2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)

Caching dynamic contents with varying popularity

Anu Krishna, Ramya Burra, Chandramani Singh

Department of Electronic Systems Engineering
Indian Institute of Science
Bangalore 560012, India
Email: {vasanthakuma, burra, chandra}@iisc.ac.in

***Abstract*—We study content caching in a cellular network consisting of a base station with a cache. New contents arrive in the network according to a Poisson process and the contents stay for exponentially distributed times. At any given time, all the contents in the network have the same popularity or request rate. Also, contents' request rates are time-varying, instantaneous request rates being a decreasing function of the number of contents in the network. Precaching contents at the base station incurs a cost. However, fetching contents from the server on being requested incurs an even higher cost. We formulate caching problem as a Markov decision process and derive the optimal caching policy. We also propose a Reinforcement Learning based algorithm that yields precaching decisions when system parameters are unknown. Numerical results show that the proposed algorithm's time averaged cost is close to the optimal average cost.**

I. Introduction

With the widespread use of smartphones and laptops with Internet connectivity, the Internet backhaul is experiencing a severe overload due to the enormous demand for downloading multimedia contents such as video. A significant portion of multimedia traffic is due to recurrent transmissions of a few popular contents, e.g., popular music videos, which lead to a huge amount of redundant traffic. However, the current IP based Internet, owing to its end-point based communication model, is inept for content distribution services. This fundamental mismatch hurts network performance in terms of end-user's quality of experience, bandwidth costs, delay, and energy consumption. One can easily tackle this by caching contents at the Base Stations.

Adding cache to the Base Station invariably offers several advantages:

a) Reduces latency: On caching contents at a nearby base station, users encounter a less delay in fetching these from the base station than fetching from the central server.

b) Saves network resources: Caching reduces traffic over the back-haul link that connect BS to the server.

However, caching involves a cost.

c) Cost of caching infrastructure: Caching all the contents from the server at the base station is not desirable because the caches have finite capacity. Therefore, unnecessary caching of contents adds to the bulk of infrastructure.

d) Cost of precaching: Caching content from the server always incurs certain caching cost. Therefore, caching content that would never be requested is undesirable. There are two common caching strategies [1].

The first author and the second author are supported by Visvesvaraya PhD Scheme and the third author by INSPIRE Faculty Research Grant (DSTO-1363).

1) Reactive caching: Contents are cached only on request. For example: Least Recently Used (LRU), Least Frequently used (LFU) are reactive caching policies used to replace existing contents in cache with the requested content.
2) Proactive caching: Contents may be proactively cached even before being requested based on their popularity.

Reactive caching leads to more cache miss. Since cellular networks face increasingly unprecedented change in demand of contents, proactive caching is preferred to reactive caching. This reduces cache miss and in long term saves cost [2]. Therefore, in view of the above discussion, we aim to study proactive caching policies. In our work, we study caching problem in the context of dynamic contents. Throughout this work, the contents have time varying popularity. We also propose a reinforcement learning algorithm to decide precaching decisions when system parameters are unknown.

A. Related Work

Plenty of literature is available dedicated to caching in the context of cellular networks.

The authors of [3] address the problem of content replication and request routing in a distributed caching system. In this paper, the number of contents at the server scales with the cache size. The authors in [4] provide algorithms for cache content update in a cellular network, motivated by Gibbs sampling techniques. In this paper, the number of contents at the server is fixed. The request rate for the contents follow a time homogeneous process.

The authors in [5] and [6] study content placement of fixed collection of contents in a cellular network, over geographically distributed caches. The authors in [7] propose an intelligent proactive caching scheme to cache fixed collection of contents, in online social network, to reduce the energy expenditure while downloading the contents.

The authors in [8] propose a dynamic probabilistic caching for the scenario when the instantaneous content popularity may vary with time. However, the paper assumes that the average popularity is known. The authors in [9] propose a caching algorithm to derive the accurate optimal caching probabilities, which satisfy the caching capacity constraint of every caching device. The authors in [10] propose optimal dynamic schemes to minimise the expected network cost aggregated across caching entities, contents and time instants. The authors in [11] provides a caching policy that is guaranteed to learn-and-adapt to unknown policies of leaf nodes and space-time evolution of contents requests. The authors in [12] study *reactive* caching of dynamic contents.

ISBN 978-3-903176-37-9 © 2021 IFIP

All the works from [3] to [13], study proactive caching problem with a huge catalogue of *fixed number* of contents at the server. However, in reality, the number of contents at the server evolve [14]. To the best of our knowledge, there has been little work that focuses on modelling the temporal evolution of contents.

B. Contributions

We consider content caching at a base station when the number of contents at the server and their popularities evolve with time. We characterize the optimal caching policy for a few variants of this problem. We first consider a special scenario where all the contents have the same requests rate which evolves over time. We see that, it is enough to consider optimal caching of single content in this case. We then extend the results to heterogeneous contents. We study these problems under the assumption that the parameters guiding evolution and popularity of contents are known a priori. However, in reality, these parameters are unknown. So, we also propose a *reinforcement learning* based algorithm to make precaching decisions for dynamic contents when the system parameters are unknown.

C. Organization

The rest of the paper is organized as follows. In Section II, the system model is defined for the caching problem where the contents are all homogeneous. That is, all content have same costs and popularity associated with it. In Section III, we argue that it suffices to obtain optimal caching policy for any single content. In the context of single content, we formulate the optimal caching problem as a discrete time controlled Markov Chain. We derive the optimal caching policy for both infinite and finite cache sizes. In Section IV, we discuss the caching problem in the context of heterogeneous contents at the server. That is, each content has different cost and popularity associated with it. We provide optimal caching policy in this context. In Section V, we provide a Reinforcement Learning based algorithm for caching in the context of unknown system parameters. Finally, we provide numerical results in Section VI.

