \mu_{ip}[k+1] = \mu_{ip}[k] - \eta_{ip}\frac{\partial L(\boldsymbol{\nu},\boldsymbol{\mu})}{\partial\mu_{ip}}, \tag{20}$$

where γ_l and η_{ip} are the step sizes, and $\frac{\partial L(\boldsymbol{\nu},\boldsymbol{\mu})}{\partial\nu_l}$ and $\frac{\partial L(\boldsymbol{\nu},\boldsymbol{\mu})}{\partial\mu_{ip}}$ are the partial derivative of $L(\boldsymbol{\nu},\boldsymbol{\mu})$ w.r.t. ν_l and μ_{ip}, respectively, satisfying

$$\frac{\partial L(\boldsymbol{\nu},\boldsymbol{\mu})}{\partial\nu_l} = -\left(\sum_{i\in\mathcal{D}}\left[1-\prod_{p:l\in\{1,\cdots,|p|\}}(1-h_{il}^{(p)})\right] - B_l\right),$$
$$\frac{\partial L(\boldsymbol{\nu},\boldsymbol{\mu})}{\partial\mu_{ip}} = -\left(\sum_{j\in\{1,\cdots,|p|\}}h_{ij}^{(p)} - 1\right). \tag{21}$$

Sufficient and necessary conditions for the uniqueness of $\boldsymbol{\nu},\boldsymbol{\mu}$ are given in [19]. The convergence of primal-dual algorithm consisting of (19) and (20) is guaranteed if the original optimization problem is convex. However, our problem is not convex. Nevertheless, we next show that the duality gap is zero, hence (19) and (20) converge to the globally optimal solution. To begin with, we introduce the following results

Theorem 1 *[27] (Sufficient Condition). If the price based function $\boldsymbol{h}^*(\boldsymbol{\nu}, \boldsymbol{\mu})$ is continuous at one or more of the optimal lagrange multiplier vectors $\boldsymbol{\nu}^*$ and $\boldsymbol{\mu}^*$, then the iterative algorithm consisting of (19) and (20) converges to the globally optimal solution.*

Theorem 2 *[27]. If at least one constraint of (15) is active at the optimal solution, the condition in Theorem 1 is also a necessary condition.*

Hence, if we can show the continuity of $\boldsymbol{h}^*(\boldsymbol{\nu}, \boldsymbol{\mu})$ and that constraints (15) are active, then given Theorems 1 and 2, the duality gap is zero, i.e., (19) and (20) converge to the globally optimal solution.

Take the derivative of $L(\boldsymbol{h}, \boldsymbol{\nu}, \boldsymbol{\mu})$ w.r.t. $h_{il}^{(p)}$ for $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$, we have

$$\frac{\partial L(\boldsymbol{h}, \boldsymbol{\nu}, \boldsymbol{\mu})}{\partial h_{il}^{(p)}} = \psi^{|p|-l} \lambda_{ip} U'_{ip}(\lambda_{ip} h_{il}^{(p)}) - \mu_{ip} - \nu_l \left(\prod_{\substack{q:q \neq p, \\ j \in \{1, \cdots, |q|\}}} (1 - h_{ij}^{(q)}) \right). \quad (22)$$

Setting (22) equal to zero, we obtain

$$U'_{ip}(\lambda_{ip} h_{il}^{(p)}) = \frac{1}{\psi^{|p|-l} \lambda_{ip}} \left(\nu_l \left(\prod_{\substack{q:q \neq p, \\ j \in \{1, \cdots, |q|\}}} (1 - h_{ij}^{(q)}) \right) + \mu_{ip} \right). \quad (23)$$

Consider the utility function $U_{ip}(\lambda_{ip} h_{il}^{(p)}) = w_{ip} \log(1 + \lambda_{ip} h_{il}^{(p)})$, then $U'_{ip}(\lambda_{ip} h_{il}^{(p)}) = w_{ip}/(1 + \lambda_{ip} h_{il}^{(p)})$. Hence, from (23), we have

$$h_{il}^{(p)} = \frac{w_{ip} \psi^{|p|-l}}{\nu_l \left(\prod_{\substack{q:q \neq p, \\ j \in \{1, \cdots, |q|\}}} (1 - h_{ij}^{(q)}) \right) + \mu_{ip}} - \frac{1}{\lambda_{ip}}. \quad (24)$$

