Improved Policy Representation and Policy Search for Proactive Content Caching in Wireless Networks

Samuel O. Somuyiwa, András György and Deniz Gündüz
Department of Electrical and Electronic Engineering
Imperial College London
Email: {samuel.somuyiwa12, a.gyorgy, d.gunduz}@imperial.ac.uk

***Abstract*—We study the problem of proactively pushing contents into a finite capacity cache memory of a user equipment in order to reduce the long-term average energy consumption in a wireless network. We consider an online social network (OSN) framework, in which new contents are generated over time and each content remains relevant to the user for a random time period, called the *lifetime* of the content. The user accesses the OSN through a wireless network at random time instants to download and consume all the relevant contents. Downloading contents has an energy cost that depends on the channel state and the number of downloaded contents. Our aim is to reduce the long-term average energy consumption by proactively caching contents at favorable channel conditions.**

In previous work, it was shown that the optimal caching policy is infeasible to compute (even with the complete knowledge of a stochastic model describing the system), and a simple family of threshold policies was introduced and optimised using the finite difference method. In this paper we improve upon both components of this approach: we use linear function approximation (LFA) to better approximate the considered family of caching policies, and apply the REINFORCE algorithm to optimise its parameters. Numerical simulations show that the new approach provides reduction in both the average energy cost and the running time for policy optimisation.

I. Introduction

Caching popular contents at the edge of a wireless network, i.e., at base stations (BSs) and/or directly at user equipments (UEs), has received a lot of recent attention as a means to improve quality of user experience in wireless content delivery [1]–[5]. In the traditional, so-called *reactive* content delivery paradigm, contents are delivered to the network edge from a content delivery network (CDN) server through the back-haul links at the time of request. However, the request time may be an unfavorable time, for example, a peak traffic period or a period during which the channel conditions are poor. Instead, prerecorded contents, such as video-on-demand or video contents available through content providers such as YouTube, Hulu or BBC iPlayer, which constitute a significant portion of the videos downloaded over the Internet, can be delivered *proactively* over favourable time periods, and stored in a cache memory at the network edge. This approach reduces both the energy consumption and latency in content delivery.

Content caching in unknown or dynamic settings have been studied in several works in the literature [6]–[11]. The goal in these problems is to learn to manage the cache contents, i.e., which content to cache, when to cache, etc., in order to maximize a particular performance measure over a specified time horizon. In [6], [7], collaborative filtering and transfer learning techniques are used to learn the optimal cache contents based on the estimated content popularity, traffic load, back-haul and storage capacities. In [8]–[10] the problem of learning the optimal cache contents and optimal distribution strategies are modeled as a multi-armed bandit problem. Other works include exploiting device-to-device communications to maximize throughput with caching [12]–[14], exploiting the broadcast nature of wireless communications to improve caching gain through coded caching [1], [15], and to reduce the energy consumption by proactive caching [16]. Proactive content caching has also been studied from an energy efficiency perspective in [17] and [18] under a deterministic model; that is, assuming that the user demands and the channel conditions are known in advance, for point-to-point and device-to-device scenarios, respectively.

In this paper we consider proactive content caching for a single user in an online social network (OSN) framework, where contents are generated randomly by the user's social connections, and are added to the user's *news feed*. Each content remains *relevant* to the user for a random period of time, which we refer to as *lifetime*. We assume that the user is not interested in a content whose lifetime has expired. At random time instants, the user accesses the OSN, say, through an app on her mobile device and requests all the relevant contents; that is, contents generated by the user's social connections whose lifetime has not expired at the time of access. All the relevant contents must be delivered to the application layer of the UE at the time of user request.

When contents are delivered through the wireless link, the serving BS incurs a transmission energy cost that depends on the actual channel condition and on the number of contents delivered[1]. The channel condition depends on user mobility, environmental and traffic conditions, and we assume that it is independently and identically distributed (i.i.d.) across time. A cache manager (CM) is responsible for downloading and removing contents to and from the UE. We assume that the CM has a causal knowledge of all the underlying statistical distributions, i.e, the number and lifetimes of new contents generated at every time period, user request interval and

[1]The energy cost of the BS can be replaced by other metrics, such as the delay of delivering all the contents to the UE, the channel resources required for successful downloading of the contents, or the energy used for decoding the messages at the UE.

channel conditions. It decides whether or not to proactively push (or remove) contents to (from) the finite-capacity cache memory whenever there is no user request. Contents that have been proactively cached into the UE cache memory do not need to be downloaded at the time of user access, and are moved from the cache to the application layer without incurring any additional cost.

