A Reinforcement-Learning Approach to Proactive Caching in Wireless Networks

Samuel O. Somuyiwa, András György and Deniz Gündüz, *Senior Member, IEEE*

***Abstract*—We consider a mobile user accessing contents in a dynamic environment, where new contents are generated over time (by the user's contacts), and remain relevant to the user for random lifetimes. The user, equipped with a finite-capacity cache memory, randomly accesses the system, and requests all the relevant contents at the time of access. The system incurs an energy cost associated with the number of contents downloaded and the channel quality at that time. Assuming causal knowledge of the channel quality, the content profile, and the user-access behavior, we model the proactive caching problem as a Markov decision process with the goal of minimizing the long-term average energy cost. We first prove the optimality of a threshold-based proactive caching scheme, which dynamically caches or removes appropriate contents from the memory, prior to being requested by the user, depending on the channel state. The optimal threshold values depend on the system state, and hence, are computationally intractable. Therefore, we propose parametric representations for the threshold values, and use reinforcement-learning algorithms to find near-optimal parametrizations. We demonstrate through simulations that the proposed schemes significantly outperform classical reactive downloading, and perform very close to a genie-aided lower bound.**

***Index Terms*—Markov decision process, proactive content caching, policy gradient methods, reinforcement learning.**

I. Introduction

Content delivery networks (CDNs), such as Amazon Web Service (AWS) and Akamai, replicate contents from a local repository at servers that are geographically closer to users; specifically, at Internet exchange points or Internet service providers. This approach significantly improves utilization of the Internet "backbone" capacity, thereby reducing latency and improving reliability [3]. However, today a large proportion of high-rate contents, e.g., videos, are delivered to users through cellular/wireless networks, which may introduce bottlenecks. Researchers have recently proposed *proactive caching* of contents at the wireless network edge, that is, at the micro/macro base stations (BS) and/or even directly at user equipments (UEs). Proactive caching is particularly appropriate for prerecorded contents, e.g., YouTube videos or user generated contents in online social networks (OSNs), and is based on the assumption that the system knows/predicts in advance which contents are likely to be requested by the users.

Manuscript received 10 December 2017; revised 8 March 2018. Parts of this work were presented at the Int'l Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Nets. (WiOpt), and at the 2nd Content Caching and Delivery in Wireless Networks Workshop (CCDWN), Paris, France, May 2017 [1], [2]. 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).

The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, UK. Email: {samuel.somuyiwa12, a.gyorgy, d.gunduz}@imperial.ac.uk.

Content popularity is highly dynamic, and the performance of caching systems depends critically on tracking the relevance of contents. Learning theoretic tools have been applied to proactive caching at wireless access points to reduce congestion in back-haul links [4], to reduce the service delay [5], and to improve hit-rate [6]–[9]. Caching at macro and micro BSs is modeled as a stochastic optimization problem in [10], with the objective of minimizing the average transmission power and delay in an heterogeneous wireless network. Proactive caching of contents directly at user devices has been studied in [11] and [12] from an energy efficiency perspective, considering an *offline* setting, that is, the user demands and channel conditions are known in advance. Both offline and online proactive caching is considered in [13] to improve the effective throughput (hit-rate) given random user requests and a limited cache capacity. However, most of these works do not take into account the time-varying nature of content generation, particularly in the context of OSNs, where new contents are generated over time, and the popularity of each content is non-stationary. In practice, popularity typically diminishes soon after a content is generated [14]; for example, the average *lifetime*, that is, the duration a content remains popular is approximately 2 hours for a video posted on Facebook, and 18 minutes on Twitter [15]. Instead, most caching schemes in the literature make caching decisions based on a static content-popularity profile and a fixed content library, which results in performance degradation. For example, it is shown in [16] that service delay increases and cache-hit ratio decreases, when caching decisions for social media contents are done without taking the time variations in popularity into consideration. Learning time-varying popularity of contents is studied in [8], [9]. Time variations in the wireless channel quality and traffic conditions, together with variations in the lifetime and popularity of contents require intelligent content placement and cache update mechanisms that can adapt to these variations.

In this paper, we consider proactive content caching into a mobile UE in the framework of an OSN, such that new contents (messages, videos, pictures), posted by a user's connections, become available over time. Each content has a finite lifetime, which is known at the time of generation[1]. Contents are delivered via a wireless link at a transmission energy cost[2]. The cost depends on the number of contents downloaded as well as the channel and network conditions, which typically

[1]In practice, a content's lifetime depends on its popularity in a dynamic manner; however, online popularity estimation is out of the scope of this paper; hence, we assume that the lifetime of each content is known at generation.

[2]The proposed framework can be easily adapted to any other network resource, e.g., bandwidth, delay or the energy cost at the UE.

vary over time due to traffic, user mobility, pathloss, as well as large scale fading effects.

Conventional wireless networks employ *reactive* content delivery; that is, every time the user accesses the OSN through an application software (*app*), all the *relevant* contents whose lifetime has not yet expired, are downloaded to the UE. Alternatively, in *proactive caching*, contents can be downloaded to the UE by a cache manager (CM) before the user accesses the OSN to request these contents. Downloaded contents are stored in the cache, and are retrieved and delivered to the application layer, i.e., to the app, whenever the user accesses the OSN. Therefore, contents can be downloaded under more favorable channel conditions, providing energy savings. On the other hand, the CM may push contents that will not be requested by the user within their lifetimes, increasing the energy consumption. The limited cache capacity at the UE limits the amount of contents that can be proactively cached, or may require replacing already downloaded contents, increasing the cost. Hence, we aim to answer the question of which contents, and at what time, should be pushed to the cache.

We consider a slotted time model, in which a random number of relevant contents are generated with random lifetimes at each time slot. For simplicity, we assume that all the contents have equal size, which is without loss of generality if we assume that larger contents are split into smaller chunks of equal size, e.g., video segments in DASH. We model both the channel quality and the user behavior as stochastic processes. The user randomly accesses the contents in her OSN feed in order to view/consume all the relevant contents at the time of access. We will propose reinforcement-learning algorithms for the CM that can learn and adapt to an unknown environment even when the statistics governing the system are unknown.

Our specific contributions can be summarized as follows:

- We formulate the problem as an infinite-horizon average-cost Markov decision process (MDP), with the objective of minimizing the long term average energy consumption.
- To overcome the technical difficulty due to the continuous distribution of the channel quality, we introduce a new MDP model, referred to as an MDP with side information (MDP-SI). We show the optimality of a threshold-based proactive caching policy, which downloads contents into, and removes contents from the cache depending on the remaining lifetime of the contents and the relative value of the current channel state with respect to a threshold.
- Since the optimal threshold values depend on the system state, the prohibitively large size of the state space makes it practically infeasible to compute and store them. Hence, we introduce two low-complexity parametric policy representations that are able to approximate the optimal performance. The first policy, called *Longest lifetime In–Shortest lifetime Out (LISO)*, assigns a single threshold to each pair of contents, the one with the longest remaining lifetime outside the cache, and the one with the shortest remaining lifetime inside the cache, independent of the system state. The second policy, called linear function approximation (LFA), represents the threshold values for every possible pair of remaining lifetimes as a linear function of the system state.

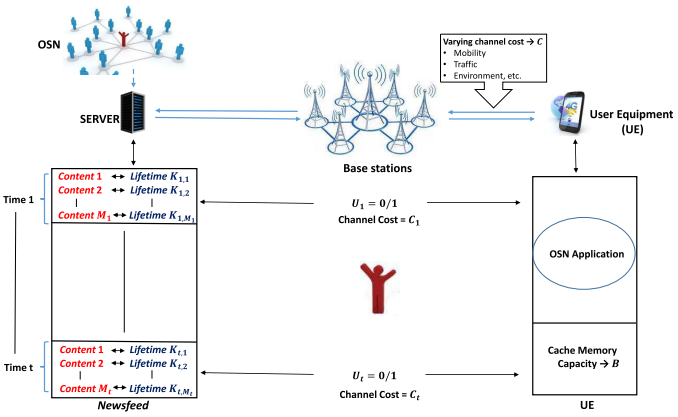


Fig. 1: Illustration of the system architecture. The OSN server has a list of relevant contents with random lifetimes. Contents can be pushed to the cache memory before being requested by the user, to take advantage of favorable channel conditions.

We use reinforcement learning techniques to optimize the threshold values for the proposed caching schemes. In particular, we apply two policy gradient schemes, the finite difference method (FDM) and the likelihood-ratio method (LRM) [17].

- We present two lower bounds on the performance: one assuming unlimited cache capacity, and another assuming non-causal knowledge of the user-access times. Through numerical simulations, we demonstrate that the proposed schemes perform close to the latter lower bound when the cache capacity is small, and to the former for larger cache capacities, and significantly outperform reactive caching. We also show that the LFA policy outperforms LISO to some considerable extent, and that the PG with LRM finds a better solution than with FDM.
- We introduce memory into the stochastic processes for content (lifetime) generation and channel quality, and show via simulations that the performance gain of the LFA policy over LISO is higher in such scenarios.

II. System Model

We consider a slotted (discrete time) system model (see Fig. 1). At the beginning of each time slot t, a random number of contents, denoted by M_t, are generated. We denote the set of newly generated contents by $\mathcal{N}_t$, where $|\mathcal{N}_t| = M_t$. Each content is generated with a lifetime, after which it becomes irrelevant, and the lifetime of every content is assumed to be known perfectly by the CM when they are generated. In particular, if, at the beginning of time slot t, content i is generated with lifetime $K_{t,i}$, it can be consumed in time slots $t, t+1, \ldots, t+K_{t,i}-1$, and otherwise will be removed from the system after time slot $t+K_{t,i}-1$. We denote the set of contents that are already in the cache at the beginning of time slot t by $\mathcal{I}_t$, and the set of relevant contents not inside the cache, including the M_t newly generated contents, by $\mathcal{O}_t$.

At each time slot, the user either accesses the system and consumes all the relevant contents, or does not access the system. User access behavior is represented by the binary

random variable U_t; that is, $U_t = 1$ if the user accesses the system, and $U_t = 0$ otherwise. When $U_t = 1$, all the contents that are not in the cache, $\mathcal{O}_t$, are downloaded, and moved, together with all the contents already in the cache, $\mathcal{I}_t$, to the application layer. Even if $U_t = 0$, the CM has the option of downloading some contents to the cache, and removing others if needed. We denote the set of contents that are downloaded at time slot t by $A_t^{(1)} \subset \mathcal{O}_t$, and those that are discarded from the cache by $A_t^{(2)} \subset \mathcal{I}_t$. To unify notation, if $U_t = 1$ we set $A_t^{(1)} = \mathcal{O}_t$ and $A_t^{(2)} = \mathcal{I}_t$.[3]

Since all the contents have the same size, it will be convenient to represent each content by its remaining lifetime. Following this representation, all sets of contents, that is, $\mathcal{N}_t$, $\mathcal{O}_t$, $\mathcal{I}_t$, $A_t^{(1)}$ and $A_t^{(2)}$, are multisets of remaining lifetimes (positive integers, with the set of all positive integer tuples denoted by $\mathbb{N}^*$). To simplify the treatment, when it does not cause confusion, we will only talk about sets instead of multisets, or subsets instead of sub-multisets of multisets, and operations, such as union, should be treated in a multiset manner. For a multiset $\mathcal{Y}$ with positive elements, we let $\mathcal{Y} - 1 = \{y > 0 : y + 1 \in \mathcal{Y}\}$ denote the multiset obtained by reducing each element of $\mathcal{Y}$ by 1 and removing the elements which become 0. With these definitions in mind, if $U_t = 0$, the system evolves according to the following equations:

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

and according to the following equations if $U_t = 1$:

$$\mathcal{I}_{t+1} = \emptyset \quad \text{and} \quad \mathcal{O}_{t+1} = \mathcal{N}_{t+1}. \tag{2}$$

We assume that the user is equipped with a cache of capacity B, that is, $|\mathcal{I}_t| \leq B$, for all t. Hence, the CM's actions, $A_t = (A_t^{(1)}, A_t^{(2)})$, are constrained by the available cache capacity, and any valid action leads to a new state with $|\mathcal{I}_t| \leq B$.