Lemma 1. *Constraints (15b) and (15c) cannot be both non-active, i.e., at least one of them is active.*

Proof. We prove this lemma by contradiction. Suppose both constraints (15b) and (15c) are non-active, i.e., $\boldsymbol{\nu} = (\mathbf{0})$, and $\boldsymbol{\mu} = (\mathbf{0})$. Then the optimization problem (14) achieves its maximum when $h_{il}^{(p)} = 1$ for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$. If so, then the left hand size of (15b) equals $|\mathcal{D}|$ which is much greater than B_l for $l \in V$, which is a contradiction. Hence, constraints (15b) and (15c) cannot be both non-active.

From Lemma 1, we know that the feasible region for the Lagrangian multipliers satisfies $\mathcal{R} = \{\nu_l \geq 0, \mu_{ip} \geq 0, \nu_l + \mu_{ip} \neq 0, \forall i \in \mathcal{D}, l \in \{1, \cdots, |p|\}, p \in \mathcal{P}^i\}$.

Theorem 3. *The hit probability $h_{il}^{(p)}$ given in (24) is continuous in ν_l and μ_{ip} for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$ in the feasible region $\mathcal{R}$.*

Proof. From Lemma 1, we know at least one of ν_l and μ_{ip} is non-zero, for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$. Hence there are three cases, (i) $\nu_l \neq 0$ and $\mu_{ip} = 0$; (ii) $\nu_l = 0$ and $\mu_{ip} \neq 0$; and (iii) $\nu_l \neq 0$ and $\mu_{ip} \neq 0$.

For case (i), we have

$$h_{il}^{(p)} = \frac{w_{ip}\psi^{|p|-l}}{\nu_l \left(\prod_{\substack{q:q\neq p,\\ j\in\{1,\cdots,|q|\}}} (1 - h_{ij}^{(q)}) \right)} - \frac{1}{\lambda_{ip}}, \tag{25}$$

which is clearly continuous in ν_l, for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$. Similarly for case (ii), we have

$$h_{il}^{(p)} = \frac{w_{ip}\psi^{|p|-l}}{\mu_{ip}} - \frac{1}{\lambda_{ip}}, \tag{26}$$

which is also clearly continuous in μ_{ip}, for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$.

For case (iii), from (24), it is obvious that $h_{il}^{(p)}$ is continuous in ν_l and μ_{ip} for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$. Therefore, we know that $h_{il}^{(p)}$ is is continuous in ν_l and μ_{ip} for all $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$.

Remark 4. Note that similar arguments (by using Lemma 1) hold true for various other choice of utility functions such as: β- fair utility functions (Sect. 2 [26]). Therefore, the primal-dual algorithm consisting of (19) and (20) converges to the globally optimal solution for a wide range of utility functions.

Algorithm 1 summarizes the details of this algorithm.

Algorithm 1. Primal-Dual Algorithm

Input: $\forall \boldsymbol{\nu_0}, \boldsymbol{\mu_0} \in \mathcal{R}$ and $\boldsymbol{h}_0$
Output: The optimal hit probabilities $\boldsymbol{h}$
Step 0**:** $t = 0$, $\boldsymbol{\nu}[t] \leftarrow \boldsymbol{\nu}_0$, $\boldsymbol{\mu}[t] \leftarrow \boldsymbol{\mu}_0$, $\boldsymbol{h}[t] \leftarrow \boldsymbol{h}_0$
Step $t \geq 1$
while Equation (21) $\neq 0$ **do**
  First, compute $\boldsymbol{h}_{il}^{(p)}[t+1]$ for $i \in \mathcal{D}$, $l \in \{1, \cdots, |p|\}$ and $p \in \mathcal{P}^i$ through (24);
  Second, update $\nu_l[t+1]$ and $\mu_{ip}[t+1]$ through (20) given $\boldsymbol{h}[t+1]$, $\boldsymbol{\nu}[t]$ and $\boldsymbol{\mu}[t]$ for $l \in V$, $i \in \mathcal{D}$ and $p \in \mathcal{P}^i$

5 Numerical Evaluation

In this Section we validate our analytical results with simulations for MCDP across different network topologies.