II. System Model

In this section, we define the system model for the caching problem.

a) Network Model: We consider a cellular network of server and a single Base Station (BS) such that the BS has users associated with it. A BS has a cache of size B. In our model, the server can host a huge catalogue of contents, and the total number of contents in the server evolve with time. A user can place a content request to its Base Station. Contents can be precached at the cache from server. The contents that are not precached are automatically cached at cache, on being requested and served, if there is space. If the content is already cached at the BS, the user can download it from the BS. We assume that there is no delay involved in downloading content directly from BS.

b) Dynamic Contents: Below, we present the details of content evolution.

1) *Arrival of contents*: The content arrival at the server is a Poisson process of rate λ. The authors in [12] also used a similar arrival process.
2) *Exit of contents*: Each content stays at the server for a life time that has an exponential distribution of rate μ.

Most of the previous work, for example, [15], [5], [2], [16], had assumed fixed number of contents. This can be modelled by assuming mean life times of the contents much larger than the time scale of requests.

c) Popularity or Request rate: Unlike existing works, which assume that any contents' popularity or request rate remains constant throughout its lifetime, we let contents' popularity vary. *In particular, we assume that all the contents have same popularity or request rate at any given time. But their instantaneous request rates are function of number of contents at the server*. We denote request rate by $r(n)$. As explained in [14], the popularity of a content evolves with time and they compete for attention. As the attention for a content depends on the total number of contents, attention decreases as n increases. So we assume that $r(n)$ decreases as n increases and $\lim_{n\to\infty} r(n) = 0$. For e.g., $r(n) = \frac{r_0}{n^\alpha}, \frac{r_0}{\log n}$ where $r_0 > 0,\ \alpha \in [0, 1]$.

d) Content Delivery Costs: The cost of caching a content at the cache from the server includes

1) *Communication Cost*: The communication cost essentially includes power (or bandwidth) cost. It depends on the server location. In general, it also depends on the contents. We use c to denote this cost.
2) *Content Delay Cost*: If the requested content is available at the BS, then this cost includes the look-up cost. If the requested content is not available at the BS, in addition to the above cost, a cost d modeling the user's wait time is also incurred.

Therefore, a content precached incurs a cost c and a content that is procured on request incurs a cost $c + d$.

A. Heterogenous contents

In the context of homogeneous contents, all the costs are identical. However, in reality the caching cost involved in caching a content from a server near by is different from the caching a content that is located at a distant server. According to [17], recent experimental studies show that popularity evolution of different contents can be clustered into few classes that exhibit similar temporal popularity profiles. Therefore, contents may be grouped into various class. Each class may have its own parameters like $c, d, r(\cdot)$ etc.

Our aim is to study system with heterogeneous contents. We obtain an optimal policy in this context. To obtain the optimal policy for heterogeneous contents, we use results from homogeneous content variation. Therefore, we first focus on homogeneous content problem. And later, we extend the results to heterogeneous content problem in Section IV.

B. Optimal Caching Problem

We see from the system model that it is beneficial to precache a content if it is likely to be requested during its lifetime. Let us also observe from II-0c, that if the number of contents at the server remain high throughout the lifetime of a content, this content is less likely to be requested. Clearly, the precaching decision for a content at anytime should also depends on the expected future evolution of the number of contents. This evolution, being Markovian, is independent of the past given the current state.

The optimal caching problem entails determining whether or not to precache an uncached content given the system state.

ISBN 978-3-903176-37-9 © 2021 IFIP

Notice that a content, once cached, either due to precaching or on being requested remains in the cache until it exits the system. Also, once a content is cached or exits the system, no further cost is incurred on this content.

We now introduce some notation to formalise the problem. Notice that the total cost is the sum of the costs incurred on contents. Let us index the contents in the order of their arrival. Let τ_i denote the time at which i^{th} content is cached or exits without being cached. Also, L_i be the cost associated with i^{th} content. L_i is defined as follows.

$$L_i = \begin{cases} c, \text{ if } i^{\text{th}} \text{ content is precached at } \tau_i \\ c+d, \text{if } i^{\text{th}} \text{ content is requested and cached at } \tau_i \\ 0, \text{if } i^{\text{th}} \text{ content exits at } \tau_i \text{ without being cached} \end{cases}$$

Let us also define $\mathcal{A}(T)$, which is the set of contents till time T.

$$\mathcal{A}(T) = \{i : \tau_i \leq T\}$$

We aim to minimize the following time-averaged cost of downloading the contents to the users.

$$\lim_{T\to\infty} \frac{1}{T}\, \mathbb{E}\left(\sum_{i\in\mathcal{A}(T)} L_i\right).$$

Let $\mathcal{N}(t)$ be the set of contents at the server at time t. $N(t) = |\mathcal{N}(t)|,\ t \geq 0$. Further, let $\mathcal{B}(t)$ be the set of cached contents at time t. While $\mathcal{N}(t),\ t \geq 0$, are independent of the caching decisions, $\mathcal{B}(t),\ t \geq 0$, are dependent on these caching decisions. Since all the contents have equal popularity at any given time, a cached content is removed from the cache only when it exits the system. In Section III, we formulate the optimal caching problem as a discrete time controlled Markov Chain. We derive the optimal caching policy. In the following we derive results for infinite cache varient i.e., $B = \infty$. We then extend these results to finite cache size in Section III-B.

III. Optimal Caching Policy

We see that the total cost is sum of the costs incurred on individual contents. As mentioned in the previous section, the precaching decision for a content at anytime should also depends on the expected future evolution of the number of contents. Therefore, the cost incurred on the i^{th} content depends only on the evolution of $N(t)$ until τ_i, and not on the caching decisions on other contents. Hence, we can minimize the time averaged cost by individually optimizing the costs incurred on different contents. In the following, we fix a content, referred to as the tagged content, and minimize the cost incurred on it.

