Reinforcement Learning for Proactive Caching of Contents with Different Demand Probabilities

Samuel O. Somuyiwa, Deniz Gündüz and András György
Department of Electrical and Electronic Engineering,
Imperial College London, UK

***Abstract*—A mobile user randomly accessing a dynamic content library over a wireless channel is considered. At each time instant, a random number of contents are added to the library and each content remains relevant to the user for a random period of time. Contents are classified into finitely many classes such that whenever the user accesses the system, he requests each content randomly with a class-specific demand probability. Contents are downloaded to the user equipment (UE) through a wireless link whose quality also varies randomly with time. The UE has a cache memory of finite capacity, which can be used to proactively store contents before they are requested by the user. Any time contents are downloaded, the system incurs a cost (energy, bandwidth, etc.) that depends on the channel state at the time of download, and scales linearly with the number of contents downloaded. Our goal is to minimize the expected long-term average cost. The problem is modeled as a Markov decision process, and the optimal policy is shown to exhibit a threshold structure; however, since finding the optimal policy is computationally infeasible, parametric approximations to the optimal policy are considered, whose parameters are optimized using the *policy gradient* method. Numerical simulations show that the performance gain of the resulting scheme over traditional reactive content delivery is significant, and increases with the cache capacity. Comparisons with two performance lower bounds, one computed based on infinite cache capacity and another based on non-casual knowledge of the user access times and content requests, demonstrate that our scheme can perform close to the theoretical optimum.**

I. Introduction

In traditional wireless networks, content delivery takes place only after a user requests a content. This method, which is referred to as *reactive* content delivery, is very sensitive to the actual quality of the channel at the time of request. On the other hand, *proactive* content delivery, in which contents are delivered in advance of a user request by leveraging the available storage at the user equipment (UE), can be used to improve the performance, particularly in wireless networks in which the network utilization (and hence the quality of the available wireless channel) is highly varying across time and space [1], [2]. However, to achieve performance gain with proactive content delivery, cache decision, that is, what to cache and when, must be made intelligently. Reinforcement learning [3] has been considered as a viable technique of making intelligent cache decisions at the edge of the wireless network to reduce service delay [4], to reduce congestion in back-haul links [5]–[7], to improve throughput [8], and to improve cache hit-rate (or reduce cache miss-rate) [9], [10]. However, most existing works do not consider dynamic generation and lifetime of contents; particularly in an online social network (OSN) context, where contents are continuously generated, but remain popular for only a limited time duration, e.g., average lifespan of a content on Twitter is 18 minutes [11], [12].

In this paper, we consider a mobile user randomly requesting contents from a library, which are generated by the user's social connections in an OSN, through a wireless link. The OSN consists of a remote server in which the content library is located, and a mobile application (app) that runs on the UE through which the user accesses the library. The UE is equipped with a cache memory of finite capacity, so that contents in the library can be replicated from the server into the UE in advance of user requests. At every time period, a random number of contents are generated and added to the library, and each content remains in the library for a random period of time, called the *lifetime*, after which it is considered to have expired and irrelevant to the user in the sense that it will no longer be requested. Without loss of generality, we assume that the user is interested in consuming a content only once. Therefore, any content that is consumed by the user becomes irrelevant afterwards, and is removed from the library. The relevance of contents to the user may be different, that is, the user may be interested in some contents more than others. Hence, contents are classified based on their relevance, and we assume that whenever the user accesses the OSN, each content is consumed with a class-specific request probability. In the extreme case, each content may have a different request probability, but we expect that in most practical situations, contents can be grouped into a few relevance categories.

The state of the wireless link through which contents are transmitted to the UE varies with time as it depends on several factors, such as, user location, channel conditions, network load, coverage, bandwidth, etc. Every time contents are transmitted to the UE, the serving base station (BS) incurs a cost that depends on the number of bits downloaded and the state of the wireless link at that time period. We consider the energy cost of transmission in this paper, but the cost can also be the receiver's energy cost, channel bandwidth, or other resources required for content delivery. Without loss of generality, we assume homogeneity in the size of contents

This work received support from the European Research Council (ERC) through the Starting Grant BEACON (grant agreement no. 725731) and the Petroleum Technology Development Fund (PTDF).

(larger contents can be split into smaller chunks of equal size). Hence, each content transmitted to the UE in the same time period incurs the same energy cost. A content manager (CM) "refreshes" the cache by downloading contents into the cache, prior to a user request, and removing contents from the cache as necessary. If the user requests a content that is already in the cache, the CM delivers it from the cache to the mobile app without any additional cost. However, if the requested content is not in the cache, the CM downloads it from the remote server at a transmission energy cost that depends on the current channel state, and delivers it directly to the user through the mobile app. The goal of the CM is to minimize the average energy cost of content delivery in the long term.