Downloading a content at time t has a cost C_t that depends on the channel state. The total instantaneous cost at time t is $\mu_t = |A_t^{(1)}| \cdot C_t$, while the average cost after T time slots is given by $J_T = \frac{1}{T}\sum_{t=1}^{T} \mu_t$. The goal is to minimize the long-term expected average cost defined as

$$\rho \triangleq \limsup_{T\to\infty} \mathbb{E}\left[J_T\right] = \limsup_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mu_t\right].$$

1) User Access Model: We assume that the user access sequence $\{U_t\}$ is an arrival process with i.i.d. inter-arrival times $\{D_n\}$, where D_n denotes a positive-integer-valued random variable. Throughout, we will make one of the two assumptions regarding D_n: (i) Bounded inter-arrival times, i.e., $0 \leq D_n \leq D_{max}$; (ii) Geometric inter-arrival times: D_n has a geometric distribution with parameter p_a; hence, $\{U_t\}$ is an i.i.d. process with $\mathbb{P}\left[U_t = 1\right] = p_a$ (in turn, $D_{\max} = \infty$ in this case). The latter assumption is standard in the literature, is known as the *independent reference model* (IRM) [10], [18].

[3]We do not allow the CM to download and remove the same content in the same time slot, which is obviously suboptimal.

2) Content Generation Model: We assume that $\{M_t\}$ is an i.i.d. sequence with generic random variable M, and is upper-bounded by $M_{max} \in \mathbb{Z}^+$. We further assume that the lifetimes are also i.i.d. with generic random variable K, and upper-bounded by $K_{\max} \in \mathbb{Z}^+$.

3) Channel Model: We assume that the energy cost for downloading a content $C_t > 0$ is a continuous random variable with cumulative distribution function (cdf) $F_C(c)$, and it is assumed to be i.i.d. across time, and bounded by $C_{max} \in \mathbb{R}^+$. Aside from simplifying our system model, the i.i.d. assumption here is appropriate for micro BS deployments, where the user switches micro BSs across time slots. We assume that the micro BSs can operate at the same time without any interference because they operate at a relatively low transmit power. We also assume zero download delay [10], [13], implying that the duration of a time slot is long enough to download the required contents. Hence, the channel is approximately ergodic within a time slot, and is only subject to large-scale fading effects.

In the rest of the paper, we assume that the sequences $\{C_t\}, \{D_n\}, \{M_t\}, \{K_{t,i}\}$ are independent of each other. In the following section, we assume that the CM is aware of the above stochastic model governing the system behavior.

III. Optimal solution

In this section we derive a general result concerning the structure of the optimal cache management policy. First, we define a special class of MDPs, called MDP-SI, and show that our problem is an instance of this class. Then, we derive a general structural result for optimal policies in MDP-SI under some assumptions, and show that they apply to our problem.

A. Standard MDP model

A finite-state finite-action MDP is characterized by a quadruple $(\mathcal{S}, \mathcal{A}, P, \mu)$, where $\mathcal{S}$ and $\mathcal{A}$, the state and action spaces, respectively, are finite sets, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a probability kernel (we will write $P(s'|s, a)$ instead of $P(s, a, s')$), and $\mu : \mathcal{S} \times \mathcal{A} \to [0, \mu_{max}] \cup \{\infty\}$ is a cost function with some $\mu_{max} > 0$. The purpose of introducing an infinite cost is to allow a different action set in every state without complicating the notation too much: 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 (otherwise the agent suffers infinite cost), and we assume that $\mathcal{A}_s \neq \emptyset$, for all $s \in \mathcal{S}$. In an MDP, an agent controls a Markov chain and pays some cost over time. Assuming the agent selects an action $a \in \mathcal{A}_s$ at state $s \in \mathcal{S}$, the system evolves to state s' with probability $P(s'|s, a) \triangleq \mathbb{P}\left[S_{t+1} = s' | S_t = s, A_t = a\right]$ (for any time slot t), where $\sum_{s'\in\mathcal{S}} P(s'|s, a) = 1$, for all $s \in \mathcal{S}, a \in \mathcal{A}$. The cost of taking action a in state s is $\mu(s, a)$. Denoting the state of the system at time t by S_t and the agent's action by A_t, the agent's goal is to minimize the infinite horizon average cost $\rho = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mu(S_t, A_t)\right]$.

A deterministic policy is a mapping $\pi : \mathcal{S} \to \mathcal{A}$, which selects a single action for each state; and let Π denote the set of all deterministic policies. For policy π, let $P^\pi : \mathcal{S}\times\mathcal{S} \to [0, 1]$ denote the transition kernel induced by π, that is $P^\pi(s'|s) = P(s'|s, \pi(s))$. Assuming the Markov chain defined by P^π is

irreducible and aperiodic for all π, let ρ^π denote the infinite-horizon average cost ρ when $A_t = \pi(S_t)$, that is,

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

Due to our assumption on P^π, the initial state S_0 does not matter, and the limit in (3) exits thanks to the non-negativity assumption on $\mu(s,a)$. It is well-known (see, e.g., [19]) that there exists a deterministic policy π^* that minimizes the infinite-horizon average cost over all, possibly non-stationary and non-deterministic causal control policies, that is,

$$\pi^* = \operatorname*{argmin}_{\pi} \rho^\pi, \tag{4}$$

where the minimum is taken over all admissible (causal) control strategies, in which A_t may depend on the history $H_t \triangleq (S_1,\dots,S_t,A_1,\dots,A_{t-1})$ and some randomization.

B. MDPs with side information (MDP-SI)

In the MDP-SI model, we extend the classical MDPs such that there is an i.i.d. sequence of side information $Z_t \in \mathcal{Z}$ for some $\mathcal{Z} \subset \mathbb{R}$, which is available to the agent before selecting A_t, and effects the cost μ, that is, $\mu : \mathcal{S}\times\mathcal{A}\times\mathcal{Z} \to [0,\mu_{max}]\cup\{\infty\}$. Then the decision of the agent may depend on H_t, the randomization, and $(Z_1,\dots,Z_t)$. This setup can be easily modeled in the MDP framework by changing the state space to $\mathcal{S}\times\mathcal{Z}$, but if $\mathcal{Z}$ is not finite, the analysis of the resulting MDP is significantly more complicated. Before delving into the analysis of the MDP-SI model, first we show that our problem can be cast as an MDP-SI problem.

At the end of time slot t, the state of the contents can be described by the sets $\mathcal{I}_t$ and $\mathcal{O}_t$, while the state of the user can be described by the time elapsed since the last access, denoted by E_t. To be precise, we assume that the user accesses the OSN at time $t=0$ (i.e., we set $U_0 = 1$); then E_t is defined as $E_t \triangleq \min\{t-n : t>0, 0\le n\le t, U_n = 1\}$. We denote by $\mathcal{S} \subset \mathbb{N}^*\times\mathbb{N}^*\times\mathbb{N}$ the set of all possible combinations of $\mathcal{O}_t,\mathcal{I}_t$, and E_t. That is, the system state in time slot t is $S_t = (\mathcal{O}_t,\mathcal{I}_t,E_t)$. Under the IRM user-access model (i.e., when the user-access process is i.i.d. and the inter-access times are geometrically distributed), the memoryless property of the geometric distribution implies that the exact value of E_t does not affect the future given that $E_t > 0$, and hence, the state of the user can be redefined as $\mathbb{I}_{\{E_t>0\}}$, the indicator function of the event $\{E_t>0\}$.[4] Unless otherwise stated explicitly, we will use $\mathbb{I}_{\{E_t\}}$ in place of E_t for the IRM model; accordingly, $\mathcal{S}$ will denote the possible combinations of $\mathcal{O}_t,\mathcal{I}_t$, and $\mathbb{I}_{\{E_t>0\}}$. Note that under both of our user-access models (i.e., IRM or bounded inter-access times–$D_{\max}<\infty$), the state space $\mathcal{S}$ is finite. Furthermore, let $\mathcal{A}_s$ denote the set of download/discard actions available to the CM in a state $s\in\mathcal{S}$. The action of the agent in time slot t is the pair $A_t = (A_t^{(1)}, A_t^{(2)})$, and C_t can be regarded as the i.i.d. side information Z_t. Indeed, the decision of the CM (i.e., the agent) depends on C_t, as the cost of action A_t is $\mu(S_t,A_t,C_t) = C_t\cdot|A_t^{(1)}|$.

[4]For an event $\mathcal{E}$, $\mathbb{I}_{\{\mathcal{E}\}}=1$ if $\mathcal{E}$ holds, and 0 otherwise.

The state $s\in\mathcal{S}$ of the system evolves according to (1) and (2), where the user access sequence depends on E, which evolves independently according to the distribution of D_n. The channel cost C_t, which is the side information, also evolves independently, with cdf F_C in every time slot t. These independence assumptions ensure that the resulting model is indeed an MDP-SI.