A. Single Content Problem

Let us consider a tagged content and without loss of generality, assume that it arrives at time 0. Recall that $N(t),\ t \geq 0$, represent the number of contents at the server. Precaching decisions depend on $N(t),\ t \geq 0$. We thus consider a system with $N(t),\ t \geq 0$, until the content is cached or exited. The cost incurred on the tagged content depends on whether it is precached, is requested before being cached or is exited before being precached or requested. Moreover, once any of these epochs happens, no cost is incurred on the tagged content in the future, and the problem terminates. Accordingly, we define terminal states $C,\ R,$ and E as follows.

1) State C is encountered if the tagged content is precached.
2) State R is encountered if the tagged content is requested.
3) State E is encountered if the tagged content exits.

We thus have a system with a state space $\mathcal{S} = \{1, 2, 3, 4, ...\} \cup \{C, R, E\}$. This is an absorbing Markov Chain.

Let $X(t),\ t \geq 0$, denote the state, $X(t) = N(t)$ until the termination epoch, say τ. $X(\tau) \in \{C, R, E\}$ and $X(t) = X(\tau), \forall t \geq \tau$. Caching problem can be cast as a continuous time Markov Decision Process. However, the problem is essentially the same as the discrete time problem with the same transition probabilities. We describe the discrete time Markov Decision Process as follows.

State Space: $\mathcal{S} = \{1, 2, 3, 4, ...\} \cup \{C, R, E\}$

Action Space: $\mathcal{A} = \{0, 1\}$, 1 represents precaching whereas 0 denotes not precaching.

State Transition: Transition Probability Matrix P, such that such that $P_{i,j}(a)$ is the transition probability from the state i to j when you choose action a. Note that we can take actions in states $\{1, 2, 3, ...\}$ only. $P_{i,j}(a)$, at $a = 0$ is given by

$$P_{i,j}(0) = \begin{cases} \frac{(i-1)\mu}{\lambda+i\mu+r(i)}, & i = \{1, 2, 3, ...\},\ j = i-1 \\ \frac{\lambda}{\lambda+i\mu+r(i)}, & i = \{1, 2, 3, ...\},\ j = i+1 \\ \frac{r(i)}{\lambda+i\mu+r(i)}, & i = \{1, 2, 3, ...\},\ j = R \\ \frac{\mu}{\lambda+i\mu+r(i)}, & i = \{1, 2, 3, ...\},\ j = E \\ 0, & \text{otherwise.} \end{cases}$$

$P_{i,j}(a)$, at $a = 1$ is given by

$$P_{i,j}(1) = \begin{cases} 1, & i = \{1, 2, 3, ...\},\ j = C \\ 0, & \text{otherwise.} \end{cases}$$

Average cost per stage: Average cost at state $i \in \mathcal{S}$ when action $a \in \mathcal{A}$ is chosen is $g(i, a)$.

$$g(i,a) = \begin{cases} \frac{r(i)}{r(i)+i\mu+\lambda}(c+d), & a = 0 \\ c, & a = 1 \end{cases} \tag{1}$$

Policy: It is a sequence of functions $\bar{\pi} = (\pi_1, \pi_2, ...)$, $\pi_i : \{1, 2, 3, ..\} \to \{0, 1\}, \forall i$. A stationary policy $\bar{\pi}$ is of the form $\bar{\pi} = (\pi, \pi, \pi...)$.

Objective function: Let the content i be the tagged content. Since the goal is to minimise the average cost of downloading the content i, the objective function is given by

$$\mathbb{E}_\pi\left(\sum_{t=0}^{\tau_i} g(N(t), \pi)\right).$$

As cost per stage (see (1)) in this formulation is non negative and action set is finite for every state, Proposition 3.1.3 from [18, Chapter-3] implies existence of an optimal stationary policy (see the paragraph following proof of Proposition 3.1.3 in [18, Chapter-3]). Thus, this problem assumes solution in the class of stationary policies. Hence we look at the policies of the form $(\pi, \pi, \pi...)$; for brevity we use π to denote this policy. A stationary policy π is called a threshold policy if

$$\pi(n) = \begin{cases} 1, n \leq \bar{n} \\ 0, n > \bar{n} \end{cases}$$

for an integer $\bar{n} \geq 0$. $\bar{n}$ is referred as the threshold of π. We refer to a threshold policy with threshold $\bar{n}$ as policy $\pi^{\bar{n}}$. Let π^* denote the optimal policy.

ISBN 978-3-903176-37-9 © 2021 IFIP

Bellman equation: Let $V : \{1, 2, 3..\} \rightarrow \mathbb{R}_+$ be the optimal cost function for the problem. It is the solution of the following Bellman's equation. For all $n \in \{1, 2, 3, ..\}$, [1]

$$V(n) = \min_{a\in\{0,1\}} \left\{ g(n, a) + \sum_{j\in\mathcal{S}} P_{n,j}(a)V(j) \right\}. \quad (2)$$

Substituting the values of $g(n, a)$ and $P_{n,j}(a)$ in (2), and using the fact that on exit no cost is incurred on the tagged content, the following can be observed.

$$V(n) = \min\Bigg\{ c, \quad \frac{r(n)}{r(n) + n\mu + \lambda}(c + d) + \frac{(n-1)\mu}{r(n) + n\mu + \lambda} V(n-1) + \frac{\lambda}{r(n) + n\mu + \lambda} V(n+1) \Bigg\}.$$

We also define the following

$$V_{\pi^{\bar{n}}}(n) = \begin{cases} c, & n \leq \bar{n} \\ \frac{r(n)}{r(n)+n\mu+\lambda}(c + d) & \\ +\frac{(n-1)\mu}{r(n)+n\mu+\lambda} V_{\pi^{\bar{n}}}(n-1) & \\ +\frac{\lambda}{r(n)+n\mu+\lambda} V_{\pi^{\bar{n}}}(n+1), & n > \bar{n}. \end{cases} \quad (3)$$

In the following lemma, we show that the optimal policy is a threshold policy. We first focus on infinite cache size (i.e., $B = \infty$) case. In Section III-B, we consider finite cache sizes.