The key challenge addressed in this paper is to find the caching policy which minimizes the long term average energy consumption. While contents can be proactively cached over favorable channel conditions, this also has the risk of downloading, and incurring the associated cost, contents that would never be requested by the user, if the user does not access the OSN within their lifetimes. A caching policy decides which relevant contents should be pushed to the cache memory, and which should be removed. We model this scenario as a Markov decision process (MDP). It is possible to prove the optimality of a threshold-based policy [19], which may swap contents in the cache with shorter remaining lifetime with those that are not in the cache, but have longer remaining lifetime, according to some pre-fixed threshold values on the channel quality. Due to the size of the state and action spaces, and the corresponding number of thresholds, which is exponential in the maximum lifetime of contents (denoted by K_{max}), characterizing the optimal threshold values; and hence, the optimal policy, is computationally infeasible. Instead, in [19], we proposed a suboptimal policy called *longest lifetime in–shortest lifetime out* (LISO), which has a single threshold value for every pair of lifetimes and time elapsed since the last user access, and swaps the content with the shortest remaining lifetime in the cache memory with the content with the longest remaining lifetime among the relevant files that are not in the cache, if the channel cost is below the corresponding threshold. The policy has $O(K_{max}^2)$ threshold parameters, and we used the finite difference method (FDM) for policy search to obtain those threshold values. In this paper, we improve the performance of the LISO policy by using the REINFORCE algorithm [20] for policy search, and by using a linear function approximation with $O(K_{max}^3)$ parameters to represent the threshold values. Simulations show an improvement in the performance, in terms of both the running time for policy optimisation and in the long term average energy consumption. We also observe that the energy consumption is very close to the lower bound we have derived.

The rest of the paper is organised as follows. The system model is introduced in Section II. The general behavior of the optimal threshold-based policy is explained in Section III. The new parametrisation of the threshold-based policy, and two new policy search methods that use LFA are presented in Section IV. Performance lower bounds are presented in Section V. Section VI is dedicated to the presentation of the simulation setup and the corresponding numerical results, and further discussions. Finally, Section VII concludes the paper.

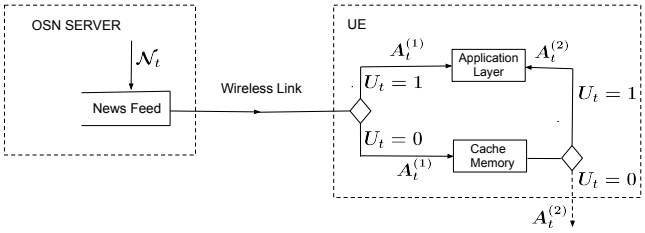

Fig. 1. Illustration of the system model: $\mathcal{N}_t$ is the set of newly generated contents, U_t is the user state, and $A_t^{(1)}$ and $A_t^{(2)}$ are the set of actions taken in time slot t. The dashed line from the cache (when $U_t = 0$) illustrates the contents *removed* from the cache.

II. System Model

Consider a mobile user equipped with a cache memory of finite capacity B, which can be used to proactively download and store contents that are generated in an OSN. We assume that time is slotted in equal length intervals, and at the beginning of each slot a random number of equal-size contents are generated by the user's social connections, and are added to the user's *news feed*. We denote the set of contents generated at the beginning of time slot t by $\mathcal{N}_t$, and their number by M_t, which is an i.i.d. sequence with generic random variable M. Each content remains relevant to the user for a random lifetime after which it is removed from the news feed and the cache memory (if it has already been downloaded). We denote the lifetime of the ith content generated in time slot t by $K_{t,i}$, which is also i.i.d. with generic random variable K. We denote the set of contents not in the cache at time slot t by $\mathcal{O}_t$, and the set of contents in the cache memory by $\mathcal{I}_t$; note that $|\mathcal{I}_t| \leq B$. A cache manager (CM), which can be placed either at the BS or UE depending on the protocol employed, decides which contents to download from $\mathcal{O}_t$ and which to remove from $\mathcal{I}_t$ at each time slot. We denote the set of contents downloaded in time slot t by $A_t^{(1)}$, and the set of contents removed from the cache by $A_t^{(2)}$. We have $A_t^{(1)} \subseteq \mathcal{O}_t$ and $A_t^{(2)} \subseteq \mathcal{I}_t$.