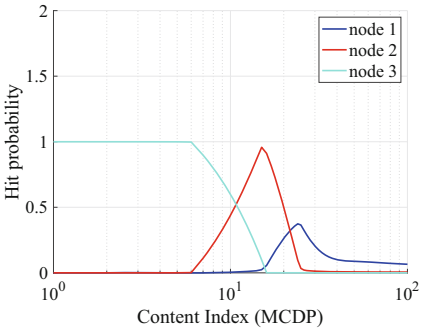

Fig. 2. Hit probability for MCDP in a three-node path.

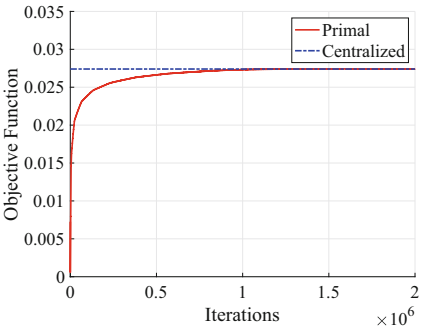

Fig. 3. Convergence of primal algorithm.

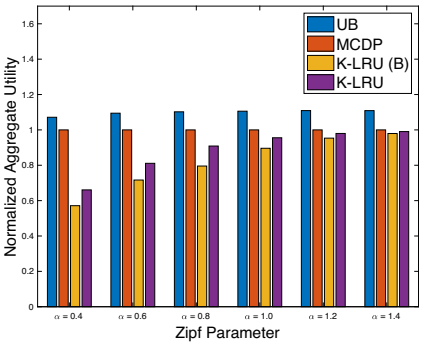

Fig. 4. Optimal aggregated utilities in a three-node path.

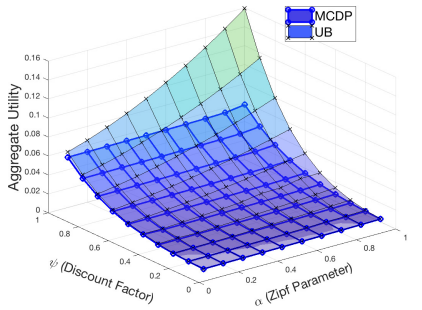

Fig. 5. Aggregate utility under MCDP and UB across different network parameters.

5.1 Linear Edge Network

First, we consider a three-node path with cache capacities $B_l = 10$, $l = 1, 2, 3$. The total number of unique contents considered in the system is $n = 100$. We consider the Zipf popularity distribution with parameter $\alpha = 0.8$. W.l.o.g., we consider log based utility function[3] $U_i(x) = \lambda_i \log(1+x)$ [28], and discount factor $\psi = 0.1$. We assume that requests arrive according to a Poisson process with aggregate request rate $\Lambda = 1$.

We solve optimization problem (8) using a Matlab routine `fmincon`. From Fig. 2, we observe that popular contents are assigned higher hit probabilities at cache node 3, i.e. at the edge cache closest to the user as compared to other caches. The optimal hit probabilities assigned to popular contents at other caches

[3] One can also choose $U_i(x) = \lambda_i \log x$. However, $U_i(x)$ evaluated at $x = 0$ becomes negative infinity, which may produce undesired results while comparing the performance of MCDP with other caching policies.

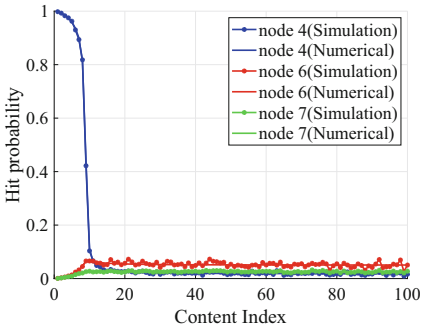

Fig. 6. Hit probability of MCDP under three-layer edge network with distinct contents.

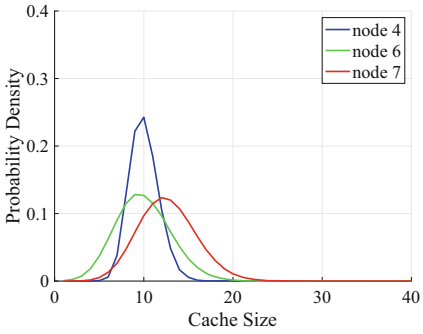

Fig. 7. Cache size pdf of MCDP under three-layer edge network with distinct contents.

are almost negligible. However, the assignment is reversed for moderately popular contents. For non-popular contents, optimal hit probabilities at cache node 1 (closest to origin server) are the highest.