The following lemma establishes that the optimal policy, π^*, is a threshold policy.

Lemma 3.1: The optimal policy is a threshold policy when $B = \infty$. In other words, $\pi^* = \pi^{\bar{n}}$ for some $\bar{n} \geq 0$.

Proof: See [19, Appendix A]. ∎

We next define n^* as

$$n^* := \min\{n : V_{\pi^n}(n + 1) \leq c\}. \quad (4)$$

We prove that π^{n^*} is an optimal policy. The optimality proof uses the following lemma which in particular establishes that $n^* < \infty$.

Lemma 3.2: If $V_{\pi^{\bar{n}}}(\bar{n} + 1) \geq c$, then

$$V_{\pi^{\bar{n}+1}}(\bar{n} + 2) < V_{\pi^{\bar{n}}}(\bar{n} + 1).$$

Moreover, if $\mu c > r(n)d$, then $V_{\pi^n}(n + 1) \leq c$.

Proof: See [19, Appendix B]. ∎

It is observed from Lemma 3.2 that $V_{\pi^n}(n + 1)$ decreases in n only when $V_{\pi^n}(n + 1) \geq c$. Since $\lim_{n\rightarrow\infty} r(n) = 0$, there exist a $\bar{n} < \infty$ such that $r(\bar{n}) < \frac{\mu c}{d}$. Then, from the above lemma, $V_{\pi^{\bar{n}}}(\bar{n} + 1) \leq c$. Consequently, $n^* \leq \bar{n}$.

We also need the following lemma which says that, under certain conditions, both n^* and $n^* + 1$ may yield same cost.

Lemma 3.3: If $V_{\pi^{n^*}}(n^* + 1) = c$, then $V_{\pi^{n^*+1}}(n) = V_{\pi^{n^*}}(n)$, $\forall\ n$.

Proof: See [19, Appendix C]. ∎

The following theorem establishes optimality of π^{n^*}.

Theorem 3.1:

(a) If $V_{\pi^{n^*}}(n^* + 1) < c$, policy π^{n^*} is the unique optimal policy.
(b) If $V_{\pi^{n^*}}(n^* + 1) = c$, both, policy π^{n^*} and policy π^{n^*+1}, are optimal.

Proof: See [19, Appendix D]. ∎

B. Finite Cache Sizes

In reality, caches have only finite capacity (i.e., $B < \infty$). Thus in this subsection, we extend our analysis when the cache size is finite. Let us define $n^*_B = \min\{n^*, B\}$ and $n^*_{B_1} = \min\{n^* + 1, B\}$. The following Proposition yields the optimal policy for $B < \infty$.

Proposition 3.1:

(a) If $V_{\pi^{n^*_B}}(n^*_B + 1) < c$, policy $\pi^{n^*_B}$ is the unique optimal policy.
(b) If $V_{\pi^{n^*_B}}(n^*_B + 1) = c$, both, policy $\pi^{n^*_B}$ and policy $\pi^{n^*_{B_1}}$, are optimal.

Proof: See [19, Appendix E]. ∎

C. Computation of $V_{\pi^n}(n + 1)$

The goal of this subsection is to determine n^*. Recollect that, to determine n^* one needs to evaluate $V_{\pi^n}(n+1)$, $\forall n \in \{1, 2, ..\}$ (See (4)). In the following, we provide the details of computation of $V_{\pi^n}(n + 1), \forall n \in \{1, 2, 3, ...\}$. Note that $V_{\pi^n}(n + 1)$ can be written as

$$V_{\pi^n}(n + 1) = \mathbb{P}_{req}(c + d) + \mathbb{P}_c c$$

where $\mathbb{P}_{req}$ is the probability that a tagged content is requested when there are $n + 1$ content at the server, and $\mathbb{P}_c$ is the probability that a tagged content is copied when there are $n+1$ content at the server. We cannot compute $\mathbb{P}_{req}$ and $\mathbb{P}_c$ exactly. Since there are $n + 1$ contents at the server, under policy π^n, when the number of contents drop to n, we hit the terminal state and the problem ends. Therefore, to obtain $\mathbb{P}_{req}$ and $\mathbb{P}_c$ approximately, we truncate the state space of $X(t)$ to $\{n, n + 1, n+2, ., N\} \cup \{R, E\}$ for a sufficiently large integer N . Note that $C = n$. Let us call the truncated CTMC, $X^N(t), t \geq 0$.

The transition probability matrix of the embedded Markov Chain of $X^N(t), t \geq 0$, say P^N, will have the following form

$$P^N = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix}$$

where,

1) Q is a $N - n \times N - n$ matrix with elements $Q_{i,j}$ given by:

$$Q_{i,j} = \begin{cases} \frac{\lambda}{r(i)+i\mu+\lambda}, & i \in \{n + 1, n + 2, ..., N - 1\}, j = i + 1 \\ \frac{(i-1)\mu}{r(i)+i\mu+\lambda}, & i \in \{n + 2, n + 2, ..., N\}, j = i - 1 \\ 0, & \text{otherwise} \end{cases}$$

2) R is a $N - n \times 3$ matrix with elements $R_{i,j}$ given by:

$$R_{i,j} = \begin{cases} \frac{\mu}{r(i)+i\mu+\lambda}, & i \in \{n + 1, n + 2, ..., N\}, j = E \\ \frac{r(i)}{r(i)+i\mu+\lambda}, & i \in \{n + 1, n + 2, ..., N\}, j = R \\ \frac{(i-1)\mu}{r(i)+i\mu+\lambda}, & i = n + 1, j = i - 1 \\ 0, & \text{otherwise} \end{cases}$$

Let us define D

$$D = (I - Q)^{-1}R.$$

[1] We have countable state space with unbounded transition rates. However, unlike [18, Chapter-5], we do not need uniformization in our problem, since we do not have a cost that is accrued over time.