C. Structure of the optimal policy in MDP-SI

In this section we derive the structure of the optimal policy for a general MDP-SI under certain conditions. To begin with, assume we have an MDP-SI characterized by $(\mathcal{S},\mathcal{A}_{SI},P_{SI},\mu_{SI},\mathcal{Z},F_Z)$, where F_Z is the cdf of the real-valued side information, $\mathcal{S}$ and $\mathcal{A}_{SI}$ are countable, and $\mathcal{Z}\subset\mathbb{R}$. Let $\mathcal{A}$ denote the set of Borel-measurable[5] functions $\{g:\mathcal{Z}\to\mathcal{A}_{SI}\}$, and consider the MDP $(\mathcal{S},\mathcal{A},P,\mu)$ where

$$P(s'|s,g) = \mathbb{E}\left[P_{SI}(s'|s,g(Z))\right], \tag{5}$$

and

$$\mu(s,g) = \mathbb{E}\left[\mu_{SI}(s,g(Z),Z)\right], \tag{6}$$

where the expectations are taken over F_z. It is easy to see that any deterministic policy $\pi_{SI}:\mathcal{S}\times\mathcal{Z}\to\mathcal{A}_{SI}$ for the MDP-SI can be turned into a deterministic policy for the corresponding MDP using

$$\pi(s) = \pi_{SI}(s,\cdot)\in\mathcal{A}, \tag{7}$$

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 $(\mathcal{S},\mathcal{A},P,\mu)$. If $\mathcal{Z}$ is finite, the new MDP is finite, and we can use standard results (see, e.g., [19]) to analyze the structure of the optimal policy: Assume that the MDP is finite, P^π is irreducible and aperiodic for any deterministic policy $\pi\in\Pi$, and let $S_1,A_1,S_2,A_2,\dots$ denote the state-action sequence obtained by following policy π. Then ρ^π in (3) exists, and the differential value function for any state $s\in\mathcal{S}$ is defined as

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=1}^{\infty}(\mu(S_t,\pi(S_t))-\rho^\pi)\,\middle|\,S_1=s\right]. \tag{8}$$

Furthermore, the optimal policy π^* in (4) satisfies

$$V^{\pi^*}(s) = \min_{a\in\mathcal{A}}\left\{\mu(s,a)-\rho^{\pi*}+\sum_{s'\in\mathcal{S}}P(s'|s,a)V^{\pi^*}(s')\right\}, \tag{9}$$

and $a=\pi^*(s)$ minimizes the right hand side. While these results make the analysis easy, unfortunately they do not directly apply to our case, where the state space $\mathcal{S}$ can be countably infinite and, due to the fact that $\mathcal{Z}$ is not finite, the action set $\mathcal{A}$ is infinite (and uncountable). Luckily, it is possible to extend the above results, specifically (9), to the MDP $(\mathcal{S},\mathcal{A},P,\mu)$ when $\mathcal{Z}$ is an interval (this will be done in the proof of Lemma 1). Using the definition of the MDP in (5)–(7) and the expression for the optimal value function in

[5]Throughout the paper we assume the existence of the necessary probability spaces and the measurability of functions as required.

(9), we can prove the following property of the optimal policy $\pi^*(s,\cdot)$ of an MDP-SI (the proof is given in Appendix A).

Lemma 1. *Consider the countable-state, finite-action MDP-SI problem $(\mathcal{S}, \mathcal{A}_{SI}, P_{SI}, \mu_{SI}, \mathcal{Z}, F_Z)$. Suppose that P^π is ergodic for any policy π in the corresponding MDP $(\mathcal{S}, \mathcal{A}, P, \mu)$ and that $\mathcal{Z}$ is an interval. Then* (9) *holds for $(\mathcal{S}, \mathcal{A}, P, \mu)$. Furthermore, if $\mu_{SI}(s, a_{SI}, z)$ is a linear function of z for any $s \in \mathcal{S}, a_{SI} \in \mathcal{A}_{SI}$, then the optimal policy $\pi^*(s,\cdot)$ is a piecewise constant function for any $s \in \mathcal{S}$.*

Combining the MDP-SI formulation of Section III-B with Lemma 1, we obtain that for any state $s \in \mathcal{S}$, the optimal decision is a piecewise constant function of the channel cost C_t with values taken from $\mathcal{A}_s$. Also note that the technical condition in the lemma that P^π is ergodic is easily satisfied in our cache management problem due to the user access model: The fact that the user access clears all contents from both inside and outside of the cache at least once in every $D_{\max}$ time slots or after a geometric waiting time ensures that any state visited with positive probability is positive recurrent. Thus, to achieve ergodicity, it is sufficient to guarantee that the process is aperiodic. This readily follows from the IRM model, and also holds for the bounded inter-access time model under mild assumptions (e.g., if the user can access the OSN in two consecutive time slots with positive probability).

D. Structure of the optimal cache management policy

Here we will describe the structure of the optimal policy for the proactive caching problem.

We start with the technical definition of partial ordering for multisets, which will be useful to characterize the effect of good actions: For two multisets $\mathcal{Y}_1$ and $\mathcal{Y}_2$ with nonnegative elements, we write $\mathcal{Y}_1 \leq \mathcal{Y}_2$, if, either (i) they are of equal size and there is a one-to-one mapping between the elements of $\mathcal{Y}_1$ and $\mathcal{Y}_2$ such that the element from $\mathcal{Y}_1$ is never larger than the corresponding element from $\mathcal{Y}_2$; or (ii) if they are of different size, but the same relationship holds after adding zeros to the smaller set to equalize their sizes.

Now consider two states of the MDP describing the caching problem: $s = (\mathcal{O}, \mathcal{I}, E) \in \mathcal{S}$ and $s' = (\mathcal{O}', \mathcal{I}', E') \in \mathcal{S}$. We will say that s *is better than* s', and write $s \succeq s'$, if $E = E'$, the remaining lifetimes of all the contents are the same, that is, $\mathcal{O} \cup \mathcal{I} = \mathcal{O}' \cup \mathcal{I}'$, and $\mathcal{O} \leq \mathcal{O}'$ and $\mathcal{I} \geq \mathcal{I}'$. Intuitively, $s \succeq s'$ means that the same contents are available for pre-caching in s and s', but in state s, "better" contents have already been downloaded to the cache (i.e., the contents in the cache remain relevant longer while the ones outside expire earlier). The next lemma formalizes this statement:

Lemma 2. *Assume the conditions of Lemma 1 hold. Let $s, s' \in \mathcal{S}$ and suppose that $s \succeq s'$. Then, $V^{\pi^*}(s) \leq V^{\pi^*}(s')$, that is, the future average download cost starting form s is not larger than the cost starting from s'.*

Proof. It is easy to see that if any action a' is performed in s', it is always possible to find another action $\hat{a}$ in s such that the cost of $\hat{a}$ is no more than that of a', that is, $\mu(s', a') \geq \mu(s, \hat{a})$, and the resulting new states satisfy $\hat{s}_2 \succeq s'_2$, where s'_2 and $\hat{s}_2$ denote the next state for the chains starting from s' and s, respectively, assuming the content generation process and the user access process are the same (e.g., if a' downloads a content from outside the cache of s', $\hat{a}$ should download the content with the largest remaining lifetime from outside the cache of s, unless all the contents in the cache of s have larger lifetimes, in which case $\hat{a}$ should not do anything). Now consider three coupled realizations of the MDP: $\{(S'_t, A'_t)\}$ starts from $S'_1 = s'$, and follows the optimal policy π^*; the second realization $\{(\hat{S}_t, \hat{A}_t)\}$ starts from $\hat{S}_1 = s$, and selects $\hat{A}_t$ such that $\hat{S}_t \succeq S'_t$ and $\mu(\hat{S}_t, \hat{A}_t) \leq \mu(S'_t, A'_t)$ for all t; finally, $\{(S_t, A_t)\}$ starts form $S_1 = s$, and follows the optimal policy π^*. Then, using the optimality of A_t and π^*, by (9) (which holds by Lemma 1), we have

$$
\begin{aligned}
&V^{\pi^*}(s) = V^{\pi^*}(S_1) \leq \mu(\hat{S}_1, \hat{A}_1) - \rho^{\pi^*} + \mathbb{E}\left[V^{\pi^*}(\hat{S}_2)\right] \\
&\leq \mu(\hat{S}_1, \hat{A}_1) - \rho^{\pi^*} + \mathbb{E}\left[\mu(\hat{S}_2, \hat{A}_2) - \rho^{\pi^*} + \mathbb{E}\left[V^{\pi^*}(\hat{S}_3)\right]\right] \\
&\quad \vdots \\
&\leq \mathbb{E}\left[\sum_{t=1}^{\infty} (\mu(\hat{S}_t, \hat{A}_t) - \rho^{\pi^*}) \,\middle|\, \hat{S}_1 = s\right] .
\end{aligned}
$$

Furthermore, by the coupling of the realizations,

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{\infty} (\mu(\hat{S}_t, \hat{A}_t) - \rho^{\pi^*}) \,\middle|\, \hat{S}_1 = s\right] \\
&\qquad \leq \mathbb{E}\left[\sum_{t=1}^{\infty} (\mu(S'_t, A'_t) - \rho^{\pi^*}) \,\middle|\, S'_1 = s'\right] = V^{\pi^*}(s') .
\end{aligned}
$$

Putting everything together, we obtain $V^{\pi^*}(s) \leq V^{\pi^*}(s')$. □

Next we express the actions in $\mathcal{A}_s$ more intuitively by defining a *simple action*, which we also denote as a, to simplify the notation.

Definition 1 (Simple Action). *For any $l \in \mathcal{I}$ and $L \in \mathcal{O}$ (recall that l and L denote the remaining lifetime of some contents), a simple action $a = (l|L)$ is defined as follows: If $E_t > 0$, $a = (l|L)$ replaces a cache content with remaining lifetime l with a relevant content outside the cache with remaining lifetime L, by removing the former content from the cache and downloading and caching the latter; i.e., it "swaps" the two contents. If $E_t = 0$, $a = (l|L)$ downloads the content with remaining lifetime L, and moves both contents to the app.*

In this definition, $l = 0$ means that the content with lifetime L is downloaded without removing any content from the cache. Similarly, $L = 0$ means that no content is downloaded while a content with lifetime l is removed. Note that, with the optimal policy, the latter (i.e., $L = 0$) can only happen if either $l = 0$ (i.e., an expired content is removed from the cache), or when $E_t = 0$ and more contents are moved from the cache to the app than those downloaded from the OSN server to the app. At every time slot t, due to the cache capacity constraint, the CM can take up to B simple actions if $E_t > 0$. Therefore, at such instances, any action of an optimal policy can be expressed as at most B consecutive simple actions,

and an action $A_t = (\{L_1, \dots, L_b\}, \{l_1, \dots, l_b\})$,[6] for some $0 \le b \le B$ can be written as a sequence of simple actions $\{(l_1|L_1) \cdots (l_b|L_b)\}$.

For a state $s = (\mathcal{O}, \mathcal{I}, E)$ with $E > 0$, assume that $l_1 \le \cdots \le l_B$ are the contents in $\mathcal{I}$, and $L_1 \ge \cdots \ge L_B$ denote the B largest elements in $\mathcal{O}$. To find the optimal action, first we determine the best simple action. Let s_1^* denote the next state if action $(l_1|L_1)$ is taken, and let s_1' denote the state after a different simple action $(l'|L')$. Since l_1 is the smallest element of the cache, and L_1 is the largest element outside, assuming the same content generation, it is immediate that $s_1^* \succeq s_1'$. Then, by Lemma 2, $V^{\pi^*}(s_1^*) \le V^{\pi^*}(s_1')$. Therefore, by (9), $(l_1|L_1)$ is the best simple action. Considering larger actions composed of b simple actions for $b \ge 2$, it follows similarly that the optimal action is $A^b = \{(l_1|L_1) \cdots (l_b|L_b)\}$ (note that the energy cost associated with such an action is $b\,C$, where C is the channel cost of a single download). To find the optimal action, it remains to compare the actions A^b for different values of $b \in \{0, \dots, B\}$. Denoting the next state following action A^b by s^b, the relative action value of A^b for channel cost C is given by

$$Q^{\pi*}(s, A^b, C) = b\,C - \rho^{\pi^*} + \mathbb{E}\left[V^{\pi^*}(s^b)\right],$$

and, by (9), the optimal action is the one minimizing $Q^{\pi*}(s, A^b, C)$ for a given C: that is, A^{b^*} with $b^* = \operatorname{argmin}_b Q^{\pi*}(s, A^b, C)$. Notice that, as a function of C, $Q^{\pi*}(s, A^b, C)$ is a linear function with slope b and intersecting the y axis at $\mathbb{E}\left[V^{\pi^*}(s^b)\right]$. Since, obviously, $s^b \succeq s^{b'}$ for any $b > b'$, we have $V^{\pi^*}(s^b) \le V^{\pi^*}(s^{b'})$, and so $\mathbb{E}\left[V^{\pi^*}(s^b)\right]$ is non-increasing in b. Therefore, there exist thresholds $0 = \mathcal{T}_{B+1} \le \mathcal{T}_B \le \cdots \le \mathcal{T}_1 \le C_{max}$, such that the optimal action is A^b if the channel cost belongs to the interval $[\mathcal{T}_{b+1}, \mathcal{T}_b]$. Since $A^{b'} \subset A^b$ for $b > b'$, this also means that the simple action $a^b = (l_b|L_b)$ is performed whenever $C \le \mathcal{T}_b$ (note that $\mathcal{T}_b = 0$ means that action a^b is never performed because $C > 0$). This implies the following theorem.