We then implement our primal algorithm given in (13), where we take the following penalty functions [26] $C_l(x) = \max\{0, x - B_l \log(B_l + x)\}$ and $\tilde{C}_i(x) = \max\{0, x - \log(1 + x)\}$. Figure 3 shows that the primal algorithm successfully converges to the optimal solution.

We also compare the performance of MCDP to other policies such as K-LRU (K=3), K-LRU with big cache abstraction: K-LRU(B) and the UB based bound. We plot the relative performance w.r.t. the optimal aggregated utilities of all above policies, normalized to that under MCDP shown in Fig. 4. We observe that MCDP significantly outperforms K-LRU and K-LRU(B) for low and moderate values of Zipf parameter. Furthermore, the performance gap between UB and MCDP increases in Zipf parameter.

Finally, we consider the effect of α and ψ on the performance gap (difference between optimal aggregate utility) between UB and MCDP. We present the simulation results in Fig. 5. Note that utility under both UB and MCDP increases in either α or ψ or both. When either α or ψ is small, irrespective of the value of the other, the performance gap is minor or negligible. However, the gap is considerably large when both α and ψ are high. We believe, due to high communication overhead between successive layers of caches, ψ has a small value, thus indicating a minor performance gap between MCDP and UB.

5.2 General Edge Network with Non-Common Requested Contents

We consider a three-layer edge network shown in Fig. 1 with node set $\{1, \cdots, 7\}$, which is consistent with the YouTube video delivery system [23]. Nodes 1-4 are edge caches, and node 7 is tertiary cache. There exist four paths $p_1 = \{1, 5, 7\}$, $p_2 = \{2, 5, 7\}$, $p_3 = \{3, 6, 7\}$ and $p_4 = \{4, 6, 7\}$. Each edge cache serves requests for 100 distinct contents, and cache size is $B_v = 10$ for $v \in \{1, \cdots, 7\}$. Assume

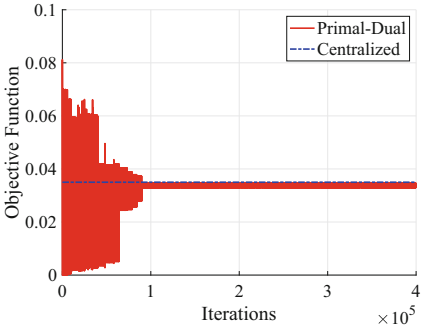

Fig. 8. Convergence of Primal-Dual algorithm.

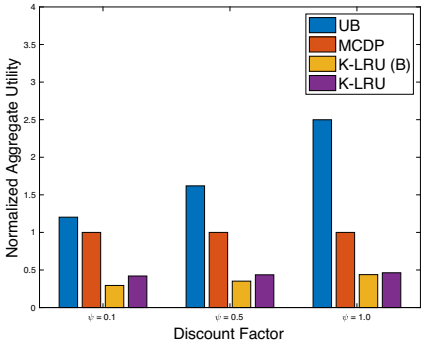

Fig. 9. Optimal aggregated utilities under common requested contents.

that content follows a Zipf distribution with parameter $\alpha_1 = 0.2$, $\alpha_2 = 0.4$, $\alpha_3 = 0.6$ and $\alpha_4 = 0.8$, respectively. We consider utility function $U_{ip}(x) = \lambda_{ip} \log(1 + x)$, where λ_{ip} is the request arrival rate for content i on path p, and requests are described by a Poisson process with $\Lambda_p = 1$ for $p = 1, 2, 3, 4$. The discount factor $\psi = 0.1$.

Figure 6 shows results for path $p_4 = \{4, 6, 7\}$. From Fig. 6, we observe that our algorithm yields the exact optimal and empirical hit probabilities under MCDP. Figure 7 shows the probability density for the number of contents[4] in the edge network. As expected, the density is concentrated around their corresponding cache sizes.

5.3 General Edge Network with Common Requested Contents

We evaluate the performance of Algorithm 1 on a three-layer edge network shown in Fig. 1. We assume that there are totally 100 unique contents in the system requested from four paths. The cache size is given as $B_v = 10$ for $v = 1, \cdots, 7$. We consider the utility function $U_{ip}(x) = \lambda_{ip} \log(1 + x)$, and the popularity distribution over these contents is Zipf with parameter 0.8. W.l.o.g., the aggregate request arrival rate is one. The discount factor $\psi = 0.1$.