ISBN 978-3-903176-37-9 © 2021 IFIP

Note that D is a $N-n \times 3$ matrix with elements $D_{i,j},\ i \in \{n+1, n+2, .., N\},\ j \in \{R, E, n\}$. From [20, Chapter 11], it follows that, for the truncated CTMC $X^N(t)$, $\mathbb{P}_{req} = D_{n+1,R}$ and $\mathbb{P}_c = D_{n+1,n}$. Therefore, we approximate $V_{\pi^n}(n+1)$ as

$$V_{\pi^n}(n+1) = D_{n+1,R}(c+d) + D_{n+1,n}c.$$

D. Optimal Caching Policy of Homogeneous Content Caching Problem

In Section III-C, we computed n^* for a single content problem. In our original problem (see Section II-B), we want to take optimal decision for every content. In the next section we derive optimal caching algorithm for heterogeneous contents. An optimal algorithm for homogeneous case can be deduced from Algorithm 1 by setting $n_f^* = n^*, \forall f$.

IV. Heterogeneous Contents

We now allow the contents to have different communication costs, content delay costs and request rates. This is justifiable since the caching cost and delay cost of a content depends on several factors like content size, location of the server, memory device for caching etc. Also, in reality, the request rates differ depending on the type of contents.

Recall that the cost incurred on the i^{th} content depends on the other contents only through the evolution of number of other contents until content i is either precached, requested or exited. Therefore, the optimal caching decision for the i^{th} content is independent of $c_j,\ d_j$, and $r_j(n),\ \forall j \neq i$. In particular, the analysis in Section III-A for single content applies to this scenario as well, with $c,\ d$ and $r(n)$ replaced with $c_i,\ d_i$ and $r_i(n)$, respectively. Let us define cost-to-go function under policy $\pi^{\bar{n}}$ for any content i, $V_{\pi^{\bar{n}}}^{(i)}(n)$, as follows.

$$V_{\pi^{\bar{n}}}^{(i)}(n) = \begin{cases} c_i, & n \leq \bar{n} \\ \frac{r_i(n)}{r_i(n)+n\mu+\lambda}\,(c_i+d_i) & \\ +\frac{(n-1)\mu}{r_i(n)+n\mu+\lambda} V_{\pi^{\bar{n}}}^{(i)}(n-1) & \\ +\frac{\lambda}{r_i(n)+n\mu+\lambda} V_{\pi^{\bar{n}}}^{(i)}(n+1), & n > \bar{n}. \end{cases}$$

The optimal policy for the i^{th} content $\pi^{n_i^*}$, where

$$n_i^* = \min\{n : V_{\pi^n}^{(i)}(n+1) \leq c_i\}.$$

Therefore, we use policy $\pi^{n_i^*}$ for i^{th} content to obtain minimum cost. The optimal caching algorithm for the heterogeneous contents case can now be given as follows. Recall that $N(t), t \geq 0$, is a CTMC that represents the number of contents at the server. State change of this CTMC is brought by arrivals and departures. Let us define $t_m,\ m \in \{1, 2, ..\}$, as the arrival, departure and request epochs of any content.

Optimal caching decisions for heterogeneous contents are given by Algorithm 1. Observe that $\mathcal{N}(t_m)$ and $\mathcal{B}(t_m)$ denote the set of contents at the server and the cache, respectively, at time t_m.

V. Unknown System Parameters

In the previous sections, we discussed the optimal caching policy, when the parameters are known. However, in reality, the system parameters e.g., $\lambda,\ r(n)$ and μ are unknown[2].

[2]We assume that the parameters c, d are known. This assumption is justified as the server would know content delivery costs.

Algorithm 1 Optimal Caching Policy for Heterogeneous Contents

Input: $\mathcal{B}(t_0) = \emptyset$
for $m = 1, 2, ...$ **do**
  if t_m is arrival epoch of a content f **then**
    $\mathcal{N}(t_m) \leftarrow \mathcal{N}(t_m) \cup \{f\}$
    $N(t_m) \leftarrow N(t_{m-1}) + 1$
    if $N(t_m) \leq n_f^*$ **then**
      $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_m) \cup \{f\}$
    end if
  else if t_m is departure epoch of a content f **then**
    $\mathcal{N}(t_m) \leftarrow \mathcal{N}(t_m) \setminus \{f\}$
    $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_m) \setminus \{f\}$
    $N(t_m) \leftarrow N(t_{m-1}) - 1$
    for every content $j \notin \mathcal{B}(t_m)$ **do**
      if $N(t_m) \leq n_j^*$ **then**
        $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_m) \cup \{j\}$
      end if
    end for
  else
    $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_m) \cup \{f\}$
  end if
end for

In this section, we want to develop a proactive caching algorithm that does not rely on apriori knowledge of the parameters ($\lambda, r(n)$ and μ) and still gives a time average cost close to the optimal average cost. Towards this, we use the theory of reinforcement learning. More precisely, we propose Algorithm 2 based on Monte Carlo Simulation [21, Chapter 5.1]. For clarity of exposition, we restrict to the scenario of homogeneous files. Our algorithm iteratively learns estimates of required parameters and also uses the current estimates for caching decisions at any point of time. While we do not have any theoretical bound on performance of this algorithm, simulation show that it yields time average costs that approach the optimal cost as the number of iterations increase.

Recall from Theorem 3.1 and (4) that the optimal policy π^{n^*} depends on $V_{\pi^n}(n+1), \forall n \geq 0$, which in turn depend on the parameters λ, μ and $r(n),\ n \geq 1$. So we have following two approaches of learning π^{n^*}.