The user's access to the OSN is also random, and has a binary representation where $U_t = 0$ means that the user does not access the system in time slot t, while $U_t = 1$ denotes user access. The user access sequence $\{U_t\}$ is assumed to be an arrival process with i.i.d. inter-arrival times D_n, where $D \geq 1$ is a random variable with generic distribution of inter-arrival times. Whenever the user accesses the OSN, all the relevant contents must be made available to the user for consumption, that is, all contents in $\mathcal{O}_t$ must be downloaded, and together with the contents already in the cache, $\mathcal{I}_t$, they are moved to the application layer. The block diagram in Figure 1 illustrates the system model and shows the behavior of the CM with respect to the user behavior. Both the cache and the server are reset after a user access, that is, if $U_t = 1$ for some time slot t, then we have $\mathcal{O}_{t+1} = \mathcal{N}_{t+1}$ and $\mathcal{I}_{t+1} = \emptyset$.

Without any loss of generality, because of the homogeneity assumption on the size of the contents, we can represent the sets $\mathcal{N}_t, \mathcal{O}_t, \mathcal{I}_t, A_t^{(1)}$, and $A_t^{(2)}$ as multi-sets of remaining

TABLE I

TABLE OF ACTION SETS, AND THE STATE TRANSITIONS OF THE CONTENTS IN AND OUT OF THE CACHE MEMORY FOR GIVEN USER BEHAVIOR IN TIME SLOT t.

$U_t = 0$	$U_t = 1$
$A_t^{(1)} \subset \mathcal{O}_t$	$A_t^{(1)} = \mathcal{O}_t$
$A_t^{(2)} \subset \mathcal{I}_t$	$A_t^{(2)} = \mathcal{I}_t$
$\mathcal{O}_{t+1} = \left((\mathcal{O}_t \cup A_t^{(2)} \setminus A_t^{(1)}) - 1 \right) \cup \mathcal{N}_{t+1}$	$\mathcal{O}_{t+1} = \mathcal{N}_{t+1}$
$\mathcal{I}_{t+1} = (\mathcal{I}_t \cup A_t^{(1)} \setminus A_t^{(2)}) - 1$	$\mathcal{I}_{t+1} = \emptyset$

lifetimes (positive integers, with the set of all positive integer tuples denoted by $\mathbb{N}^*$). For any multi-set Y with positive elements, let $Y-1 = \{y > 0 : y+1 \in Y\}$ denote the multi-set obtained by reducing each element of Y by 1 and removing the zeros. With this definition, the evolution of the set of contents in and out of the cache ($\mathcal{I}_t$ and $\mathcal{O}_t$, respectively), as well as some constraints on the set of contents to be downloaded to and the set to be removed from the cache ($A_t^{(1)}$ and $A_t^{(2)}$, resp.) is given in Table I.

The wireless link is subject to random variations, causing the serving BS to incur a random energy cost $C_t \geq 0$ for each content downloaded in slot t. We assume that C_t is i.i.d. across time, and follows a continuous random variable with cumulative distribution function (cdf) $F_C(c)$.

The average energy cost after T time slots is

$$J_T = \frac{1}{T} \sum_{t=1}^{T} C_t |A_t^{(1)}|,$$

and the objective of the CM is to minimize the long term expected average cost $\lim_{T\to\infty} \mathbb{E}[J_T]$. Finally, we assume that the random variables M, K, D and C are bounded by $M_{max}, K_{max}, D_{max} \in \mathbb{N}$, $C_{max} \in \mathbb{R}^+$, the variables $\{M_t\}, \{K_{t,i}\}, \{C_t\}, \{D_n\}$ are independent, and that the CM knows all the underlying statistical distributions involved.