Previously, we studied a similar problem with a single content class in [13], where the request probability of all the contents were assumed to be 1; that is, the user downloads and consumes all the relevant contents whenever it accesses the mobile app. In [13] we formulated this problem as a Markov decision process with side information (MDP-SI), a modification of standard Markov decision process (MDP) [14] that treats the uncontrollable part of the state space as side information, which allowed us to easily handle the continuous state space for the channel, and show that the optimal cache management policy has a threshold structure. In this paper, we extend this result to the case of multiple content classes and show that the optimal policy still has a threshold structure in this more complicated scenario. However, due to the large number of threshold values that need to be computed because of the dependence of the threshold values on the prohibitively large state space of the system, computing and applying the optimal policy is computationally infeasible. Therefore, similarly to [13], we introduce parametric policy representations with reduced policy spaces to approximate the optimal policy, and employ reinforcement learning (more specifically, the policy gradient method) to optimize the parameters. Finally, we compare the performance of the resulting policy with two lower bounds and the traditional reactive content delivery scheme. The first lower bound, called LB-UC, is computed as the performance of an optimal policy that is allowed to use an infinite cache, while the second lower bound, called LB-NCK, is obtained as the performance of an optimal policy that has non-causal knowledge of the contents that are requested by the user and the user access times. It is shown that LB-NCK provides a tight bound for smaller cache capacities, whereas LB-UC becomes tighter as the cache capacity increases. The experimental results demonstrate that our method can provide near-optimal performance while keeping the computational costs at an affordable level.

II. System Model

We consider a slotted time system model. At the beginning of slot t, a set $\mathcal{N}_t$ consisting of a random number of new contents, each having a random lifetime, is generated and added to a library located in a remote server. The number of generated contents is denoted by $M_t \in \{0,\dots,M_{max}\}$, and the lifetime of the mth content in $\mathcal{N}_t$ is denoted by $K_{t,m} \in \{1,\dots,K_{max}\}$. That is, a content generated with lifetime $K_{t,m}$ expires by the end of slot $t+K_{t,m}-1$, and it is automatically removed from the library. All the contents are assumed to be of equal size, and the UE is equipped with a cache memory of capacity B; that is, B files can be proactively stored in the cache memory. We denote the set of contents in the user's cache at the beginning of slot t by $\mathcal{I}_t$, where $|\mathcal{I}_t| \leq B$. We denote the set of contents that are in the library but not in the cache at the beginning of slot t by $\mathcal{O}_t$. Note that $\mathcal{O}_t \supseteq \mathcal{N}_t$, that is, contents generated at the beginning of slot t are already included in $\mathcal{O}_t$.

The user randomly accesses the library in order to consume contents. We denote the user access behavior in slot t by U_t, which takes a binary value, where $U_t = 0$ implies that the user does not access the library in slot t, and thus, does not request any content, and $U_t = 1$ implies that the user accesses the library. We assume that the user access behavior is an independent and identically distributed (i.i.d) process with $\mathbb{P}(U_t = 1) = p_a$. Each content in the library belongs to one of Q classes. We assume that if the user accesses the library, a content belonging to class q will be requested with probability p_q, independently for each content in the library.

We denote the set of contents requested by the user in slot t by $R_t = (R_t^{(1)}, R_t^{(2)})$, where $R_t^{(1)} \subseteq \mathcal{O}_t$ and $R_t^{(2)} \subseteq \mathcal{I}_t$. Note that $R_t = \emptyset$ when $U_t = 0$. In slot t, when $U_t = 1$, the CM downloads all the requested contents that are not in the cache, $(R_t^{(1)})$, from the server, and moves them together with the requested contents that are already in the cache, $(R_t^{(2)})$, to the application layer. In any slot t, the CM may download additional contents that have not been requested, to be stored in the cache, and may also discard already prefetched contents from the cache if necessary. We denote the set of all contents downloaded to the UE in slot t by $A_t^{(1)} \subseteq \mathcal{O}_t$, and the set of all contents discarded from the cache in slot t by $A_t^{(2)} \subseteq \mathcal{I}_t$.