Theorem 1. *Let $s = (\mathcal{O}^s, \mathcal{I}^s, E^s) \in \mathcal{S}$ denote a state of the MDP-SI, and let C denote the channel cost. Let $l_1^s \le \cdots \le l_B^s$ denote the contents in $\mathcal{I}^s$, and $L_1^s \ge \cdots \ge L_B^s$ denote the B largest elements of $\mathcal{O}^s$. Then, for all states s, there exist thresholds $0 \le \mathcal{T}_B^s \le \mathcal{T}_{B-1}^s \le \cdots \le \mathcal{T}_1^s \le C_{max}$, and an optimal caching policy that, in state s, performs the simple actions $a^i = (l_i^s|L_i^s)$ for all i such that $C \le \mathcal{T}_i^s$ if $E^s > 0$ (i.e., the user does not access the OSN).*

The thresholds for different simple actions depend on what other simple actions are available, and on the cache contents. This is because if we cache a content, its value depends on the likelihood of the content to be removed from the cache before being consumed by the user, and this likelihood is affected by the lifetime of the other contents in the cache.

We still have to evaluate the optimal thresholds to characterize the optimal policy and its performance. The number of thresholds to be determined is in the order of the cardinality of the state space $\mathcal{S}$, which is extremely large. This makes it computationally infeasible to compute the optimal threshold values. Interestingly, this is not the case if we have a sufficiently large cache capacity, e.g., $B \ge M_{\max} K_{\max}$, in which case we never remove a content from the cache unless it is consumed, or has expired; and hence, we can decide about each content individually, and independently of the cache contents. We refer to this as the case of unlimited cache capacity.

Corollary 1. *For unlimited cache capacity, i.e., $B = \infty$, for any state $s = (\mathcal{O}, \mathcal{I}, E)$ with $E > 0$, there exist thresholds $0 \le \mathcal{T}_{1,E} \le \cdots \le \mathcal{T}_{K_{\max},E} \le C_{max}$ (where $K_{\max}$ is the maximum lifetime), which depend only on E, such that a content with remaining lifetime $L \in \mathcal{O}$ is downloaded if $C \le \mathcal{T}_{L,E}$.*

Since the decision to download any content is independent of the others, the problem can be modeled as a finite-horizon MDP-SI, where the horizon equals the remaining lifetime L with maximum horizon $K_{\max}$. Thus, we can apply dynamic programming [19] to determine the optimal thresholds recursively: Let $V_{L,E}$ denote the future cost associated with a content with lifetime L from a state with time E since the past user access following an optimal policy. Since there is no need to proactively download a content with lifetime 1, $\mathcal{T}_{1,E} = 0$ for all $E > 0$, and so $V_{1,E} = 0$ for any $E > 0$. Let $p_E = \mathbb{P}[U_{t+1} = 1 | E_t = E] = \mathbb{P}[D_1 \le E + 1] - \mathbb{P}[D_1 \le E]$ denote the probability of user access in the next time slot.[7] Assuming optimal decisions are made for lifetimes up to $L-1$ for all E, a decision with threshold $\mathcal{T}$ for lifetime $L > 1$ and elapsed time E has a future download cost

$$\begin{aligned} V_{L,E,\mathcal{T}} = {} & \mathbb{P}[C \le \mathcal{T}]\,\mathbb{E}[C | C \le \mathcal{T}] \\ & + \mathbb{P}[C > \mathcal{T}]\left(p_E \mathbb{E}[C] + (1 - p_E) V_{L-1,E+1}\right). \end{aligned} \tag{10}$$

By setting the derivative of the above expression to zero, we obtain the optimal threshold $\mathcal{T}_{L,E} = p_E \mathbb{E}[C] + (1 - p_E) V_{L-1,E+1}$, which is exactly the expected future cost if the content is not downloaded in the current state. Noticing that $\mathcal{T}_{L,E}$ equals the last term in parentheses in (10), we obtain the following result.

Corollary 2. *For unlimited cache capacity, i.e., $B_{\max} = \infty$, the optimal thresholds $\mathcal{T}_{L,E}$ can be computed recursively as follows: $\mathcal{T}_{1,E} = 0$, $\forall E > 0$. For $L \ge 1$, given $\mathcal{T}_{L,E}$ for all E, the optimal thresholds for $L+1$ can be obtained for all E as*

$$\begin{aligned} \mathcal{T}_{L+1,E} = {} & p_E \mathbb{E}[C] \\ & + (1 - p_E)\Big(\mathbb{P}[C \le \mathcal{T}_{L,E+1}]\,\mathbb{E}[C | C \le \mathcal{T}_{L,E+1}] \\ & + \mathbb{P}[C > \mathcal{T}_{L,E+1}]\,\mathcal{T}_{L,E+1}\Big). \end{aligned}$$

For the IRM user access model, the same thresholds can be used in all states, and the expression for the thresholds simplifies to $\mathcal{T}_1 = 0$, and for $L \ge 1$,

$$\begin{aligned} \mathcal{T}_{L+1} = {} & p_a \mathbb{E}[C] + (1 - p_a) \\ & \cdot \left(\mathbb{P}[C \le \mathcal{T}_L]\,\mathbb{E}[C | C \le \mathcal{T}_L] + \mathbb{P}[C > \mathcal{T}_L]\,\mathcal{T}_L\right). \end{aligned} \tag{11}$$

[6] If either $|\mathcal{O}|$ or $|\mathcal{I}|$ is less than b, we simply zero-pad the set so that $|A_t^{(1)}| = |A_t^{(2)}| = b$.

[7] Recall that D_n denotes the nth inter-access time, and D_n are i.i.d.

The optimal performance with an infinite cache capacity will be studied as a lower bound on the optimal performance for a practical finite cache capacity system in Section VI.

IV. Low-Complexity Caching Schemes via Policy Approximation

According to Theorem 1, the optimal policy has a threshold structure, and the threshold for each simple action depends in general on the remaining lifetimes of all the relevant contents and the time elapsed since the last user access. Hence, the optimal policy may employ different threshold values for the same simple action at different states, and it belongs to the family of policies parametrized by these thresholds. The dimension of this policy space is $|\bar{\mathcal{S}}|$, where $\bar{\mathcal{S}} \subset \mathcal{S}$ denotes the set of states in which the user does not access the OSN. Moreover, we have approximately $K_{\max}^2/2$ potential simple actions, each of which can have a different threshold at each state. For any reasonable cache size B, this is huge; and hence, it is infeasible to compute an optimal policy (e.g., if $M_{\max} \geq B$, then just the cache content $\mathcal{I}$ can take $\binom{B+K_{\max}}{K_{\max}}$ different values, which is already prohibitively large for even moderate values of B or $K_{\max}$). To resolve this problem, we use policy approximation techniques, and approximate the policy space using some simple parametrized form.

From now on we adopt the IRM user access model, which alleviates the need to consider the time E_t elapsed since the last user access, reducing the state space. We introduce two low-dimensional approximations to the policy space, which allow us to run optimization algorithms (policy search algorithms, described in Section V) to find computationally feasible policies with good performance. These schemes are not based on the knowledge of the system statistics, and optimize the policy parameters based on observations (which can be obtained either from interactions with the real system, or via simulations through a generative model). Therefore, the proposed methods can be used in a learning context, where an agent learns from its actions and updates its policy to adapt to the unknown environment in a reinforcement learning fashion.

A. LISO policy

LISO is a suboptimal threshold-based caching policy with a simplified structure. It employs a single threshold value for each simple action (corresponding to the content pair consisting of the content with the shortest remaining lifetime in the cache and the one with the longest remaining lifetime outside the cache), independent of the state. For every such pair, if the channel cost is below this threshold, the two contents are "swapped," and no action is taken otherwise. LISO is directly parametrized by the threshold values:

$$\mathcal{T}(l|L) = \theta(l, L),$$

where $\theta(l, L) \in [0, C_{\max}]$, for all pairs $a = (l|L)$, $l, L \in \{0, \dots, K_{\max}\}$. The set of policies parametrized this way is of dimension $(K_{\max}+1)^2$, which is feasible. We can further reduce the dimension by explicitly forbidding simple actions $(l|L)$ with $l \geq L$ (by setting the corresponding $\theta(l, L)$ to zero), since an optimal policy will not replace a cached content with a content with a shorter remaining lifetime. Hence, for such simple actions, we have $\mathcal{T}(l|L) = 0$. We also note that the optimal policy has a monotonic structure; that is, $\mathcal{T}(l|L_1) \leq \mathcal{T}(l|L_2)$ if $L_1 < L_2$ and $\mathcal{T}(l_1|L) \geq \mathcal{T}(l_2|L)$ if $l_1 < l_2$, which further limits the search space, and speeds up the policy search.

B. LFA policy

Next, we propose an improved policy representation (an extension of LISO), which takes into account the remaining lifetimes of the contents in the cache memory when determining the threshold values; which can be useful in estimating the likelihood that a downloaded content will be removed from the cache before it expires or is consumed. To characterize the state of the cache, we define *features* of the cache-state based on the number of contents in the cache with a particular remaining lifetime. We define the vector $\Phi_t \triangleq [\phi_t(0), \phi_t(1), \dots, \phi_t(K_{\max})]$, where $\phi_t(i)$ denotes the ratio of the number of contents with lifetime i in the cache at time t, that is,

$$\phi_t(i) \triangleq \frac{\sum_{l \in \mathcal{I}} \mathbb{I}_{\{l=i\}}}{B}, \quad \text{for} \quad i = 0, 1, \cdots, K_{\max},$$

where $l = 0$ denotes the empty locations as before. Clearly, $0 \leq \phi_t(i) \leq 1$, and $\sum_{i=0}^{K_{\max}} \phi_t(i) = 1$. To keep the computational complexity feasible, the threshold value for each simple action $a(l|L)$ for $l < L$, $l, L \in \{0, \dots, K_{\max}\}$, is defined as a linear function of vector Φ_t as

$$\mathcal{T}(l|L) = \sum_{i=0}^{K_{\max}} \phi_t(i)\theta_i(l, L) = \Phi_t^\top \theta(l, L),$$

where $\theta_i(l, L) \in \mathbb{R}$ for $l < L$, and $\theta_i(l, L) = 0$ otherwise. This results in a $K_{\max}(K_{\max}+1)^2/2$-dimensional policy space.