III. THE OPTIMAL THRESHOLD POLICY

In a discrete time finite-state finite-action MDP [21], a controller takes an action in every time step, and depending on the current state and the action taken, the system moves to a new state and the controller suffers some cost. The 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 transition probability function, and $\mu : \mathcal{S} \times \mathcal{A} \to [0, \mu_{max}]$ is a cost function with some $\mu_{max} > 0$. Assuming the system is in state $s \in \mathcal{S}$, the controller selects an action $a \in \mathcal{A}$, and as a result, the system moves to a new state s' with probability $P(s'|s, a)$, and the controller incurs a cost $\mu(s, a)$. The goal of the controller is to minimize its long term 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 of the controller is a mapping $\pi : \mathcal{S} \to \mathcal{A}$, selecting an action for each state. Let Π denote the set of all deterministic policies. For any policy $\pi \in \Pi$ 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, and assuming that $\sum_{s'\in\mathcal{S}} P(s'|s, a) = 1$ for all $s \in \mathcal{S}, a \in \mathcal{A}$, let ρ^π denote the average cost ρ if $a = \pi(s),\ \forall s \in \mathcal{S}$, then there exists (e.g., in [21]) a deterministic policy π^* such that, $\rho^{\pi^*} = \min \rho$, with the minimum taken over all possible actions in $\mathcal{A}$, and a differential value function for any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, which is defined as

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

Furthermore, the optimal policy π^* satisfies

$$V^{\pi^*}(s) = \min_{a\in\mathcal{A}} \left\{ \mu(s, a) + \sum_{s'\in\mathcal{S}} P^{\pi^*}(s'|s, a) V^{\pi^*}(s') \right\}, \quad (1)$$

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

A. Optimal Cache Management Policy

For the problem of minimizing the long term expected average energy cost with proactive caching, we can extend the MDP model to include a side information, such that, at the end of time slot t, the channel state C_t is regarded as an i.i.d side information, and the state of the MDP S_t is defined by the set of contents in the news feed $\mathcal{O}_t$, the set of contents in the cache $\mathcal{I}_t$, with elements of both sets represented by the remaining lifetimes of the relevant contents, and the elapsed time since the last user access E_t, where, assuming that the user accesses the OSN at time $t = 0$ (i.e., we set $U_0 = 1$); E_t is defined as $E_t \triangleq \min\{t-n : 0 \leq n \leq t, U_n = 1\}$. The set of all possible combinations of $\mathcal{O}_t, \mathcal{I}_t$ and E_t is $\mathcal{S} \subset \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}$. The action taken by the CM in every time slot is the pair $A_t = (A_t^{(1)}, A_t^{(2)}) \in \mathcal{A}_s$, where $\mathcal{A}_s$ is the set of actions available to the CM in state $s \in \mathcal{S}$. The state of the system evolves as shown in Table I, and the cost function is given by $\mu(S_t, A_t) = C_t \cdot |A_t^{(1)}|$. We will show in a longer version of this paper that there exists an optimal policy $\pi^*(s, \cdot)$ that is a piecewise constant function of the channel state C_t for any $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$.

B. Structure of the Policy

The action $A_t = (A_t^{(1)}, A_t^{(2)})$ can be written as $A_t = (\{L_1, L_2, \cdots\}, \{l_1, l_2, \cdots\})$, where L_i's and l_i's denote the remaining lifetimes of the contents pushed into, and removed from the cache, respectively. In fact, since each action is constrained by the cache capacity B, we have $A_t = (\{L_1, \cdots, L_{B'}\}, \{l_1, \cdots, l_{B'}\})$, for some $B' \leq B$. Note that we can always zero-pad one of the sets $A_t^{(1)}$ and $A_t^{(2)}$ such that we have $A_t^{(1)} = A_t^{(2)} = B'$. Here, $L = 0$ implies that no content is actually downloaded, and $L = 0$ would be used when more files are removed from the cache than those that are downloaded. Note that, this can only happen when $U_t = 1$. On the other hand, $l = 0$ implies that no content is removed from the cache memory, and $l = 0$ is