Since all the contents are of equal size, we can represent each content by its remaining lifetime L and the class q it belongs to, which will be denoted by $L^{[q]}$. Any arithmetic operation involving $L^{[q]}$ will be applied to L, e.g., $L^{[q]} - 1 = (L-1)^{[q]}$. Given this representation, all the sets of contents introduced so far, $\mathcal{N}_t$, $\mathcal{O}_t$, $\mathcal{I}_t$, $R_t^{(1)}$, $R_t^{(2)}$, $A_t^{(1)}$ and $A_t^{(2)}$, can be represented as multisets of pairs consisting of remaining lifetimes and classes (with the set of all tuples of pairs denoted by $\mathbb{N}^*$). For simplicity, we will refer to multisets of pairs simply as multisets. For a multiset $\mathcal{Y}$, we let $\mathcal{Y} - 1 = \{y^{[j]} > 0 : y^{[j]} + 1 \in \mathcal{Y}\}$ denote the multiset obtained by reducing the lifetime of each element in $\mathcal{Y}$ by 1, and removing the elements whose lifetimes become 0. Given these definitions, the system evolves according to the following equations:

$$\begin{aligned}\mathcal{O}_{t+1} &= \left(\left(\left(\mathcal{O}_t \setminus A_t^{(1)}\right) \cup \left(A_t^{(2)} \setminus R_t^{(2)}\right)\right) - 1\right) \cup \mathcal{N}_{t+1},\\ \mathcal{I}_{t+1} &= \left(\left(\mathcal{I}_t \setminus A_t^{(2)}\right) \cup \left(A_t^{(1)} \setminus R_t^{(1)}\right)\right) - 1.\end{aligned} \tag{1}$$

We denote the wireless link quality, which is the instantaneous cost of downloading a content to the UE in slot t by $C_t \in \mathbb{R}^+$, which is an independent realization of a continuous random variable C with cumulative distribution function (cdf) $F_C(c)$, and bounded by C_{max}. Given these, the total instantaneous cost incurred in slot t is $|A_t^{(1)}| \cdot C_t$, and the average cost incurred after T slots is $J_T = \frac{1}{T}\sum_{t=1}^{T} |A_t^{(1)}| \cdot C_t$. The goal of the CM is to minimize the long-term expected average cost defined as

$$\rho \triangleq \limsup_{T\to\infty} \mathbb{E}\left[J_T\right] = \limsup_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} |A_t^{(1)}| \cdot C_t\right]. \quad (2)$$

We assume that the sequences $\{M_t\}$, $\{K_{t,m}\}$, $\{U_t\}$, $\{R_t\}$ and $\{C_t\}$ are independent of each other. In the next section, we first show that this problem can be modeled as a MDP-SI [13], and then derive the structure of the optimal policy.

III. Optimal Policy of the Cache Manager

A. The MDP-SI Model

In a standard discrete-time MDP with finite-state and action spaces, there exists a state space $\mathcal{S}$, an action space $\mathcal{A}$, a probability kernel $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, and a cost function $\mu : \mathcal{S} \times \mathcal{A} \to [0, \mu_{max} > 0] \cup \{\infty\}$, such that, in order to avoid an infinite cost, a different action set is allowed in every state. That is, for every state $s \in \mathcal{S}$, the set $\mathcal{A}_s = \{a \in \mathcal{A} : \mu(s,a) < \infty\}$ denotes the set of feasible actions, and we assume that $\mathcal{A}_s \neq \emptyset$, for all $s \in \mathcal{S}$. Thus, a standard MDP is characterized by the tuple $(\mathcal{S}, \mathcal{A}, P, \mu)$.

With the objective of minimizing the infinite-horizon average cost $\rho = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mu(S_t, A_t) \middle| S_1\right]$, a Markov chain controller takes action $a \in \mathcal{A}_s$ at every slot t, and the system transitions from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ with probability $P(s'|s,a) \triangleq \mathbb{P}\left[S_{t+1} = s' \middle| S_t = s, A_t = a\right]$, where $\sum_{s'\in\mathcal{S}} P(s'|s,a) = 1$, for all $s \in \mathcal{S}, a \in \mathcal{A}$. The actions of the controller are governed by a deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$. Let Π denote the set of all deterministic policies.

In a discrete-time MDP-SI, the state space is separated into a controllable and an uncontrollable part. Thus, $\mathcal{S}$ is considered a controllable state space, and there is an additional state space $\mathcal{Z} \subset \mathbb{R}$, where $Z_t \in \mathcal{Z}$ is an i.i.d sequence with cdf F_Z, which is uncontrollable but affects the cost function $\mu_{SI} : \mathcal{S} \times \mathcal{A}_{SI} \times \mathcal{Z} \to [0, \mu_{max}] \cup \{\infty\}$, where $\mathcal{S}$ and $\mathcal{A}_{SI}$ are countable state and action spaces, while $\mathcal{Z}$ may not be a finite state space. If the controller takes action $a \in \mathcal{A}_{SI}$ in states $s \in \mathcal{S}$ and $z \in \mathcal{Z}$ at any slot, the system transitions to states $s' \in \mathcal{S}$ and $z' \in \mathcal{Z}$ with probability $P_{SI}(s', z'|s, a, z)$. Since Z_t is not controlled by the strategies of the controller, it is regarded as a side information, and the MDP-SI can be characterized by $(\mathcal{S}, \mathcal{A}_{SI}, P_{SI}, \mu_{SI}, \mathcal{Z}, F_Z)$. The actions of the controller in MDP-SI is governed by a deterministic policy $\pi_{SI} : \mathcal{S} \times \mathcal{Z} \to \mathcal{A}_{SI}$. It is shown in [13] that the policy π_{SI} of the MDP-SI can be turned into policy $\pi \in \Pi$ of the MDP, and vice versa, and that the expected average cost of the two models are the same for the corresponding policies. Therefore, it is enough to consider the MDP $(\mathcal{S}, \mathcal{A}, P, \mu)$, where the controllable and uncontrollable state spaces are not separated.