We solve the optimization problem in (15) using a Matlab routine `fmincon`. Then we implement our primal-dual algorithm given in Algorithm 1. The result for aggregate optimal utility is presented in Fig. 8. It is clear that the primal-dual algorithm successfully converges to the optimal solution.

Similar to Sect. 5.1, we compare MCDP to K-LRU, K-LRU(B) and UB. Figure 9 compares the performance of different eviction policies to our MCDP policy. We plot the relative performance w.r.t. the optimal aggregated utilities of all above policies, normalized to that under MCDP. We again observe a huge

[4] The constraint (14b) in problem (14) is on average cache occupancy. However it can be shown that if $n \to \infty$ and B_l grows in sub-linear manner, the probability of violating the target cache size B_l becomes negligible [8].

gain of MCDP w.r.t. K-LRU and K-LRU(B) across all values of discount factor. However, the performance gap between MCDP and UB increases with an increase in the value of discount factor.

5.4 Applications to General Graphs

Here we consider a general network topology with overlapping paths and common contents requested along different paths. We consider two network topologies: Grid and lollipop. *Grid* is a two dimensional square grid while a (a,b) *lollipop* network is a complete graph of size a, connected to a path graph of length b. Denote the network as $G = (V, E)$. For grid, we consider $|V| = 16$, while we consider a $(3, 4)$ lollipop topology with $|V| = 7$ and clique size 3. The library contain $|\mathcal{D}| = 100$ unique contents. We assign a weight to each edge in E, selected uniformly from the interval $[1, 20]$.

Next, we generate a set of requests in G as described in [16]. To ensure that paths overlap, we randomly select a subset $\tilde{V} \subset V$ nodes to generate requests. Each node in $\tilde{V}$ can generate requests for contents in $\mathcal{D}$ following a Zipf distribution with parameter $\alpha = 0.8$. Requests are then routed over the shortest path between the requesting node in $\tilde{V}$ and the node in V that caches the content. Again, we assume that the aggregate request rate at each node in $\tilde{V}$ is one and the discount factor to be $\psi = 0.1$.

Table 1. Optimal aggregate utilities under various network topologies.

Topology	α	MCDP	UB	% Gap
Grid	0.8	0.0923	0.1043	11.50
Grid	1.2	0.3611	0.4016	10.08
Lollipop	0.8	0.0908	0.1002	9.38
Lollipop	1.2	0.3625	0.4024	9.91

We evaluate the performance of MCDP over the graphs across various Zipf parameter in Table 1. It is clear that for both network topologies, aggregate utility obtained from our TTL-based framework with MCDP policy is higher for higher zipf parameter as compared to lower zipf parameter. With increase in Zipf parameter, the difference between request rates of popular and less popular contents increases. The aggregate request rate over all contents is the same in both cases. Thus popular contents get longer fraction of rates which in turn yields higher aggregate utility.

6 Related Work

There is a rich literature on the design, modeling and analysis of cache networks, including TTL caches [3,13,24] and optimal caching [16,20]. In particular, Rodriguez et al. [24] analyzed the advantage of pushing content upstream,

Berger et al. [3] characterized the exactness of TTL policy in a hierarchical topology. A unified approach to study and compare different caching policies is given in [14] and an optimal placement problem under a heavy-tailed demand has been explored in [12].

Dehghan et al. [9] as well as Abedini and Shakkottai [1] studied joint routing and content placement with a focus on a bipartite, single-hop setting. Both showed that minimizing single-hop routing cost can be reduced to solving a linear program. Ioannidis and Yeh [17] studied the same problem under a more general setting for arbitrary topologies.

An adaptive caching policy for a cache network was proposed in [16], where each node makes a decision on which item to cache and evict. An integer programming problem was formulated by characterizing the content transfer costs. Both centralized and complex distributed algorithms were designed with performance guarantees. This work complements our work, as we consider TTL cache and control the optimal cache parameters through timers to maximize the sum of utilities over all contents across the network. However, [16] proposed only approximate algorithms while our timer-based models enable us to design optimal solutions since content occupancy can be modeled as a real variable.