included in action $A_t^{(2)}$ when a new content is pushed into an empty location in the cache memory. We can express an action A more intuitively by pairing the elements of the sets $A^{(1)}$ and $A^{(2)}$. We call each of these B' pairs a *simple action* and denote it as $a' = (l|L)$. A simple action $a' = (l|L)$ replaces a cache content with remaining lifetime l with another relevant content out of the cache with remaining lifetime L, i.e., it swaps the two contents. Therefore, any action A can be expressed as a sequence of simple actions, i.e., $A = \{(l_1|L_1), \cdots, (l_{B'}|L_{B'})\}$. Recall that, whenever the user accesses the OSN, i.e., $U_t = 1$, all the relevant contents outside the cache are downloaded, and moved to the application layer together with all the contents in the cache memory. Therefore, whenever $U_t = 1$, there is no degree of freedom, and all policies behave similarly. When $U_t = 0$, the action will be a sequence of simple actions, where B' new contents will be pushed into the cache, while no more than B' contents will be removed. Expressing the action A as a sequence of simple actions helps us characterize the structure of the optimal policy as stated in Theorem 1 below.

Theorem 1. *For any state $s = (\mathcal{O}, \mathcal{I}, E) \in \mathcal{S}$ and channel cost C, let $l_1 \leq \cdots \leq l_B$ denote the contents of the cache $\mathcal{I}$, and let $L_1 \geq \cdots \geq L_B$ denote the B largest elements of $\mathcal{O}$. Then there is a $B' \leq B$ and corresponding threshold values $\mathcal{T}(a'_{B'}) \leq \mathcal{T}(a'_{B'-1}) \leq \cdots \leq \mathcal{T}(a'_1) \leq C_{max}$ such that there is an optimal cache management policy that performs the simple actions $a'_i = (l_i|L_i)$ for all i for which $C \leq \mathcal{T}(a'_i)$ if $E > 0$ (i.e., when the user does not access the contents).*

It is easy to see that replacing $l \in \mathcal{I}$ with $L \in \mathcal{O}$ when $l < L$ will increase the value of a state. Intuitively, it is better to have contents with longer remaining lifetimes in the cache. Therefore, for any state $s = (\mathcal{O}, \mathcal{I}, E)$ with $E > 0$, if $l_1 \leq \cdots \leq l_B$ are the contents in $\mathcal{I}$, and $L_1 \geq \cdots \geq L_B$ are the B largest elements of $\mathcal{O}$, the best action that performs exactly $B' \leq B$ swaps (all such actions have the same cost) is $A_{B'} \triangleq \{(l_1|L_1), \ldots, (l_{B'}|L_{B'})\}$. Therefore, an optimal policy can only use actions $A_1, \ldots, A_B$. If $n > n'$, after applying A_n, the future value V^{π^*} of the new state is at least as large as the value after applying $A_{n'}$; therefore, $A_{n'}$ can only be applied in an interval of the channel cost that is smaller than the interval for A_n. Using that $A_{n'} \subset A_n$, we obtain the structural result about the optimal policy.