B. The Optimal Policy of the MDP-SI

Considering a finite MDP $(\mathcal{S}, \mathcal{A}, P, \mu)$, let ρ^π denote the infinite-horizon average cost ρ when $A_t = \pi(S_t)$, that is,

$$\rho^\pi = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mu(S_t, \pi(S_t)) \middle| S_1\right]. \quad (3)$$

The limit exists as $\mu(s,a) > 0$ and the initial state S_1 does not matter assuming that the policy $\pi \in \Pi$ of the MDP $(\mathcal{S}, \mathcal{A}, P, \mu)$ is defined by an irreducible and aperiodic transition kernel $P^\pi : \mathcal{S} \times \mathcal{S} \to [0,1]$. There exists a deterministic policy π^* [14] that minimizes ρ^π over all, possibly non-stationary and non-deterministic causal control policies. The differential value function for any state $s \in \mathcal{S}$ is defined as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} (\mu(S_t, \pi(S_t)) - \rho^\pi) \middle| S_1 = s\right]. \quad (4)$$

Furthermore, the optimal policy π^* that minimizes ρ^π in (3) satisfies

$$V^{\pi^*}(s) = \min_{a\in\mathcal{A}} \left\{\mu(s,a) - \rho^{\pi^*} + \sum_{s'\in\mathcal{S}} P(s'|s,a) V^{\pi^*}(s')\right\}, \quad (5)$$

and $a = \pi^*(s)$ minimizes the right hand side.

We show in [13] that an optimal policy $\pi^*_{SI}(s,\cdot) = \pi^*$ of the MDP-SI exists even if the uncontrollable state space $\mathcal{Z}$ is not finite but is an interval.

C. Behavior of the Optimal Policy of the Cache Manager

The cache management problem is indeed an MDP-SI: The state $s \in \mathcal{S}$ of the system in slot t is $S_t = (\mathcal{O}_t, \mathcal{I}_t)$, that is, the state space $\mathcal{S} \subset \mathbb{N}^* \times \mathbb{N}^*$. The states $\mathcal{O}_t$ and $\mathcal{I}_t$ transition according to (1). The i.i.d. side information $z \in \mathcal{Z}$ in slot t is the channel state C_t and user requests R_t, which are governed by the cdf F_C and by the probabilities p_a and p_q, $q = 1, 2, \ldots, Q$, respectively. The action of the CM in slot t is $A_t = (A_t^{(1)}, A_t^{(2)})$. The decision of the CM depends on Z_t and S_t, as the cost of taking action A_t is $\mu(S_t, A_t, Z_t) = |A_t^{(1)}| \cdot C_t$.

Note that, the user request (R_t) in slot t limits the set of possible actions as the requested contents must be delivered to the user in that slot. Hence, we only need to characterize the optimal policy for the relevant contents that have not been requested by the user. We can think of downloading these contents in the *intermediate state* $(\mathcal{O}'_t, \mathcal{I}'_t)$ obtained after satisfying the user request, that is, $\mathcal{O}'_t = \mathcal{O}_t \setminus R_t^{(1)}$ and $\mathcal{I}'_t = \mathcal{I}_t \setminus R_t^{(2)}$, and given an *intermediate side information*, which excludes the user requests, that is, it is the channel state C_t only. Therefore, the decision $A_t \in \mathcal{A}_s$ of the optimal policy of the CM in slot t is a piecewise constant function of the channel state C_t [13].

We denote the decision of the CM to download content $L^{[j]} \in \mathcal{O}$ into the cache while removing content $l^{[i]} \in \mathcal{I}$ by $a^d = (l^{[i]}|L^{[j]})$, and call this decision a *simple action*. Recall

that L and l are the remaining lifetimes of the contents while i and j are their classes, respectively. For the simple action a^d, $l = 0$ implies that a content with remaining lifetime $L > 0$ is downloaded into an empty space in the cache memory. Since the number of contents that can be downloaded into the cache is constrained by the cache capacity, no more than B simple actions can be taken in any slot. Let a_n^d denote the nth simple action. The proactive caching action of the CM can be represented as a sequence of $b \leq B$ simple actions as $A_t \setminus R_t = \{a_1^d, a_2^d, \cdots a_b^d\}$. This representation will help us characterize the optimal policy.