Remark 1. *We remark that the LISO policy, which is directly parametrized by the threshold values for each simple action ignoring the other contents in the cache, is a special case of the LFA policy with parameters $\theta_i(l, L) = \theta(l, L)$ for all i.*

In the next section, we describe two policy search algorithms that we use to optimize the parameters of the proposed approximate caching schemes.

V. Policy Search Methods

Optimizing parametric policies for MDPs has been extensively studied in reinforcement learning [20]. We are going to employ policy gradient (PG) methods [17] to optimize the parameters of our LISO and LFA policies. PG methods are model-free reinforcement learning algorithms to find an optimal policy in an MDP by running gradient descent over the policy space to minimize the expected average cost $\rho^{\pi_{\boldsymbol{\theta}}}$, where $\pi_{\boldsymbol{\theta}}$ denotes the policy defined by the parameter vector $\boldsymbol{\theta}$. In every step of the policy gradient algorithm, parameter $\boldsymbol{\theta}_j$ is updated using the gradient $\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}}$ of $\rho^{\pi_{\boldsymbol{\theta}}}$ as

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \lambda \nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}_j}}, \tag{12}$$

for some positive step size λ.

Since the gradient $\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}_j}}$ is not known in closed form in most cases, the gradient (and the average cost of the policy)

has to be estimated through sample averages over independent, finite trajectories obtained via Monte Carlo rollouts, i.e., instead of (12) we use a random estimate of the gradient; so, in practice, (12) becomes a stochastic gradient descent algorithm. To curtail the effect of noise introduced due to the randomness, we obtain $\boldsymbol{\theta}_{j+1}$ as the average of m policy updates, i.e., $\boldsymbol{\theta}_{j+1} = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{\theta}_{j+1,i}$, where each $\boldsymbol{\theta}_{j+1,i}$ is obtained using (12) with an independent estimate of the gradient. The estimation procedure usually requires two steps:

1) *Policy evaluation*: The average cost of a sample trajectory $\tau_{\boldsymbol{\theta}} = (S_1, C_1, A_1), \ldots, (S_T, C_T, A_T)$, obtained by following policy $\pi_{\boldsymbol{\theta}}$ with parameter vector $\boldsymbol{\theta}$, is found as
$$J(\tau_{\boldsymbol{\theta}}) = \frac{1}{T}\sum_{t=1}^{T}\mu(S_t, A_t, C_t).$$
2) *Policy exploration*: New sample trajectories are generated. Exploration is implemented either directly on the actions A_t, or on the parameter vector $\boldsymbol{\theta}$, by introducing an exploration *noise* either at every time step of the trajectory, or at the beginning of the trajectory.

In what follows, we review two practical policy gradient algorithms that employ different estimation techniques.

A. Finite difference method (FDM)

In FDM, the gradient is estimated by generating sample trajectories following policy $\pi_{\boldsymbol{\theta}_j}$, specified by parameter vector $\boldsymbol{\theta}_j$ (determining the threshold values $\mathcal{T}(l|L)$), and by new policies obtained by applying small perturbations $\Delta\boldsymbol{\theta}^{[i]}$ to $\boldsymbol{\theta}_j$. Generating trajectory $\tau^{[i]}$ for $\boldsymbol{\theta}_j$ and $\tau_{\Delta}^{[i]}$ for $\boldsymbol{\theta}_j + \Delta\boldsymbol{\theta}^{[i]}$, the change in the cost is estimated by
$$\Delta J^{[i]} = J\left(\tau_{\Delta}^{[i]}\right) - J\left(\tau^{[i]}\right), \tag{13}$$
which is approximately equal to $(\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}})^{\top}\Delta\boldsymbol{\theta}^{[i]}$. Thus, generating N independent trajectories $\tau^{[i]}$, $i = 1, \ldots, N$, the gradient can be estimated from $\Delta\boldsymbol{J}_{\pi_{\boldsymbol{\theta}}} = [\Delta J^{[1]}, \cdots, \Delta J^{[N]}]^{\top}$ and $\Delta\boldsymbol{\Theta} = [\Delta\boldsymbol{\theta}^{[1]}, \cdots, \Delta\boldsymbol{\theta}^{[N]}]^{\top}$ by linear regression as:
$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} \approx \left(\Delta\boldsymbol{\Theta}^{\top}\Delta\boldsymbol{\Theta}\right)^{-1}\Delta\boldsymbol{\Theta}^{\top}\Delta\boldsymbol{J}_{\pi_{\boldsymbol{\theta}}}\ . \tag{14}$$

In FDM, policy exploration is implemented on the parameter vector at the beginning of each trajectory. Perturbations can be chosen randomly; in this paper perturbations for each coordinate of $\boldsymbol{\theta}_j$ are drawn from a uniform distribution over $[-r, r]$, for some relatively small positive real number r.

B. Likelihood-ratio method (LRM)

In LRM, exploration is implemented directly on the actions, by using a randomized policy $\pi_{\boldsymbol{\theta}}(A|S) \in [0, 1]$, which takes action A in state S with probability $\pi_{\boldsymbol{\theta}}(A|S)$. Since A may consist of several simple actions, for each simple action $(l|L)$ for $l < L$, $l, L \in \{0, 1, \ldots, K_{\max}\}$, we define a randomized policy $\pi_{\boldsymbol{\theta}}((l|L)|S)$ as a sigmoid function with negative slope parameter η:
$$\pi_{\boldsymbol{\theta}}((l|L)|S) = \frac{1}{1 + e^{-\eta(\mathcal{T}(l|L) - C)}}\ .$$

Given cache contents $l_1 \leq \cdots \leq l_B$, and the B contents outside the cache with the largest remaining lifetimes $L_1 \geq \ldots \geq L_B$, we repeatedly try to perform the action $a^i \triangleq (l_i|L_i)$ with probability $\pi_{\boldsymbol{\theta}}(a^i|S)$ for $i = 1, \ldots, B$, until the first failure. This implies that for $b \leq B$, the probability of performing action $A^b = \{a^1, \ldots, a^b\}$ is
$$\pi_{\boldsymbol{\theta}}(A^b|S) = \left(1 - \pi_{\boldsymbol{\theta}}(a^{b+1}|S)\right)\prod_{i=1}^{b}\pi_{\boldsymbol{\theta}}(a^i|S),$$
where $\pi_{\boldsymbol{\theta}}(a^{b+1}|S)$ is defined to be zero for all states S.

Let $P_{\boldsymbol{\theta}}$ denote the density of an infinite trajectory $\tau = (S_1, A_1), (S_2, A_2), \ldots$ obtained by following policy $\pi_{\boldsymbol{\theta}}$, and let $J(\tau) = \limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mu(S_t, A_t)$. Then, under general, non-restrictive assumptions, we have
$$\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} = \int \nabla_{\boldsymbol{\theta}}P_{\boldsymbol{\theta}}(\tau)J_{\pi_{\boldsymbol{\theta}}}(\tau)d\tau\ .$$
Using the "likelihood-ratio" identity $\nabla_{\boldsymbol{\theta}}\log P_{\boldsymbol{\theta}}(\tau) = \nabla_{\boldsymbol{\theta}}P_{\boldsymbol{\theta}}(\tau)/P_{\boldsymbol{\theta}}(\tau)$, the above gradient can be expressed as
$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\rho^{\pi_{\boldsymbol{\theta}}} &= \int P_{\boldsymbol{\theta}}(\tau)\nabla_{\boldsymbol{\theta}}\log P_{\boldsymbol{\theta}}(\tau)J(\tau)d\tau \\ &= \mathbb{E}[\nabla_{\boldsymbol{\theta}}\log P_{\boldsymbol{\theta}}(\tau)J(\tau)]\ .\end{aligned} \tag{15}$$

The expectation with respect to $P_{\boldsymbol{\theta}}$ is approximated by sample averages over sample trajectories $\tau^{[i]}$ of finite length. Interestingly, this can be done without the knowledge of $P_{\boldsymbol{\theta}}$ [17]. Indeed, since $P_{\boldsymbol{\theta}}(\tau) = P(S_1)\prod_{t=1}^{T}P(S_{t+1}|S_t, A_t)\pi_{\boldsymbol{\theta}}(A_t|S_t)$, taking logarithm and differentiating with respect to $\boldsymbol{\theta}$ gives
$$\nabla_{\boldsymbol{\theta}}\log P_{\boldsymbol{\theta}}(\tau) = \sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(A_t|S_t), \tag{16}$$
which can be computed directly using the parametric form of $\pi_{\boldsymbol{\theta}}$. Thus, the expectation in (15) can be estimated by averaging over a number of independent trajectories sampled from policy $\pi_{\boldsymbol{\theta}}$. To minimize the variance of the estimate, we introduce a *baseline* vector $\boldsymbol{\beta}$, as in the REINFORCE algorithm [21], and estimate the hth coordinate of the gradient by
$$\nabla_{\boldsymbol{\theta}_h}\rho^{\pi_{\boldsymbol{\theta}}} = \mathbb{E}\left[\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}_h}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)(J(\tau) - \beta_h)\right].$$
The baseline does not introduce any bias in the gradient estimate: using (16), the likelihood-ratio identity, and the fact that $\int\nabla_{\boldsymbol{\theta}_h}P_{\boldsymbol{\theta}}(\tau)d\tau = 0$ since $\int P_{\boldsymbol{\theta}}(\tau)d\tau = 1$, we get
$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}_h}\log\pi_{\boldsymbol{\theta}}(A_t|S_t)\beta_h\right] &= \beta_h\int\nabla_{\boldsymbol{\theta}_h}P_{\boldsymbol{\theta}}(\tau)d\tau \\ &= \beta_h\nabla_{\boldsymbol{\theta}_h}\int P_{\boldsymbol{\theta}}(\tau)d\tau = 0.\end{aligned}$$
As in [21], we select the baseline β_h for $\nabla_{\boldsymbol{\theta}_h}\rho^{\pi_{\boldsymbol{\theta}}}$ by minimizing the variance of the estimate of the hth coordinate, which yields
$$\beta_h = \frac{\mathbb{E}\left[\left(\sum_{t=1}^{T}\nabla_{\theta_h}\log\pi_{\boldsymbol{\theta}}(A_t|S_t, C_t)\right)^2 J(\tau)\right]}{\mathbb{E}\left[\left(\sum_{t=1}^{T}\nabla_{\theta_h}\log\pi_{\boldsymbol{\theta}}(A_t|S_t, C_t)\right)^2\right]}\ .$$

Numerical results obtained with the proposed low-complexity caching algorithms optimized with both PG methods will be presented in Section VII. Next, we present lower bounds on the performance to evaluate the performance loss introduced by the proposed low-complexity caching policies.

VI. Lower Bounds

We present two lower bounds on the average cost: the first bound assumes infinite cache capacity while the second one assumes non-causal knowledge of the user access times.

A. Lower bound with unlimited cache capacity (LB-UC)

LB-UC is obtained by considering an unlimited cache capacity (i.e., $B = \infty$). In this case, there is no need to remove or replace any content inside the cache. The decision to download and store a content can be taken individually, and independently of the existing contents in the cache. The structure of the optimal threshold values for LB-UC follows from Corollary 2 and (11).