IV. Parameterised Threshold Policy

Due to the large state and action spaces in the cache management problem, it is infeasible to solve the MDP for the optimal threshold policy by using the policy iteration algorithm (PIA) [21]. Instead, we resort to the policy search (PS) method [22], which considers a parameterised policy $\pi_{\boldsymbol{\theta}}$, and searches for the optimal parameter values within a reduced dimension parameter space $\boldsymbol{\Theta}$, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Model-free PS allows continuous evaluation of the quality of decisions made by the CM over a *trajectory* $\tau_{\pi_{\boldsymbol{\theta}},T} = (S_1, C_1, A_1), \ldots, (S_T, C_T, A_T)$, which is obtained by generating "samples" using the transition probabilities $P(s'|s, a)$ and the probability density function $f_C(c)$. Hence, $\tau_{\pi_{\boldsymbol{\theta}},T} \sim P_{\boldsymbol{\theta},T}(\tau_{\pi_{\boldsymbol{\theta}}}) = P(\tau_{\pi_{\boldsymbol{\theta}},T}|\boldsymbol{\theta})$. There are three steps involved in any model-free PS method: *policy evaluation*, in which the average sample cost $J_{\pi_{\boldsymbol{\theta}},T}(\tau_{\pi_{\boldsymbol{\theta}}}) = \frac{1}{T}\sum_{t=1}^{T} \mu(S_t, A_t)$ of a sample trajectory is obtained; *policy exploration*, which determines how new trajectories are created for the subsequent policy evaluation step; and *policy update*, in which the parameters $\boldsymbol{\theta}$ are updated such that the trajectories with lower costs become more likely; and hence, the expected average cost $\rho^{\pi_{\boldsymbol{\theta}}} = \mathbb{E}[J_{\pi_{\boldsymbol{\theta}},T}]$ decreases with the new policy. For infinite trajectories ($T = \infty$) we shall use the simpler notation $P_{\boldsymbol{\theta}}, \tau_{\pi_{\boldsymbol{\theta}}}, J_{\pi_{\boldsymbol{\theta}}}$. We will estimate $\rho^{\pi_{\boldsymbol{\theta}}}$ on infinite trajectories by taking sample averages over independent finite trajectories via Monte-Carlo rollouts. We use the *policy gradient* (PG) method [22], [23], a model-free PS method, which uses gradient descent to minimize the expected average cost $\rho^{\pi_{\boldsymbol{\theta}}}$ by following the parameter update direction given by the gradient $\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}}$. The policy update at the end of the jth iteration step is given by

$$\boldsymbol{\theta}_{j+1} = \boldsymbol{\theta}_j - \lambda \nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}}, \tag{2}$$

where $\lambda > 0$ is the step size and the policy gradient is given by

$$\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}} = \int_{\tau} \nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}}) J_{\pi_{\boldsymbol{\theta}}}(\tau_{\pi_{\boldsymbol{\theta}}}) d\tau \ . \tag{3}$$

A. Policy Representation

Based on Theorem 1, a natural parametrisation for the cache management problem is based on the threshold values. However, the optimal threshold value for each simple action depends in general on the remaining lifetimes of all the relevant contents inside and outside the cache. This suggests that the optimal policy may employ a different threshold value for the same simple action at different system states. This results in a prohibitevely large paramater space, increasing the computational complexity of the PS approach.

To overcome this dimensionality problem, in [19] we proposed a suboptimal policy, which ignores the dependence of the threshold values on the system state, and considers a single threshold value for each simple action. In this paper, we propose an improved threshold-based policy, which takes into account the remaining lifetimes of the contents in the cache memory when deciding on the threshold values. In order to limit the computational complexity of the corresponding PS, we employ linear function approximation (LFA) [24], and call the resulting policy the LFA policy.

Note that, since all the contents have the same size, the optimal threshold values do not depend on the lifetimes of particular contents, but depend on the number of contents in the cache with each remaining lifetime. Accordingly, to describe the state of the cache at time slot t, we define a *frequency vector* $\Phi_t = [\phi_t(0), \phi_t(1), \ldots, \phi_t(K_{max})]$, where the component $\phi_t(i)$ is the ratio of contents with remaining lifetime i in the cache, defined as

$$\phi(i) \triangleq \frac{\sum_{l \in \mathcal{C}} \mathbb{I}_{\{l=i\}}}{B}, \quad \text{for} \quad i = 0, 1, \cdots, K_{max}, \tag{4}$$

where $\mathbb{I}_{\{\cdot\}}$ is the indicator function, K_{max} is the maximum lifetime, and $l = 0$ denotes the empty locations in the cache memory. Note that we have $0 \leq \phi(i) \leq 1$, and $\sum_{i=0}^{K_{max}} \phi(i) = 1$.

We define the threshold policy $\mathcal{T}(l|L)$, for $l < L$, $l, L \in \{0, \ldots, K_{max}\}$, such that the two contents with remaining lifetimes l and L, inside and not inside the cache memory, respectively, are swapped if $C \leq \mathcal{T}(l|L)$, where each threshold value is obtained as a linear function of the frequency vector Φ_t:

$$\mathcal{T}(l|L) = \sum_{i=0}^{K_{max}} \phi(i)\theta_i(l, L) = \Phi^\top \theta(l, L) \ , \tag{5}$$

where $\theta_i(l, L)$, $l < L$, $l, L \in \{0, \ldots, K_{max}\}$, are the coefficients to be optimized for each simple action.