First, we denote the *value* of a content $L^{[q]}$, for any remaining lifetime L and any class q, as $\mathcal{V}_{L^{[q]}}$, such that, the optimal policy which satisfies (5) places contents with higher values in the cache. It is easy to see that the optimal policy may take a simple action $a^d = (l^{[i]}|L^{[j]})$ only for $L > l$ if $i = j$, and for $L = l$ for which $p_j > p_i$. The value of a content is the expected future cost associated with the content given its remaining lifetime and class, when the optimal policy is followed. We can estimate $\mathcal{V}_{L^{[q]}}$ for all L and q if we assume that $L^{[q]}$ is the only content in $\mathcal{O}$ and there is space in the cache memory to proactively cache $L^{[q]}$. Given this assumption, the problem can be modeled as a finite-horizon MDP, where the horizon equals the remaining lifetime L. Thus, we can apply dynamic programming to determine the optimal values using backward induction, such that, $\mathcal{V}_{1^{[q]}} = 0$ for any q since a content with remaining lifetime 1 expires at the end of a slot, so there is no reason to proactively cache it. Assuming optimal decisions are made for remaining lifetimes up to $L-1$, then the value of content $L^{[q]}$ can be obtained as

$$\mathcal{V}_{L^{[q]}} = p_a p_q \cdot \mathbb{E}\left[C\right] + (1 - p_a p_q)\mathbb{E}\left[\min\{C, \mathcal{V}_{L^{[q]}-1}\}\right]. \quad (6)$$

The behavior of the optimal policy is described in the following theorem.

Theorem 1. *Consider an intermediate state $s' = (\mathcal{O}', \mathcal{I}') \in \mathcal{S}$ in a time slot with channel cost C. Let $f_1, \ldots, f_B$ denote the ordered contents (i.e., lifetime-class pairs) in the cache satisfying $\mathcal{V}_{f_1} \leq \cdots \leq \mathcal{V}_{f_B}$, and let $F_1, \ldots, F_B$ denote the contents not in the cache with the B largest values, that is, $\mathcal{V}_{F_1} \geq \cdots \geq \mathcal{V}_{F_B}$. Then there exists $b \leq B$ and corresponding threshold values $\mathcal{T}(a_b^d) \leq \mathcal{T}(a_{b-1}^d) \leq \cdots \leq \mathcal{T}(a_1^d) \leq C_{max}$, depending on s', such that there is an optimal cache management policy that performs simple actions $a_n^d = (f_n|F_n)$ for all n for which $C \leq \mathcal{T}(a_n^d)$.*

The proof follows similarly to the one in [13] given for a single class of files, and is omitted here due to space constraints.

The threshold values of the optimal policy depends generally on the remaining lifetimes of all the relevant contents, i.e., the optimal threshold value of each simple action depends on the state. Therefore, the number of optimal threshold values scales with the size of the state space $\mathcal{S}$, which is extremely large. Hence, computing and even storing the optimal threshold values is infeasible. In the next section, we describe two different parametric approximations to the optimal policy with reduced policy spaces.

IV. Policy Approximation

We consider two different parametrized suboptimal policies. The first policy representation takes a simple action $a^d = (l^{[i]}|L^{[j]})$ corresponding to the content with the lowest value, $\mathcal{V}_{l^{[i]}}$, inside the cache, and the content with the highest estimated value, $\mathcal{V}_{L^{[j]}}$, outside the cache. This is called the value-based policy (VBP), and it has a single threshold for each simple action irrespective of other available simple actions. The second policy representation employs, for each simple action, a threshold value that is a linear function of the state. This is called the linear function approximation (LFA) policy.

To overcome the complexity associated with the dependence of the optimal policy on the state space, we consider a parameterized policy $\pi_{\boldsymbol{\theta}}$, where the parameter $\boldsymbol{\theta}$ operates over a reduced parameter space $\boldsymbol{\Theta}$. Then, we use a policy gradient reinforcement learning method called likelihood-ratio method (LRM) [15] to optimize the parameters of these two policy representations.