B. Lower bound with non-causal knowledge of the user access times (LB-NCK)

LB-NCK is obtained by assuming non-causal knowledge of the user access times. Since the user access times are known in advance, contents that will expire before the user accesses the OSN will never be downloaded and can be removed from the system. Therefore, there is no need to remove any content from the cache, thus $A_t^{(2)} = \emptyset$ when $E_t > 0$. All the remaining contents must be downloaded before the next user access, which means that there is no difference among contents, and so whenever the CM decides to download a content, it does not matter which one it is. As such, without loss of generality, we assume that the CM may cache only the first B contents generated after a user access. This means that there will always be space for these contents to be downloaded. It follows that the optimal policy only needs to decide when to download these contents, and this decision is independent of the contents in $\mathcal{O}$ and $\mathcal{I}$, and only depends on the time till the next user access and the current cost.

To determine the optimal policy, similarly to Corollary 2 and (11), the problem can be modeled as a finite-horizon MDP-SI, where the time horizon is dictated by the time G until the next user access. Denoting by V_G^{NCK} the average energy cost of downloading a content following the optimal policy; we have $V_0^{NCK} = \mathbb{E}[C]$, since the content must be downloaded when $G = 0$. For any $G \geq 1$, the dynamic programming equations imply that $V_G^{NCK} = \mathbb{E}\left[\min\{C, V_{G-1}^{NCK}\}\right]$. Therefore, the optimal decision is to download a content if the channel cost C is smaller than the future download cost V_{G-1}^{NCK}. Thus, the optimal policy again has a threshold structure.

Corollary 3. *Assuming that the user access times are known non-causally, there exist thresholds $C_{max} \geq \mathcal{T}_1^{NCK} \geq \cdots \geq \mathcal{T}_{D_{max}}^{NCK} \geq 0$[8] such that, for any $\mathcal{O}, \mathcal{I}$, if there are G time slots left until the next user access, a content with remaining lifetime $L \in \mathcal{O}$ (with $L \geq G$) is downloaded for $G \geq 0$ (i.e., when $U_t = 0$) if $|\mathcal{I}| < B$ and $C_t \leq \mathcal{T}_G$. The thresholds are given by the recursion $\mathcal{T}_1^{NCK} = \mathbb{E}[C]$ and for $G \geq 2$, $\mathcal{T}_G^{NCK} = \mathbb{E}\left[\min\{C, \mathcal{T}_{G-1}^{NCK}\}\right]$.*

[8]Note that D_{max} is the bound on the length of the user access interval, and can be infinite under the IRM model.

VII. Numerical Results

Here we present numerical simulations implementing the proposed caching schemes with both FDM and LRM. We compare their performances with the two lower bounds in Section VI, as well as with reactive and random caching schemes. Reactive caching does not utilize the cache, and all the relevant contents are downloaded at the time of user access. In *random caching*, when $U_t = 0$, each relevant content in $\mathcal{O}$ is downloaded randomly, with a constant probability $p_r > 0$ whenever $|\mathcal{I}_t| < B$, and $p_r = 0$ whenever $|\mathcal{I}_t| = B$. This scheme exploits the cache capacity, but does not utilize any intelligence in making the caching decisions. Note that random caching is equivalent to reactive caching when $p_r = 0$.

A. System Setup

The number of contents generated at each time slot, M_t, is drawn uniformly at random from the set $\{1, \ldots, M_{max}\}$, while the lifetime $K_{t,i}$ of individual contents $i \in \{1, \ldots, M_t\}$ at the time of generation is drawn from the set $\{5, 10, \ldots, K_{\max}\}$, where $K_{\max}$ is a multiple of 5. We assume that the user accesses the system independently at each time slot, with probability $p_a = 0.15$.

We obtain C_t 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. Using parameters consistent with the Long Term Evolution (LTE) network model [22], on a dB scale, the noise power is given by

$$P_{noise} = 10\log_{10}(kT) + 10\log_{10} W + NF,$$

where $kT = -174$ dBm/Hz is the noise power spectral density, and $NF = 5$ dB is a typical noise figure. We have

$$P_{signal} = C_t + G_{TX} + G_{RX} - PL(d),$$

where G_{TX} and G_{RX} are the transmit and receive antenna gains, respectively, and $PL(d)$ is the pathloss, which is a function of the distance d between the user and the serving BS. We adopt the 3GPP channel model [23], and consider an urban micro (UMi) system, with an hexagonal cell layout in the non-line-of-sight (NLOS) scenario, in which case

$$PL(d) = 36.7\log_{10}(d) + 22.7 + 26\log 10(f_c) + \mathcal{X}_\sigma,$$

where $f_c = 2.5GHz$ is the center frequency, and $\mathcal{X}_\sigma$ is the shadow fading parameter drawn from a zero-mean log-normal distribution with standard deviation $\sigma = 4$ dB. Distance is in meters (m), and the user location is assumed to be independent across time, and uniformly distributed within a cell.

We assume that a micro BS has a radius of 250m, and the shortest possible distance of a user from a serving BS is 50m. Therefore, the user distance d from the serving BS in any time slot is drawn from a uniform distribution $d \sim \mathcal{U}(50, 250)$. We

assume that the user is only served by a single BS in every time slot. Although we focus on a single user, the savings in energy will scale proportionally with the number of users. We use the values $G_{TX} = 17$ dBi and $G_{RX} = 0$ dBi. To compute the noise power, we assume a fixed (average) bandwidth of 10 MHz in every time slot, and for the Shannon capacity formula, we assume a spectral efficiency of $R/W = 2$ bps/Hz for each content item. The required power will be linearly scaled with the number of contents downloaded at each time slot, assuming they are independently encoded and transmitted over orthogonal subbands. For all the simulations, we set the initial state as $\mathcal{O}_0 = \mathcal{I}_0 = \emptyset$ and $E_0 = 0$. The cache capacity B is measured in number of contents.

For FDM, we choose the perturbation parameters $\Delta\theta$ from a uniform distribution $\Delta\theta \sim \mathcal{U}(-0.08, 0.08)$. For each iteration, a policy update is performed after 100 trajectories, with the duration of a trajectory set as 200 time slots. For LRM, for the randomized policy to closely resemble the actual deterministic policy, the logistic function defining the policy should be as close to a unit step function as possible. Hence, we set $\eta = 10$. A policy update is performed after only 20 trajectories, with the duration of a trajectory set again to 200 time slots.

For the initial parameter vector $\boldsymbol{\theta}_0$ of LISO, we use the threshold values obtained from the unlimited cache capacity problem (as the initial $\theta(0, L)$ values for all L). Also for LFA, we use the same parameter vector as the initial values $\theta_i(l, L)$, for all l,L, and $i \in \{0, 1, \ldots, K_{\max}\}$. This provides a relatively good initial point, thus improving the convergence speed over a random initial parameter vector. For all the algorithms, an average of 5 policy updates is taken as the policy update of any iteration. In each simulation setup, we select an appropriate step size by adjusting the step size at different runs until the best result is obtained. Finally, to test the performance of any policy, we use a *test data* of 100 trajectories, each consisting of 5000 time slots.

B. Performance Evaluation

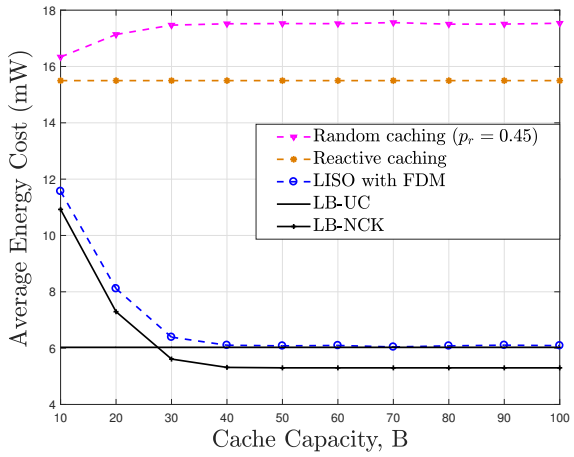
To simplify the presentation, we first compare LISO implemented with the FDM algorithm with the benchmarks in Fig. 2a, where we plot the average energy cost with respect to the cache capacity. We set $p_r = 0.45$ for the random caching policy to download contents into the cache. We observe that the random caching policy has the highest average cost (which increases with p_r). Naturally, the average cost of the reactive scheme is independent of the cache capacity as it does not utilize the cache. While the reactive scheme only downloads contents that are actually requested, the random scheme downloads many contents that will eventually expire before being requested. The performance of LB-NCK decreases with the cache capacity. This is because more contents that will remain relevant by the user access time can be downloaded at favorable channel conditions through proactive caching into a larger cache. Not surprisingly, the performance of LB-NCK meets that of the reactive scheme when $B = 0$.

LISO significantly improves the system's performance with respect to reactive caching for any nonzero cache capacity. For a cache capacity of $B = 30$, LISO achieves approximately 60% reduction in energy consumption over the reactive scheme. For relatively large cache capacities, i.e., $B \geq 40$, the performance of LISO almost meets that of LB-UC. This is because more contents that will not expire by the next user access can be stored in the cache, and almost no contents need to be removed. This means that a cache capacity of $B = 40$ is sufficient to provide all the potential gains from proactive caching in this setup. Interestingly, 40 is roughly the average number of relevant contents at any point in time. Moreover, in the low cache-capacity regime, the performance of LISO is very close to that of LB-NCK. This is because when the cache capacity is small, the system is relatively conservative in proactively caching contents, and so downloaded contents rarely expire or are swapped before the next user access. Thus, the gain from knowing the user access times is limited. We conclude from Fig. 2a that any improvement in the performance of LISO implemented with FDM can occur only for low cache capacity values ($B \leq 30$).

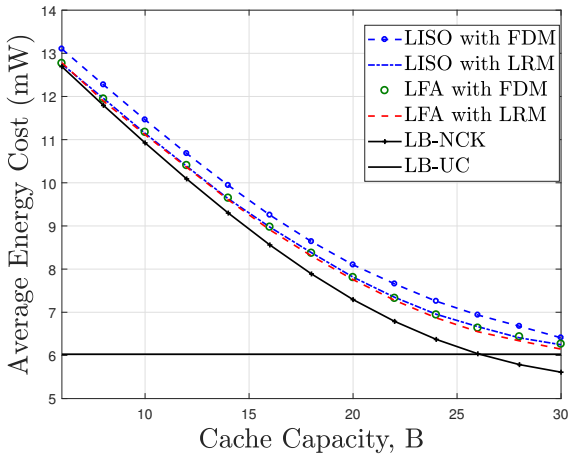
In Fig. 2b we plot the performance of both caching schemes, LISO and LFA, implemented with both FDM and LRM algorithms, in the low cache capacity regime. We observe that both the policy representation using LFA and using the LRM algorithm for gradient estimation improve the performance compared to LISO with FDM. At very low cache capacities, i.e., $B < 10$, the performances of LISO with FDM and LFA with FDM or LRM all closely follow the LB-NCK bound. Meanwhile, the LFA policy has a performance gain of up to 4.4% over the LISO policy when both schemes are implemented with the FDM algorithm. This performance gain can be attributed to the fact that the LFA policy considers the remaining lifetimes of all the contents inside the cache when making a caching decision, which is ignored by LISO. When LFA is implemented with LRM, it achieves a performance gain of up to 5.6% over LISO implemented with FDM. LRM also improves the performance of LISO with FDM up to 4.2%. We can attribute the better performance of LRM to its improved exploration strategy.