Remark 1. *We remark that the LISO policy proposed in [19], which is parameterised directly using the threshold values of the simple actions (ignoring the cache contents) can be considered as a particular special case of the policy with LFA defined in* (5)*, by setting* $\theta_{i\neq l}(l, L) = 0$.

Next we describe the methods we use to estimate the gradient $\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}}$ in (3).

B. Finite Difference Method (FDM)

In FDM, policy exploration is performed by applying small perturbations $\Delta\boldsymbol{\theta}^{[i]}$ (for trajectory i) to the current parameter vector $\boldsymbol{\theta}_j$. The perturbations are drawn from a uniform distribution with range $(\Delta\theta_{min}, \Delta\theta_{max}) = (-r, r)$, where $r \in \mathbb{R}^+$ is a relatively small number that must be carefully selected. For each trajectory, policy evaluation is performed for both perturbed and unperturbed parameter vectors to obtain $J_\pi(\boldsymbol{\theta}_j + \Delta\boldsymbol{\theta}^{[i]})$ and $J_\pi(\boldsymbol{\theta}_j)$, respectively, using the deterministic policy $\pi_{\boldsymbol{\theta}}(s) = \mathbb{I}_{\{c \leq \mathcal{T}(l|L)\}}$ $\forall l, L \in (\mathcal{O}, \mathcal{I})$, where $\mathcal{T}(l|L)$ is as defined in (5), and the corresponding performance difference $\Delta J_\pi^{[i]} = J_\pi(\boldsymbol{\theta}_j + \Delta\boldsymbol{\theta}^{[i]}) - J_\pi(\boldsymbol{\theta}_j)$ is evaluated. For policy update, the gradient is estimated numerically from samples via regression as

$$\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}} = \left(\Delta\boldsymbol{\Theta}^\top \Delta\boldsymbol{\Theta}\right)^{-1} \Delta\boldsymbol{\Theta}^\top \Delta\boldsymbol{J}_\pi, \tag{6}$$

where $\Delta\boldsymbol{\Theta} = [\Delta\boldsymbol{\theta}^{[1]}, \cdots, \Delta\boldsymbol{\theta}^{[N]}]^\top$ and $\Delta\boldsymbol{J}_\pi = [\Delta J_\pi^{[1]}, \cdots, \Delta J_\pi^{[N]}]^\top$.

C. Likelihood-Ratio Method (LRM)

In LRM, rather than perturbing the parameters of a deterministic policy as in FDM, policy exploration is performed by using a randomized policy $\pi_{\boldsymbol{\theta}}(a_t|s_t, t) \in [0, 1]$ to select actions in every time step t of each trajectory, and to obtain the average cost $J_{\pi_{\boldsymbol{\theta}}}(\tau_{\pi_{\boldsymbol{\theta}}})$ at the end of each trajectory. This implies that there is no need in tuning parameters. The policy gradient in (3) is estimated by using the "likelihood-ratio" trick given by the identity $\nabla \log P_{\boldsymbol{\theta}}(y) = \nabla P_{\boldsymbol{\theta}}(y)/P_{\boldsymbol{\theta}}(y)$ to obtain

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}} &= \int_\tau P_{\boldsymbol{\theta}}(\tau) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}}) J_{\pi_{\boldsymbol{\theta}}}(\tau_{\pi_{\boldsymbol{\theta}}}) d\tau \\ &= \mathbb{E}_\tau[\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}}) J_{\pi_{\boldsymbol{\theta}}}(\tau_{\pi_{\boldsymbol{\theta}}})] \ . \end{aligned} \tag{7}$$

Note that the derivative $\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}})$ can be obtained without knowledge of the trajectory distribution $P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}})$ as the product in $P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}}) = P(s_1) \prod_{t=1}^{T} P(s_{t+1}|s_t, a_t) \pi_{\boldsymbol{\theta}}(a_t|s_t, t)$ is transformed to sum when logarithm is introduced and all the terms that do not depend on the parameter $\boldsymbol{\theta}$ disappear during differentiation, such that,

$$\nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}}) = \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t, t) \ .$$