A. Value-Based Policy (VBP)

VBP has a simple structure, such that there is a single threshold value for each simple action, independently of the intermediate state. The policy replaces the content with the lowest estimated value inside the cache with the content with the highest estimated value outside the cache, if the channel cost is below the corresponding threshold. VBP is parameterized by its threshold values, that is, $\mathcal{T}((l^{[i]}|L^{[j]})) = \theta(l^{[i]}, L^{[j]})$, where $\theta(l^{[i]}, L^{[j]}) \in [0, C_{\max}]$ for all $l, L \in \{0, \ldots, K_{\max}\}$ and $i, j \in \{1, \ldots, Q\}$. This representation results in a parameter set of dimension $(QK_{\max}+1)^2$, which can be further reduced by ignoring all $\theta(l^{[i]}, L^{[j]})$ for which $\mathcal{V}_{l^{[i]}} \geq \mathcal{V}_{L^{[j]}}$, that is, the policy does not replace a content in the cache with a content outside the cache with a lower expected value. We also ensure the following natural monotonicity rules that for any contents $f_1, f_2 \in \mathcal{I}$ and $F_1, F_2 \in \mathcal{O}$, $\mathcal{T}((f_1|F_1)) \leq \mathcal{T}((f_2|F_2))$ if either $\mathcal{V}_{F_1} \leq \mathcal{V}_{F_2}$ or $\mathcal{V}_{f_1} \geq \mathcal{V}_{f_2}$.

B. LFA Policy

We represent the cache state in every slot by the numbers of "similar" contents, that is, contents having the same remaining lifetime and belonging to the same class that are in the cache at that time slot. To do this, we introduce the vector Φ_t where, $\Phi_t(0)$ denotes the ratio of the number of empty spaces in the cache in slot t, and $\Phi_t((q-1)K_{\max}+k)$, denotes the ratio of the number of contents in the cache in slot t having lifetime k and belonging to class q, for $k = 1, \cdots, K_{\max},\ q = 1, \cdots, Q$; that is, we have

$$\phi_t((q-1)K_{\max}+k) \triangleq \frac{\sum_{l^{[i]} \in \mathcal{I}} \mathbb{I}_{\{l=k\}} \cdot \mathbb{I}_{\{i=q\}}}{B},$$

where $\mathbb{I}_{\{\cdot\}}$ denotes the indicator function. The threshold value for each simple action $a^d = (l^{[i]}|L^{[j]})$ for which $\mathcal{V}_{l^{[i]}} \leq \mathcal{V}_{L^{[j]}}$,

$l, L \in \{0, \ldots, K_{\max}\},\ i, j \in \{1, \ldots, Q\}$ is a linear function of the vector Φ_t representing the cache state as

$$\begin{aligned}\mathcal{T}(a^d) &= \Phi_t(0)\theta_0(l^{[i]}, L^{[j]}) \\ &\quad + \sum_{q=1}^{Q}\sum_{k=1}^{K_{\max}} \Phi_t((q-1)K_{\max}+k)\theta_{(q-1)K_{\max}+k}(l^{[i]}, L^{[j]}) \\ &= \Phi_t^\top \theta(l^{[i]}, L^{[j]}),\end{aligned}$$

where $\theta_{(q-1)K_{\max}+k}(l^{[i]}, L^{[j]}) \in \mathbb{R}$ for $k = 1, \cdots, K_{max}$ and $q \in \{1, \ldots, Q\}$.

To optimize the parameters of each policy considered, we use the policy gradient algorithm described in Section IV-C.

C. Likelihood-ratio method (LRM)

We employ policy gradient, a model-free reinforcement learning method, to optimize the parameters of the specified policies. Policy gradient uses stochastic gradient descent, following sampled trajectories obtained using a parameterized policy $\pi_{\boldsymbol{\theta}}$, to minimize the expected average cost $\rho^{\pi_{\boldsymbol{\theta}}}$. The estimated gradient $\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}}$ of $\pi_{\boldsymbol{\theta}}$ is used to update $\boldsymbol{\theta}_j$ at the end of step j as

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \lambda \nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}_j}}, \tag{7}$$

where λ is some positive step size. Given an infinite trajectory $\tau = (S_1, A_1), (S_2, A_2), \ldots$ obtained using the distribution $P_{\boldsymbol{\theta}}$ by following policy $\pi_{\boldsymbol{\theta}}$, we let $J(\tau) = \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mu(S_t, A_t)$. Then the gradient estimate can be obtained as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} &= \int \nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\tau) J_{\pi_{\boldsymbol{\theta}}}(\tau) d\tau, \\ &= \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\tau) J(\tau)],\end{aligned}$$

where the expectation with respect to $P_{\boldsymbol{\theta}}$ can be approximated over sampled trajectories of finite length. Since $P_{\boldsymbol{\theta}}(\tau) = P(S_1)\prod_{t=1}^{T} P(S_{t+1}|S_t, A_t)\pi_{\boldsymbol{\theta}}(A_t|S_t)$, the gradient estimate can be taken without the knowledge of $P_{\boldsymbol{\theta}}$ as