In Fig. 3 we plot the average energy cost with respect to the maximum lifetime of contents, $K_{\max}$. We observe that the energy cost increases with the lifetime of contents. This is expected: when the contents remain relevant longer, more contents will be consumed by the user at the time of access. We can also observe that LFA outperforms LISO for all $K_{\max}$ values considered. The performance gain of LISO with LRM with respect to LISO with FDM increases with $K_{\max}$, which means that a better exploration strategy becomes more important as $K_{\max}$ increases, since the cache space becomes relatively more limited per relevant content. In Fig. 4, we compare our schemes with the bounds. We observe that they perform better than reactive caching for all values of K_{max}, with up to 50% performance gain at $K_{max} = 20$, and perform close to LB-UC at relatively small values of K_{max}, and close to LB-NCK at relatively high values of K_{max}.

In Fig. 5, we compare the convergence rates of the two PG methods. Initially, LRM performs worse than FDM, but after about 250 trajectories, LRM starts to converge at a faster rate, saturating at the best performance after approximately 1000 trajectories. LRM is known to have better theoretical convergence guarantees, and its superiority over FDM has



(a)



(b)

Fig. 2: Average energy cost vs. cache capacity with $K_{\max} = 15, M_{max} = 8, p_a = 0.15$.

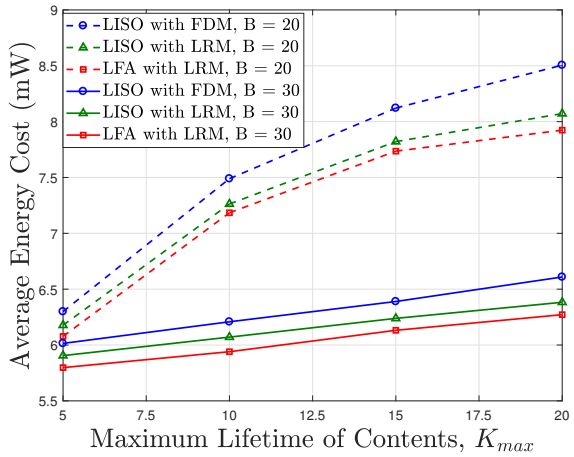


Fig. 3: Average energy cost vs. maximum lifetime of contents for $B = 20, 30$, when $M_{max} = 8, p_a = 0.15$.

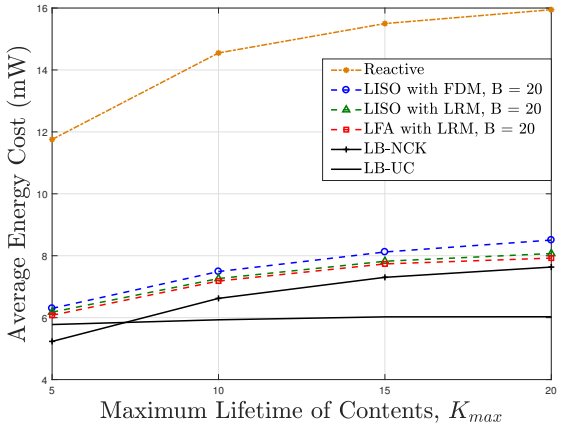


Fig. 4: Average energy cost vs. maximum lifetime of contents for $B = 20$, when $M_{max} = 8, p_a = 0.15$.

Algorithm	Step size	Number of trajectories	Runtime (secs)
LISO-FDM	0.1	18000	791.28
LFA-FDM	0.01	18000	8246.38
LISO-LRM	0.0005	1000	79.07
LFA-LRM	0.00005	1000	1556.64

TABLE I: Setup and running time of the different algorithms.

been observed in other applications as well [24]. Table I shows the relationship between the policy representations and the PG methods in terms of how fast they converge using the best parametrization. Note that the runtime includes the time it takes to run the algorithms and to test the performance after each update to monitor convergence. We have run the simulations on an Intel Core i7 − 7700K CPU with 4.2GHz processor speed. We observe that, despite the fact that we use larger values of step size for FDM, it converges at a lower rate than LRM. We also observe that, as expected, the larger the parameter space, the longer it takes to run the algorithm.

Fig. 6 shows the average energy cost with respect to the maximum number of contents generated, $M_{\max}$. Similarly to K_{max} the energy cost increases with the number of contents. The performance gain of LISO and LFA, implemented with FDM and LRM, respectively, closely follow the LB-UC bound when $M_{max} \leq 5$ for $B = 20$ and $M_{max} \leq 8$ for $B = 30$, and the LB-NCK bound when $M_{max} > 5$ for $B = 20$ and $M_{max} > 8$ for $B = 30$.

Fig. 7 shows the average energy cost with respect to the probability of user access, p_a. We can observe that the average energy cost increases with p_a since more contents will be consumed by the user. Furthermore, the gap between LISO and LB-UC diminishes as p_a increases. This is because the relative size of the cache increases as the user accesses the system more often (e.g., no cache is needed in the extreme case of $p_a = 1$).

In the next section, we go beyond our modeling assumptions to show that the performance gain of the LFA policy with respect to LISO can be more significant if the underlying stochastic processes have memory.

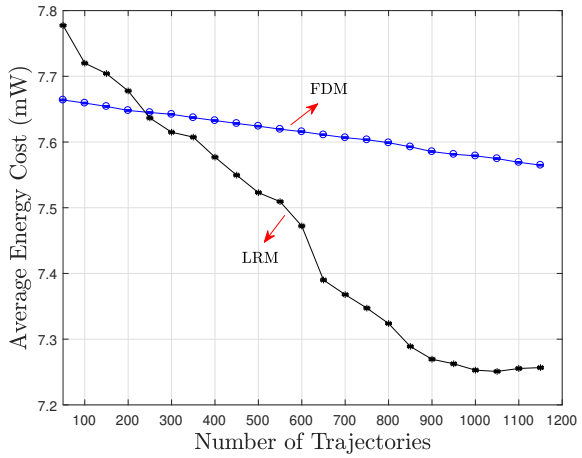


Fig. 5: The evolution of the FDM and the LRM algorithms with respect to the number of trajectories.

C. System with Memory

Here we introduce temporal memory in the generation of content lifetimes and the user location, i.e., distance from the serving BS, which is a more realistic model for a mobile user served by micro BSs. For the lifetime process, we assume that the content generator has two states, called "short" and "long" content states, respectively. When it is in the short content state, all the generated contents have an initial lifetime of 5, whereas in the long content state, all the contents are generated with a lifetime of 15. The content generator transitions from one state to the other randomly. We assume that, if it is in the short content state, it remains there with probability p_1, while it remains in the long content state with probability p_2.

We assume that at each time slot the user moves either towards or away from the serving BS. The distance from the serving base station at time slot $t+1$ is $d_{t+1} = d_t \pm \sigma$, where σ is a positive constant, which describes how fast the user is moving. The corresponding user location model is a Markov process where the state transition probabilities are given by $P(d_{t+1} = d_t + \sigma) = p_u$ and $P(d_{t+1} = d_t - \sigma) = 1 - p_u$, for all t. We further impose that the distance d is bounded between 50m and 250m, so only a single direction of movement is possible on the boundary points.

Note that the threshold structure of the optimal policy no longer holds in this model with memory; however, we can still evaluate the performances of the proposed caching schemes which exploit a threshold structure. Figure 8 shows the performances of the LFA and LISO policies, both implemented with the LRM algorithm, for values of $p_1 \in \{0.1, 0.5, 0.9\}$ and varying p_2 from 0.1 to 0.9. Note that, higher p_1 and p_2 values mean that the system is more likely to stay in the same state, and continue to generate contents with the same lifetime. The results are obtained for a cache capacity of $B = 20$. For the user location process, we set $\sigma = 5$ and $p_u = 0.5$. We observe that the average energy cost increases with increasing p_2 and with decreasing p_1, as they both lead to the generation of more contents with lifetime 15. We observe similar trends for the gain of LFA with respect to LISO; that is, the improvement with respect to LISO also increases with p_2 and decreases with p_1.

We observe that the performance gain of LFA over LISO is more significant than the memoryless scenario. For similar system parameters in the memoryless case; that is, for a cache capacity of $B = 20$, and assuming that the LRM algorithm is used, LFA policy has a performance gain of approximately 0.75% over LISO. However, when memory is introduced, LFA can provide a performance gain of approximately 2%. We note that, when the lifetime generation has memory, existing contents in the system provide more information about the future states; and hence, the LFA policy, which takes into account the remaining lifetimes of all the contents, provides higher gains.

VIII. Conclusions

We have considered the proactive caching problem in wireless networks with the aim of minimizing the long term average energy cost of delivering contents to the UE over a time-varying wireless link under random user accesses to the system, random content lifetime, and a time-varying library size. We have first showed the optimality of a threshold-based policy, which pushes contents to the cache (or may remove contents from the cache if it is full) depending on the relative value of the channel state with respect to preset threshold values that depend on the time elapsed since last user access and the remaining lifetimes of all the relevant contents in the system. Since this leads to a prohibitively large set of parameters to be optimized, we have proposed two suboptimal caching schemes, LISO and LFA, that are based on low-complexity parametrization of the system states and policy search techniques from reinforcement learning. We have further introduced two lower bounds on the performance, and through numerical simulations, we have showed that the two low-complexity proactive caching schemes perform close to optimal, with LFA performing better than LISO in general. Proactive caching under nonlinear cost functions, and in multi-user scenarios are currently being considered as interesting future extensions of this paper.

Appendix A
Proof of Lemma 1

We start the proof by showing that our MDP $(\mathcal{S}, \mathcal{A}, P, \mu)$ satisfies (9) when P^π is ergodic for any policy π and $\mathcal{Z}$ is an interval. First note that Theorems 5.1–5.3 of [25] imply that for any MDP with a countable state space and whose action space is a compact metric space, there exists an optimal deterministic policy satisfying (9). Clearly, under our assumptions, $\mathcal{S}$ is countable. Furthermore, since $g \in \mathcal{A}$ is Borel-measurable, any limit point (under pointwise convergence) of a sequence of functions from $\mathcal{A}$ also belongs to $\mathcal{A}$ (i.e., it is a Borel-measurable function). On the other hand, the representation of policies with functions from $\mathcal{A}$ is not unique, since any two functions $g, g' \in \mathcal{A}$ such that $\mathbb{P}[g(Z) = g'(Z)] = 1$ represent the same policy (up to a zero-measure event), and this causes problems in establishing the compactness of $\mathcal{A}$.