Closer to our work, a utility maximization problem for a single cache was considered under IRM [8,21] and stationary requests [22], while [12] maximized the hit probabilities under heavy-tailed demands over a single cache. None of these approaches generalizes to edge networks, which leads to non-convex formulations (See Sect. 4.2 and Sect. 4.3); addressing this lack of convexity in its full generality, for arbitrary network topologies, overlapping paths and request arrival rates, is one of our technical contributions.

7 Conclusion

We constructed optimal timer-based TTL polices for content placement in edge networks through a unified optimization approach. We formulated a general utility maximization framework, which is non-convex in general. We identified the non-convexity issue and proposed efficient distributed algorithm to solve it. We proved that the distributed algorithms converge to the globally optimal solutions. We showed the efficiency of these algorithms through numerical studies. An analysis of the MCDP algorithm with more general request arrivals would remain our future work.

Acknowledgment. This research was sponsored by the U.S. ARL and the U.K. MoD under Agreement Number W911NF-16-3-0001 and by the NSF under Grant CNS-1617437 and CRII-CNS-2104880. This research was also supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office Award Number DE-EE0009341. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, U.S. ARL, the U.S. Government, the U.K. MoD or the U.K. Government.

A Appendix

A.1 Stationary Behaviors of MCDP

[15] considered a caching policy LRU($\boldsymbol{m}$). Though the policy differ from MCDP, the stationary analysis is similar. Under IRM model, the request for content i arrives according a Poisson process with rate λ_i. As discussed earlier, for TTL caches, content i spends a deterministic time in a cache if it is not requested, which is independent of all other contents. We denote the timer as T_{il} for content i in cache l on the path p, where $l \in \{1, \cdots, |p|\}$.

Denote t_k^i as the k-th time that content i is either requested or moved from one cache to another. For simplicity, we assume that content is in cache 0 (i.e., server) if it is not in the cache network. Then we can define a discrete time Markov chain (DTMC) $\{X_k^i\}_{k\geq 0}$ with $|p|+1$ states, where X_k^i is the cache index that content i is in at time t_k^i. Since the event that the time between two requests for content i exceeds T_{il} happens with probability $e^{-\lambda_i T_{il}}$, then the transition matrix of $\{X_k^i\}_{k\geq 0}$ is given as

$$\mathbf{P}_i^{\text{MCDP}} = \begin{bmatrix} 0 & 1 & & & \\ e^{-\lambda_i T_{i1}} & 0 & 1-e^{-\lambda_i T_{i1}} & & \\ & \ddots & \ddots & \ddots & \\ & & e^{-\lambda_i T_{i(|p|-1)}} & 0 & 1-e^{-\lambda_i T_{i(|p|-1)}} \\ & & & e^{-\lambda_i T_{i|p|}} & 1-e^{-\lambda_i T_{i|p|}} \end{bmatrix}. \tag{27}$$

Let $(\pi_{i0}, \cdots, \pi_{i|p|})$ be the stationary distribution for $\mathbf{P}_i^{\text{MCDP}}$, we have

$$\pi_{i0} = \frac{1}{1+\sum_{j=1}^{|p|} e^{\lambda_i T_{ij}} \prod_{s=1}^{j-1}(e^{\lambda_i T_{is}}-1)}, \tag{28a}$$

$$\pi_{i1} = \pi_{i0} e^{\lambda_i T_{i1}}, \tag{28b}$$

$$\pi_{il} = \pi_{i0} e^{\lambda_i T_{il}} \prod_{s=1}^{l-1}(e^{\lambda_i T_{is}}-1),\ l = 2, \cdots, |p|. \tag{28c}$$

The average time that content i spends in cache $l \in \{1, \cdots, |p|\}$ is given by

$$\mathbb{E}[t_{k+1}^i - t_k^i | X_k^i = l] = \int_0^{T_{il}} \left(1 - \left[1 - e^{-\lambda_i t}\right]\right) dt = \frac{1-e^{-\lambda_i T_{il}}}{\lambda_i}, \tag{29}$$

and $\mathbb{E}[t_{k+1}^i - t_k^i | X_k^i = 0] = \frac{1}{\lambda_i}$. Given (28) and (29), the timer-average probability that content i is in cache $l \in \{1, \cdots, |p|\}$ is given by (2) with h_{il} being the hit probability for content i at cache l.