$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} = \mathbb{E}\left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t|S_t) J(\tau)\right].$$

The LRM algorithm takes a simple action a^d in state S with probability $\pi_{\boldsymbol{\theta}}(a^d|S) \in [0, 1]$. This allows for exploration, which is essential for learning new sampled trajectories. We define a randomized policy $\pi_{\boldsymbol{\theta}}(a^d|S)$ as a sigmoid function with negative slope parameter η:

$$\pi_{\boldsymbol{\theta}}(a^d|S) = \frac{1}{1 + e^{-\eta(\mathcal{T}(a^d) - C)}}\ .$$

For $b \leq B$ simple actions taken with respect to the structure of the optimal policy in Theorem 1, and noting that any requested content is always delivered –with probability 1, we can express the policy taking action A in state S as

$$\pi_{\boldsymbol{\theta}}(A|S) = \prod_{n=1}^{b} \pi_{\boldsymbol{\theta}}(a_n^d|S).$$

In the next section we present performance results of the reduced complexity schemes optimized with LRM.

V. Numerical Results

We compare the performances of the proposed proactive caching schemes with reactive content delivery and two lower bounds described below.

A. Lower Bounds

The first lower bound, called LB-UC, is obtained by considering an unlimited cache capacity. This implies that each content can be downloaded into the cache individually and independently of other contents inside and outside the cache. An optimal policy of the CM will download each content $L^{[j]}$ with respect to its value $\mathcal{V}_{L^{[j]}}$, that is, its expected future cost, which can be obtained using (6). The optimal policy of the CM downloads a content to the cache if the instantaneous cost of downloading it at that time is less than or equal to its value, otherwise it does not download the content.

The second lower bound, called LB-NCK, considers that the user access times and requested contents are known non-causally. This implies that contents that will never be requested by the user can be removed from the system. The remaining system can be solved through dynamic programming for any finite time horizon that spans several user access instants. The state of the system can be characterized by the number of contents that are not in the cache and the number of empty spaces in the cache. An optimal policy of the CM takes into consideration the time to user request of each content that is not in the cache, and finds the number of contents to download into empty spaces in the cache.

B. Simulation Setup

For simulations, we consider $Q = 2$ classes of contents, with request probabilities $p_1 = 0.9$ for class $q = 1$ and $p_2 = 0.3$ for class $q = 2$. Each newly generated content in $\mathcal{N}_t$ is allocated to either of the class uniformly at random. User access probability $p_a = 0.15$, M_t is drawn uniformly at random from the set $\{1, 2, \ldots 8\}$, and the lifetime $K_{t,m}$ of individual content $m \in \{1, 2, \ldots M_t\}$ is drawn uniformly at random from the set $\{5, 10, 15\}$.

The channel cost C_t is obtained using Shannon's capacity formula, $R = W \log_2(1 + P_{signal}/P_{noise})$, where R is a deterministic transmission rate, W is the channel bandwidth, P_{noise} is the noise power, and P_{signal} is the signal power. We use parameters consistent with the LTE network model and 3GPP channel model [16].

For the randomized policy of LRM, we set $\eta = 10$. A policy update is performed after 20 trajectories, with the duration of a trajectory set to 200 slots. In each simulation setup, we adjust the step size at different runs, and select the one that gives the best result. Finally, to test the performance of any policy, we use a *test data* of 100 trajectories, each consisting of 5000 time slots.

In Figure 1 we plot the average energy cost with respect to the cache capacity. As expected, the performances of the reactive scheme and LB-UC do not change with cache capacity since the reactive scheme does not utilize the cache and LB-UC considers an unlimited cache capacity. The performances

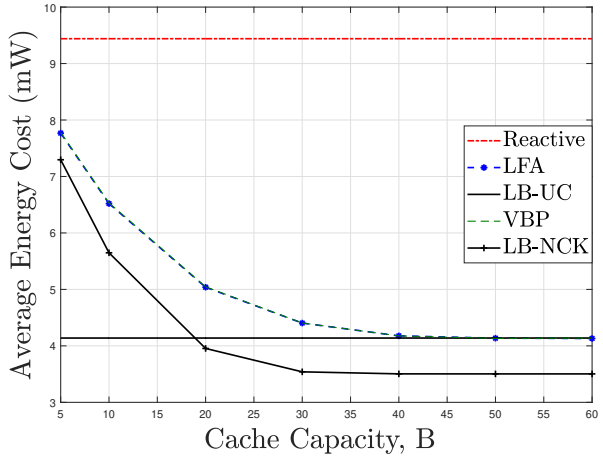

Fig. 1: Average energy cost vs. cache capacity.