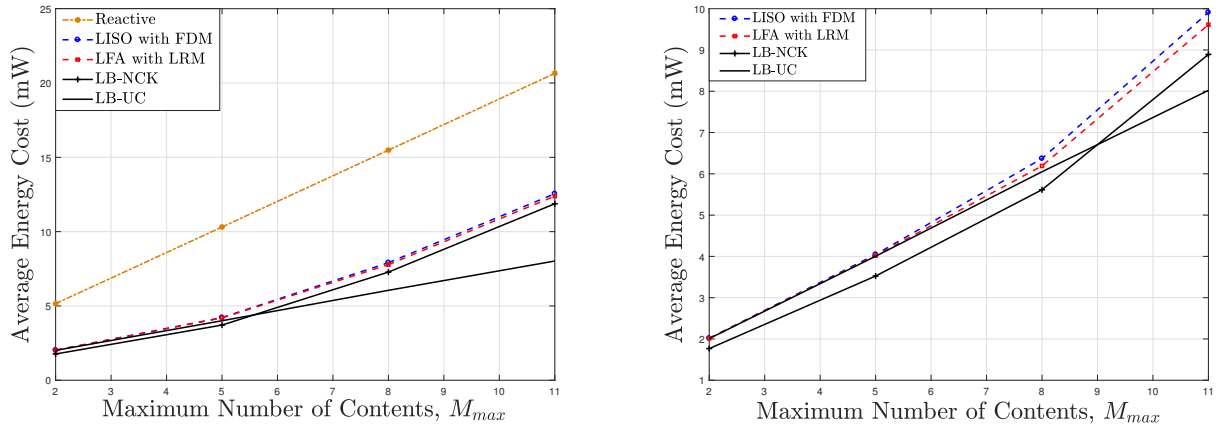


Fig. 6: Average energy cost vs. maximum number of contents for $B = 20$ (left) and $B = 30$ (right), when $K_{max} = 15, p_a = 0.15$.

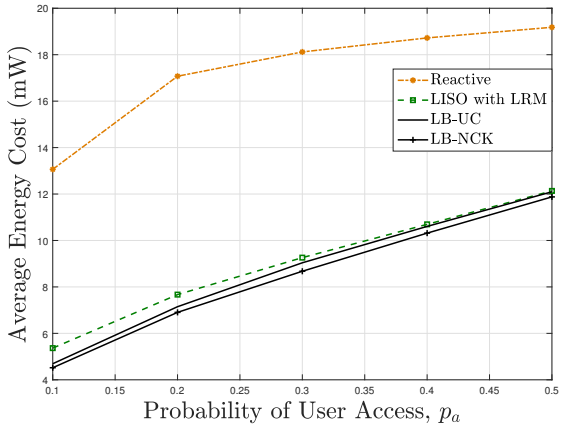


Fig. 7: Average energy cost vs. probability of user access for $B = 30$, when $K_{max} = 15, M_{max} = 8$.

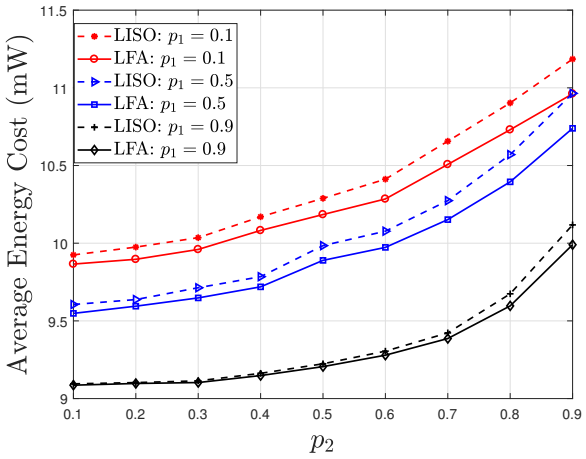


Fig. 8: Average energy cost vs. transition probabilities p_1 and p_2 for both LFA and LISO policies, with cache capacity $B = 20$, $\sigma = 5$ and $p_u = 0.5$.

To alleviate this problem, for any $g \in \mathcal{A}$, define the equivalence class $\mathcal{G}_g = \{g' \in \mathcal{G} : \mathbb{P}[g(Z) = g'(Z)] = 1\}$, and let $\mathcal{G} = \{G_g : g \in \mathcal{A}\}$ denote the family of these classes. For any $G \in \mathcal{G}$, let $f_G \in G$ be a selected element of G. Then, since each function f_G can take values only in the finite set $\mathcal{A}_{SI}$, with a slight modification to the proof of Theorem 3 in [26], one can show that the set $\bar{\mathcal{A}} = \{f_G : G \in \mathcal{G}\}$ is a compact metric space for the metric $\mathbb{P}[g(Z) \neq g'(Z)]$.

Consequently, the new MDP $(\mathcal{S}, \bar{\mathcal{A}}, P, \mu)$ satisfies (9). Furthermore, it is easy to see that the new MDP is equivalent to the original one in the sense that their trajectories are equal with probability one if any action $g \in \mathcal{A}$ in the original MDP is replaced with f_{G_g} in the new one. Therefore, the original MDP also satisfies (9).

Using (5) and (6), we can express (9) as

$$V^{\pi^*}(s) = \min_{g \in \mathcal{A}} \left\{ \mathbb{E}\left[\mu_{SI}(s, g(Z), Z) - \rho^{\pi^*} + \sum_{s' \in \mathcal{S}} P_{SI}(s'|s, g(Z)) V^{\pi^*}(s') \right] \right\}. \tag{17}$$

Since g is a mapping from $\mathcal{Z}$, the above minimum can be realized by minimizing for each value of the side information Z independently. Indeed, for any s, the minimum in (17) is achieved by any g satisfying

$$g(z) \in \operatorname*{argmin}_{a_{SI} \in \mathcal{A}_{SI,s}} \left\{ \mu_{SI}(s, a_{SI}, z) - \rho^{\pi^*} + \sum_{s' \in \mathcal{S}} P(s'|s, a_{SI}) V^{\pi^*}(s') \right\}. \tag{18}$$

Since the right hand side of (18) is a minimum of finitely many linear functions, it follows that $g(z)$ can be chosen to be a piecewise constant function: a piecewise constant function over the interval $\mathcal{Z}$ is defined by an interval partition $\mathcal{Z}_1, \ldots, \mathcal{Z}_m$ of $\mathcal{Z}$ (for some m) and some actions $a_1, \ldots, a_m \in \mathcal{A}_{SI}$ such that $g(z) = a_i$ if $z \in \mathcal{Z}_i, i = 1, \ldots, m$. Converting this policy back to the original MDP-SI problem via $\pi^*(s, z) = g(z)$ finishes the proof.

REFERENCES

[1] S. O. Somuyiwa, A. György, and D. Gündüz, "Energy-efficient wireless content delivery with proactive caching," in *Content Caching and Delivery in Wireless Nets. Work., (CCDWN)*, May 2017.

[2] ——, "Improved policy representation and policy search for proactive content caching in wireless networks," in *Int'l Symp. on Modeling and Optim. Mobile, Ad Hoc, and Wireless Nets.*, May 2017, pp. 1–8.

[3] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *IEEE INFOCOM*, March 2010, pp. 1–9.

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

[5] 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.

[6] 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.

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

[8] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[9] B. N. Bharath, K. G. Nagananda, D. Gunduz, and H. V. Poor, "Learning-based content caching with time-varying popularity profiles," in *IEEE Global Communications Conference*, Dec. 2017, pp. 1–6.

[10] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. on Wireless Comms.*, vol. 15, no. 9, pp. 6284–6297, Sept 2016.

[11] A. C. Güngör and D. Gündüz, "Proactive wireless caching at mobile user devices for energy efficiency," in *IEEE IntâĂŹl Symp. on Wireless Comm. Systems (ISWCS)*, Aug. 2015, pp. 186–190.

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

[13] W. Chen and H. V. Poor, "Joint pushing and caching with a finite receiver buffer: Optimal policies and throughput analysis," 2016. [Online]. Available: http://arxiv.org/abs/1602.04500

[14] 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.

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

[16] N. Zhang, J. Guan, C. Xu, and H. Zhang, "A dynamic social content caching under user mobility pattern," in *Int'l Wireless Comms. Mobile Comp. Conf.*, Aug 2014, pp. 1136–1141.

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

[18] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE INFOCOM*, Mar. 2012, pp. 1107–1115.

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

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

[21] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Machine Learning*, 1992, pp. 229–256.

[22] S. Stefania, T. Issam, and B. Matthew, *LTE, The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2011.

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

[24] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Int'l Conf. Intelligent Robots and Sys. (IROS)*, 2006.

[25] A. Arapostathis, V. Borkar, E. F. Gaucherand, M. Ghosh, and S. Marcus, "Discrete-time controlled Markov processes with average cost criterion: A survey," *SIAM Journal Control and Optim.*, vol. 31, no. 2, pp. 282–344, 1993.

[26] A. György and T. Linder, "On the structure of optimal entropy-constrained scalar quantizers," *IEEE Transactions on Information Theory*, vol. IT-48, no. 2, pp. 416–427, Feb. 2002.

Samuel O. Somuyiwa received the B.Sc. degree in Electronic and Electrical Engineering from Obafemi Awolowo University, Nigeria, in 2011, and the M.Sc. degree in Communications and Signal Processing from Imperial College London, UK, in 2014, where he is currently pursuing the Ph.D. degree. His research interest is in the application of machine learning for optimization of wireless networks.

He received the Presidential Special Scholarship Scheme for Innovation and Development award from the Federal Government of Nigeria in 2013.

András György received the M.Sc. (Eng.) degree (with distinction) in technical informatics from the Technical University of Budapest, in 1999, the M.Sc. (Eng.) degree in mathematics and engineering from Queen's University, Kingston, ON, Canada, in 2001, and the Ph.D. degree in technical informatics from the Budapest University of Technology and Economics in 2003. He was a Visiting Research Scholar in the Department of Electrical and Computer Engineering, University of California, San Diego, USA, in spring of 1998. In 2002-2011 he was with the Computer and Automation Research Institute of the Hungarian Academy of Sciences, where, from 2006, he was a Senior Researcher and Head of the Machine Learning Research Group. In 2003-2004 he was also a NATO Science Fellow in the Department of Mathematics and Statistics, Queen's University. He also held a part-time research position at GusGus Capital Llc., Budapest, Hungary, in 2006-2011. In 2012-2015, he was a researcher in the Department of Computing Science, University of Alberta, Edmonton, AB, Canada. In 2015 he joined the Department of Electrical and Electronic Engineering of Imperial College London, London, UK, where he is currently a Senior Lecturer. Since 2018 he has also been a Research Scientist at DeepMind, London, UK. His research interests include machine learning, statistical learning theory, online learning, adaptive systems, information theory, and optimization.

Dr. György received the Gyula Farkas prize of the János Bolyai Mathematical Society in 2001 and the Academic Golden Ring of the President of the Hungarian Republic in 2003.

Deniz Gündüz (S'03-M'08-SM'13) received his M.S. and Ph.D. degrees in electrical engineering from the NYU Polytechnic School of Engineering in 2004 and 2007, respectively. After his PhD, he served as a postdoctoral research associate at Princeton University, as a consulting assistant professor at Stanford University, and as a research associate at CTTC (Spain). In September 2012, he joined the Electrical and Electronic Engineering Department of Imperial College London, UK, where he is currently a Reader in information theory and communications. His research interests lie in the areas of communications and information theory, privacy and security in cyber-physical systems, and machine learning.

Dr. Gündüz is an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS, and the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He is the recipient of the IEEE Communications Society - Communication Theory Technical Committee (CTTC) Early Achievement Award in 2017, Starting Grant of the European Research Council in 2016, and the IEEE Communications Society Best Young Researcher Award for the Europe, Middle East, and Africa Region in 2014. He coauthored papers that received a Best Paper Award at the 2016 IEEE WCNC, and Best Student Paper Awards at 2007 IEEE ISIT and 2018 IEEE WCNC.