1) Estimate parameters λ, μ and $r(n)$ and compute $V_{\pi^n}(n+1),\ \forall n \geq 0$ and then n^* from these.
2) Directly estimate $V_{\pi^n}(n+1),\ \forall n \geq 0$, and compute n^* from these.

Clearly the first approach is computationally heavy and more error-prone as it entails estimating multiple parameters and using these to compute $V_{\pi^n}(n+1),\ \forall n \geq 0$. We therefore adopt the second approach.

A. Estimating $V_{\pi^n}(n+1),\ \forall n \geq 0$

Realize that in a caching setup, the decision maker is provided only the details of arrival, requests and departure of every file as time progresses. For brevity, we define $v_n := V_{\pi^n}(n+1),\ n \geq 0$. We start with an arbitrary estimate of $v_n,\ n \geq 0$ and update this estimate iteratively. At any time, the current estimates of $v_n,\ n \geq 0$, give an estimate of n^* in accordance with (4). As $v_n,\ n \geq 0$, evolve with time, so does n^*. Following is the proposed reinforcement learning algorithm.

ISBN 978-3-903176-37-9 © 2021 IFIP

Let A_f be the number of contents in the system on arrival of a content f(not counting f). On a arrival of a new content f, we can decide

1) to use policy π^{A_f}, not π^{n^*} for content and to use it to improve our estimate of v_{A_f}. We refer to it as *exploration*.
2) to use policy π^{n^*} for content f, where n^* is obtained based on the current estimates of $v_n,\ n \geq 0$. We refer to it as *exploitation*.

As expected, exploration helps to quickly arrive close to an optimal policy but also incurs substantial cost in the process. On the other hand, exploiting policies based on current estimates of n^* may save on instantaneous costs but do not let us approach the optimal policy. To balance between exploration and exploitation costs we use widely adopted ϵ-greedy approach, wherein for each content we perform exploration with probability ϵ and exploitation with probability $1-\epsilon$.

In practice, we may want to aggressively explore initially in order to quickly arrive at a close to optimal policy but would like to exploit more as the iterations proceed. We can formally execute this strategy as follows. Recall that $t_m,\ m \geq 1$, denote successive arrival, departure and request epochs. For a content f with arrival epoch t_m, we use $\epsilon = \epsilon(m)$ to determine whether we should perform an exploration or exploitation. More precisely, we define a Bernoulli random variable e_f that takes values 1 and 0 with probabilities $\epsilon(m)$ and $1-\epsilon(m)$, respectively; 1 corresponds to exploration and 0 to exploitation. $\epsilon(m),\ m \geq 1$ constitute a decreasing sequence, and is referred to exploration sequence. We use

$$\epsilon(m) = 1 - e^{-10^{-7}m}.$$

We also maintain a count of number of times $v_n,\ n \geq 0$ has been updated; we call this vector 'Count'. Initially Count is a zero vector. Finally, we describe how we update the estimates of $v_n,\ n \geq 0$.

a) Departure of an uncached content: If the content f exits before being cached the server incurs 0 units of cost. Therefore, v_{A_f} is updated as follows

$$v_{A_f} = \frac{\text{Count}_{A_f} v_{A_f}}{\text{Count}_{A_f} + 1}.$$

Count_{A_f} is then incremented by one unit.

b) Request of an uncached content: If the content f has been requested before it is cached, the server incurs a cost of $c+d$ units in this instant. Therefore, v_{A_f} is updated as follows

$$v_{A_f} = \frac{\text{Count}_{A_f} v_{A_f} + c + d}{\text{Count}_{A_f} + 1}.$$

And then Count_{A_f} is incremented by one unit.

Precaching an uncached content under exploration: Precaching of any content f incurs a cost of c units. Therefore, v_{A_f} is updates as follows

$$v_{A_f} = \frac{\text{Count}_{A_f} v_{A_f} + c}{\text{Count}_{A_f} + 1}.$$

And then Count_{A_f} is incremented by one unit.

Recollect that contents cached at any instant t_m is given by $\mathcal{B}(t_m)$. We now formalise the reinforcement learning based algorithm in Algorithm 2.

Remark 5.1: Note that, unlike what we do in the Q-learning, we do not update $V_{\pi^m}(n+1),\ \forall n, m \in \{1,2,3,...\}$. We only update $V_{\pi^n}(n+1),\ \forall n \geq 0$, since we need only these values to arrive at n^*(see (4)).

Algorithm 2 Optimal Caching Algorithm for Unknown Parameters

Input:$\mathcal{B}(t_0) = \emptyset, n^* = 1, \text{Count}_n = 0, \forall\ n \geq 0$
for $m = 1, 2, ..., M$ **do**
  if t_m is an arrival epoch of file f **then**
    $A_f \leftarrow N(t_{m-1})$
    $N(t_m) \leftarrow N(t_{m-1}) + 1$

$$e(f) = \begin{cases} 1 & , \text{ with a probability } \epsilon(m) \\ 0 & , \text{ with a probability } 1-\epsilon(m) \end{cases}$$

  else if t_m is a departure epoch of file f **then**
    if f is not cached and $e_f = 1$ **then**

$$v_{A_f} = \frac{\text{Count}_{A_f} v_{A_f}}{\text{Count}_{A_f} + 1}$$

      $\text{Count}_{A_f} = \text{Count}_{A_f} + 1$
    end if
    $N(t_m) \leftarrow N(t_{m-1}) - 1$
  else
    if f is not cached and $e_f = 1$ **then**
      $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_{m-1}) \cup \{f\}$

$$v_{A_f} = \frac{\text{Count}_{A_f} v_{A_f} + c + d}{\text{Count}_{A_f} + 1}$$

      $\text{Count}_{A_f} = \text{Count}_{A_f} + 1$
    end if
  end if
  for $\forall f \in \mathcal{N}(t_m)$ **do**
    if $e_f = 0$ **then**
      if f is not cached and $N(t_m) \leq n^*$ **then**
        $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_{m-1}) \cup \{f\}$
      end if
    else
      if f is not cached and $N(t_m) \leq A_f$ **then**
        $\mathcal{B}(t_m) \leftarrow \mathcal{B}(t_{m-1}) \cup \{f\}$

$$v_{A_f} = \frac{\text{Count}_{A_f} v_{A_f} + c}{\text{Count}_{A_f} + 1}$$

        $\text{Count}_{A_f} = \text{Count}_{A_f} + 1$
      end if
    end if
  end for
  $n^* \leftarrow \min\{n : v_n \leq c\}$
end for