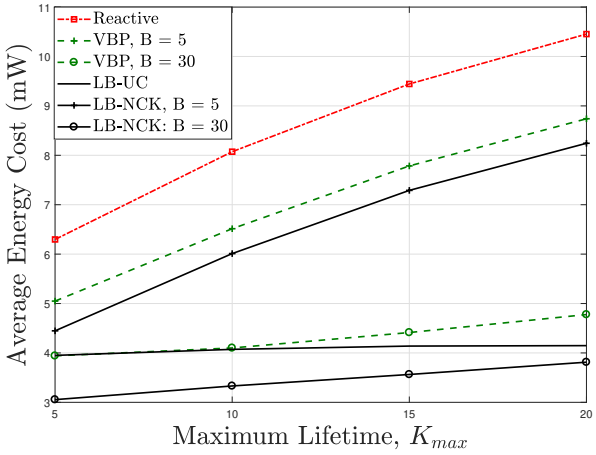

Fig. 2: Average energy cost vs. maximum lifetime of contents for $B = 5$ and $B = 30$.

of the proposed schemes, VBP and LFA, improve with increasing cache capacity before saturating at around $B = 40$, which is approximately the expected size of the library. VBP and LFA outperform the reactive scheme by approximately 17% at $B = 5$, where they perform very close to LB-NCK, and by approximately 56% at $B = 40$, where they perform very close to LB-UC. The two schemes perform close to each other for the system settings considered. In Figure 2, we plot the average energy cost with respect to the maximum lifetime of contents. The average energy cost increases with the increase in the maximum lifetime of contents. This is because contents remain relevant longer, thus, more contents are consumed by the user before their expiry. The proposed scheme outperforms reactive content delivery, with a performance gain that increases with increasing K_{max}, and perform close to the two lower bounds within the cache capacity regime considered.

VI. Conclusions

We considered the problem of proactively caching contents into a limited capacity cache memory of a mobile device in order to minimize the long term average energy cost in a wireless network with a time-varying channel state. The mobile user randomly demands contents from a dynamic content library to which a random number of contents with random lifetimes are added at each time slot. We showed that a threshold policy that downloads contents into, and may remove contents from, the cache depending on the relative value of the contents compared to a threshold value, which depends on the remaining lifetimes and request probabilities of all active contents, is optimal. To address the computational complexity associated with the huge space of the optimal policy, we proposed low complexity parameterized policy representations and used policy gradient reinforcement learning to optimize the parameters. We showed that the proposed low-complexity schemes outperform the traditional reactive content delivery, and perform close to two lower bounds obtained by considering an unlimited cache capacity and a non-causal knowledge of user access times and content requests.

References

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inform. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[2] M. Gregori, J. Gomez-Vilardebo, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and D2D networks," *IEEE Journal on Sel. Areas in Comms.*, vol. 34, no. 5, pp. 1222–1234, May 2016.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[4] M. S. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Content-aware user clustering and caching in wireless small cell networks," in *Int'l Symp. on Wireless Comms. Systems (ISWCS)*, Aug 2014, pp. 945–949.

[5] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *IEEE Int'l Conf. Comms. (ICC)*, Jun. 2014, pp. 1897–1903.

[6] P. Blasco and D. Gündüz, "Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds," *abs/1407.6154*, 2014. [Online]. Available: http://arxiv.org/abs/1407.6154

[7] S. Muller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. on Wireless Comms.*, vol. 16, no. 2, Feb 2017.

[8] S. T. ul Hassan, S. Samarakoon, M. Bennis, M. Latva-aho, and C. S. Hong, "Learning-based caching in cloud-aided wireless networks," *IEEE Communications Letters*, vol. 22, no. 1, pp. 137–140, Jan 2018.

[9] C. Zhong, C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," *ArXiv*, Dec. 2017.

[10] A. Sadeghi, F. Sheikholeslami, and G. Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Sel. Topics in Signal Proc.*, vol. 12, no. 1, Feb 2018.

[11] A. Lobzhanidze, W. Zeng, P. Gentry, and A. Taylor, "Mainstream media vs. social media for trending topic prediction - an experimental study," in *IEEE Consumer Comms. and Netw. Conf.*, Jan 2013, pp. 729–732.

[12] D. Wells. (2016) The lifespan of a social media post. [Online]. Available: http://bit.ly/29Byg3Q

[13] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 7, pp. 1–14, July 2018.

[14] M. L. Puterman, *Markov Decision Processes: Discrete Time Stochastic Control*. John Wiley and Sons, 2005.

[15] M. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Found. Trends Robot*, vol. 2, pp. 1–142, Aug. 2013.

[16] T36.814 V9.0.0, "Further advancements for E-UTRA physical layer aspects (release 9)," *3GPP*, Mar. 2010.