References

1. Abedini, N., Shakkottai, S.: Content caching and scheduling in wireless networks with elastic and inelastic traffic. IEEE/ACM Trans. Netw. **22**(3), 864–874 (2014)

2. Baccelli, F., Brémaud, P.: Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences, vol. 26. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-662-11657-9
3. Berger, D.S., Gland, P., Singla, S., Ciucu, F.: Exact analysis of TTL cache networks. Perf. Eval. **79**, 2–23 (2014)
4. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
5. Che, H., Tung, Y., Wang, Z.: Hierarchical web caching systems: modeling, design and experimental results. IEEE J. Sel. Areas Commun. **20**(7), 1305–1314 (2002)
6. Cisco, V.: Cisco Visual Networking Index: Forecast and Methodology 2014–2019 White Paper. Technical Report, Cisco (2015)
7. Coffman, E.G., Denning, P.J.: Operating Systems Theory, vol. 973. Prentice-Hall, Englewood Cliffs (1973)
8. Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., Tay, Y.: A utility optimization approach to network cache design. In: Proceedings of IEEE INFOCOM (2016)
9. Dehghan, M., et al.: On the complexity of optimal routing and content caching in heterogeneous networks. In: Proceedings of IEEE INFOCOM, pp. 936–944 (2015)
10. Fagin, R.: Asymptotic miss ratios over independent references. J. Comput. Syst. Sci. **14**(2), 222–250 (1977)
11. Feldman, M., Chuang, J.: Service differentiation in web caching and content distribution. In: Proceedings of CCN (2002)
12. Ferragut, A., Rodríguez, I., Paganini, F.: Optimizing TTL caches under heavy-tailed demands. In: Proceedings of ACM SIGMETRICS (2016)
13. Fofack, N.C., Nain, P., Neglia, G., Towsley, D.: Performance evaluation of hierarchical TTL-based cache networks. Comput. Netw. **65**, 212–231 (2014)
14. Garetto, M., Leonardi, E., Martina, V.: A unified approach to the performance analysis of caching systems. ACM TOMPECS **1**(3), 12 (2016)
15. Gast, N., Houdt, B.V.: Asymptotically exact TTL-approximations of the cache replacement algorithms LRU(m) and h-LRU. In: ITC, vol. 28 (2016)
16. Ioannidis, S., Yeh, E.: Adaptive caching networks with optimality guarantees. In: Proceedings of ACM SIGMETRICS, pp. 113–124 (2016)
17. Ioannidis, S., Yeh, E.: Jointly optimal routing and caching for arbitrary network topologies. In: Proceedings of ACM ICN, pp. 77–87 (2017)
18. Jiang, W., Ioannidis, S., Massoulié, L., Picconi, F.: Orchestrating massively distributed CDNs. In: Proceedings of ACM CoNEXT, pp. 133–144 (2012)
19. Kyparisis, J.: On uniqueness of Kuhn-Tucker multipliers in nonlinear programming. Math. Program. **32**(2), 242–246 (1985)
20. Li, J., et al.: DR-cache: distributed resilient caching with latency guarantees. In: Proceedings of IEEE INFOCOM (2018)
21. Panigrahy, N.K., Li, J., Towsley, D.: Hit rate vs. hit probability based cache utility maximization. In: Proceedings of ACM MAMA (2017)
22. Panigrahy, N.K., Li, J., Towsley, D.: Network cache design under stationary requests: exact analysis and poisson approximation. In: Proceedings of IEEE MASCOTS (2018)
23. Ramadan, E., Narayanan, A., Zhang, Z.L., Li, R., Zhang, G.: Big cache abstraction for cache networks. In: Proceedings Of IEEE ICDCS (2017)
24. Rodríguez, I., Ferragut, A., Paganini, F.: Improving performance of multiple-level cache systems. In: SIGCOMM (2016)
25. Rosensweig, E.J., Kurose, J., Towsley, D.: Approximate models for general cache networks. In: Proceedings of IEEE INFOCOM (2010)

26. Srikant, R., Ying, L.: Communication Networks: an Optimization, Control, and Stochastic Networks Perspective. Cambridge University Press, Cambridge (2013)
27. Tychogiorgos, G., Gkelias, A., Leung, K.K.: A non-convex distributed optimization framework and its application to wireless ad-hoc networks. IEEE Trans. Wirel. Commun **12**(9), 4286–4296 (2013)
28. Vecer, J.: Dynamic scoring: probabilistic model selection based on utility maximization. In: SSRN (2018)