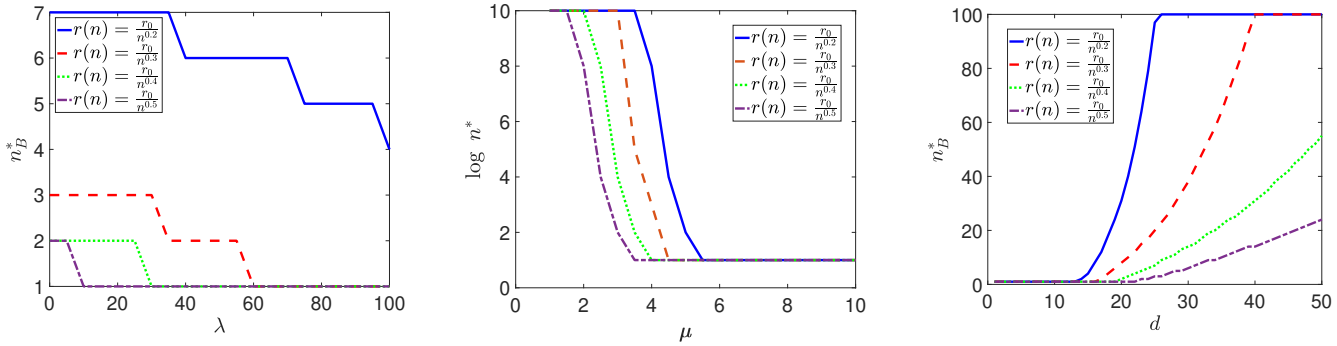
VI. Numerical results

In this section, we discuss how n_B^* varies with μ, λ and d, in the context of single content. We use the following parameters throughout the discussion: $c = 1, r(n) = r_0/n^\alpha$, for $\alpha = 0.2,\ 0.3,\ 0.4$ and 0.5. Cache size $B = 100$.

A. n_B^* vs λ, μ and d

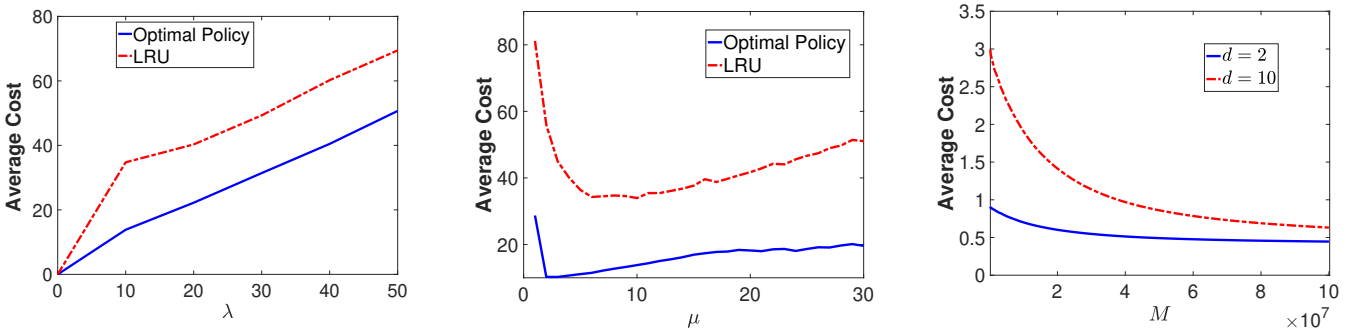
Recollect that as λ increases, the number of contents at the server increases. Recall that $r(n)$ is inversely related to

ISBN 978-3-903176-37-9 © 2021 IFIP



(a) (b) (c)

Fig. 1: n_B^* vs λ, μ and d. We simulate for $B = 100$. We use $r(n) = r_0/n^\alpha$, for $\alpha = 0.2,\ 0.3,\ 0.4,$ and 0.5. Fig 1a gives n_B^* vs λ. We use the parameters $c = 1, d = 15, \mu = 10$ and $r_0 = 1$. Fig 1b gives n_B^* vs μ. We use the parameters $c = 1, d = 1, \lambda = 20$ and $r_0 = 3.13$. Fig 1c gives n_B^* vs d. We use the parameters $c = 1, \lambda = 100, \mu = 10$ and $r_0 = 1$.



(a) (b) (c)

Fig. 2: Average cost vs λ, μ and M. Figure 2a gives Average Cost vs λ. We use $\mu = 10$. Figure 2b gives Average Cost vs μ. We use $\lambda = 10$. In both Figure 2a and Figure 2b, we use the parameters $B = 100, c = 1, d = 20, r_0 = 1$ and $r(n) = r_0/n^\alpha$, for $\alpha = 0.2$. Figure 2c gives Average cost vs M. We see average cost vs M for $d = 2$ and $d = 10$. Other system parameters are $c = 1, \lambda = 0.4, \mu = 0.01$ and $r(n) = r_0/n^\alpha$, for $r_0 = 1$. We use $\alpha = 0.2$. Under these parameters, the average cost under optimal policy corresponding to $d = 2$ and $d = 10$ are 0.3997 and 0.4, respectively.

n. Therefore, for a fixed μ, c, and d, $r(n)$ decreases as λ increases. This means that the contents are copied less aggressively. We observe the same trend in Figure 1a. We use the following parameters: $d = 15,\ \mu = 10$ and $r_0 = 1$. We observe that, for a fixed α, n_B^* decreases as λ increases. For example, at $\alpha = 0.2$, $n_B^* = 7$ for $\lambda = 10$, $n_B^* = 6$ for $\lambda = 50$, $n_B^* = 5$ for $\lambda = 80$ and $n_B^* = 4$ for $\lambda = 100$.

As μ increases, the frequency of content expiry increases. Therefore, the number of contents at the server decreases. So, n_B^* decreases as μ increases. We observe similar trend in Figure 1b. We use the following parameters $d = 1, \lambda = 20$ and $r_0 = 3.13$.

It can be noted that, as d increases, BS incurs more cost to fetch the content from the server on being requested. To avoid this, BS tends to cache more contents. Therefore, n_B^* increases with increase in d. We observe the same trend in Figure 1c. We use the following parameters, $\lambda = 100, \mu = 10$ and $r_0 = 1$, to obtain Figure 1c. We observe that n_B^* increases with d in all the cases.

B. Average Cost vs λ, μ and M

In Figure 2a, we compare the average cost incurred under the optimal policy with the average cost incurred under reactive policies, e.g., LRU. We use the parameters $B = 100, c = 1, d = 20, \mu = 10, r_0 = 1, r(n) = r_0/n^\alpha,\ \alpha = 0.2$. We observe that our policy considerably outperforms LRU. Note that in our formulation each content has same popularity at any given instant. So the average cost does not change when any other reactive caching policy is used (e.g., LFU). Therefore, we use LRU to compare our policy.

In Figure 2b, we compare the average cost incurred under the optimal policy with the average cost incurred under reactive policies, e.g., LRU. We use the parameters $B = 100, c = 1, d = 20, \lambda = 20, r_0 = 1, r(n) = r_0/n^\alpha, \alpha = 0.2$ We observe that our policy considerably outperforms LRU.

In Figure 2c, we compare the average cost from Algorithm 2 when $d = 2$ and $d = 10$. It follows that the time average cost increases as d increases. We observe the same trend in figure 2c. We use the following parameters $c = 1, \lambda = 0.4, \mu = 0.01, r_0 = 1, r(n) = r_0/n^\alpha$. We use $\alpha = 0.2$. Under these parameters, the average cost under optimal policy corresponding to $d = 2$ and $d = 10$ are 0.3997 and 0.4,

ISBN 978-3-903176-37-9 © 2021 IFIP

respectively.

VII. Conclusion

We studied a caching problem for dynamic contents with varying popularity. We first studied the problem where all the contents at the server have same communication cost and delay cost associated with them. We considered both finite and infinite cache sizes. In either case, the optimal policy is a threshold policy. In Theorem 3.1, we established the optimality of threshold policy. We provided an algorithm, Algorithm 1, to implement the optimal policy for heterogeneous content problem. Finally, we provided a reinforcement learning based algorithm, Algorithm 2, to implement the optimal policy when the system parameters are unknown. In the future, we would like to extend this work to networks with multiple BSs, each with a set of associated users.

References

[1] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 181, p. 103005, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521000321

[2] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[3] S. Moharir and N. Karamchandani, "Content replication in large distributed caches," in *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2017, pp. 128–135.

[4] A. Chattopadhyay, B. Błaszczyszyn, and H. P. Keeler, "Gibbsian online distributed content caching strategy for cellular networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 969–981, 2017.

[5] K. Avrachenkov, X. Bai, and J. Goseling, "Optimization of caching devices with geometric constraints," *Performance evaluation*, vol. 113, pp. 68–82, 2017.

[6] J. Yang, C. Ma, B. Jiang, G. Ding, G. Zheng, and H. Wang, "Joint optimization in cached-enabled heterogeneous network for efficient industrial iot," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 831–844, 2020.

[7] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2017.

[8] J. Gao, S. Zhang, L. Zhao, and X. S. Shen, "The design of dynamic probabilistic caching with time-varying content popularity," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[9] S. Zhang and J. Liu, "Optimal probabilistic caching in heterogeneous iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3404–3414, 2020.

[10] A. Sadeghi, A. G. Marques, and G. B. Giannakis, "Distributed network caching via dynamic programming," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 4574–4578.

[11] A. Sadeghi, G. Wang, and G. B. Giannakis, "Hierarchical caching via deep reinforcement learning," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3532–3536.

[12] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini, "Analyzing the Performance of LRU Caches under Non-Stationary Traffic Patterns," *arXiv e-prints*, p. arXiv:1301.4909, Jan. 2013.

[13] H. Hui, W. Chen, and L. Wang, "Caching with finite buffer and request delay information: A markov decision process approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 8, pp. 5148–5161, 2020.

[14] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 177–186.

[15] E. Altman and N. Shimkin, "Individual equilibrium and learning in processor sharing systems," *Operations Research*, vol. 46, no. 6, pp. 776–784, 1998.

[16] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *2015 IEEE international conference on communications (ICC)*. IEEE, 2015, pp. 3358–3363.

[17] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini, "Analyzing the performance of LRU caches under non-stationary traffic patterns," *CoRR*, vol. abs/1301.4909, 2013. [Online]. Available: http://arxiv.org/abs/1301.4909

[18] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 1st ed. Athena Scientific, 1995.

[19] A. Krishna. Caching dynamic contents with varying popularity. [Online]. Available: https://tinyurl.com/hcufvh6h

[20] C. M. Grinstead and J. L. Snell, *Introduction to Probability*. American Mathematical Society, 1997.

[21] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, 1st ed. Athena Scientific, 1996.

ISBN 978-3-903176-37-9 © 2021 IFIP