As $\int_\tau \nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\tau_{\pi_{\boldsymbol{\theta}}}) d\tau = 0$, a baseline b (a constant) can be introduced into the expression in (7) to minimize the variance of the gradient estimate $\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}}$ as in the REINFORCE algorithm [20]. Therefore the gradient estimate can be expressed as

$$\nabla_{\boldsymbol{\theta}} \rho^{\pi_{\boldsymbol{\theta}}} = \mathbb{E}_\tau \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t, t) (J_{\pi_{\boldsymbol{\theta}}}(\tau_{\pi_{\boldsymbol{\theta}}}) - b) \right], \tag{8}$$

The optimal baseline b depends on individual components h : $\theta_h \in \boldsymbol{\theta}$, and it satisfies the condition $\frac{\partial}{\partial b} Var[\nabla_{\boldsymbol{\theta}_h} \rho^{\pi_{\boldsymbol{\theta}}}] = 0$, whose solution yields

$$b_h = \frac{\mathbb{E}_\tau \left[\left(\sum_{t=1}^{T} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(a_t|s_t, t) \right)^2 J_{\pi_{\boldsymbol{\theta}}}(\tau_{\pi_{\boldsymbol{\theta}}}) \right]}{\mathbb{E}_\tau \left[\left(\sum_{t=1}^{T} \nabla_{\theta_h} \log \pi_{\boldsymbol{\theta}}(a_t|s_t, t) \right)^2 \right]} \ . \tag{9}$$

In time slot t, for each simple action $a'_{i,t} = (l_i|L_i)$: $l_i < L_i$, $l, L \in \{0, 1, \ldots, K_{max}\}, i \in \{1, \ldots, B'\}, B' \leq B$, we define a randomized policy $\pi_\theta(a'_{i,t}|s_t, t); \theta \in \boldsymbol{\theta}$, as a sigmoid function with negative slope parameter as follows:

$$\pi_\theta(a'_{i,t}|s_t, t) = \frac{1}{1 + e^{-\eta(c - \Phi_t^\top \theta(l,L))}} \ , \tag{10}$$

where $\eta < 0$ is a negative slope factor. This implies that for all $l \in \mathcal{I}$ and all $L \in \mathcal{O}$,

$$\pi_{\boldsymbol{\theta}}(a_t|s_t, t) \equiv \prod_{i=1}^{B'} \pi_\theta(a'_{i,t}|s_t, t) \ .$$

Note that the policy is defined only for $E_t > 0$, i.e., the user does not access the OSN. If $E_t = 0$, relevant contents outside the cache are downloaded, and all the relevant contents are moved to the application layer and removed from the cache at the end of the time slot.

V. Performance Lower Bounds

In order to assess the performance of the proposed suboptimal cache management policy with LFA, we benchmark its performance against two lower bounds.

A. Unlimited Cache Capacity

The first lower bound, which we refer to as LB-UC, is obtained by assuming that the cache capacity is sufficient to store all the relevant contents at any time (e.g., $B = \infty$). In this case there is no need to remove any content from the cache that is neither consumed nor expired; and therefore, the decision about each content that is not inside the cache can be taken individually, and independent of the contents of

the cache. Therefore, following from Theorem 1, if $E > 0$, there exists a threshold $\mathcal{T}_L$ for which a content with remaining lifetime L is downloaded if $C \leq \mathcal{T}_L$, and the threshold has the structure: $0 \leq \mathcal{T}_1 \leq \cdots \leq \mathcal{T}_{K_{max}} \leq C_{max}$. Moreover, if the user accesses the OSN with i.i.d probability p_a in each time slot, i.e., E has a geometric distribution, the thresholds can be obtained numerically via dynamic programming [21], such that, if V_L is the value function for downloading a content with remaining lifetime L, the value function, given the threshold policy $\mathcal{T}_L$, can be expressed as

$$V_{L,\mathcal{T}_L} = p_a \mathbb{E}[C] + (1-p_a)\Big(F_C(\mathcal{T}_L)\mathbb{E}\left[C\middle|C \leq \mathcal{T}_L\right] + \bar{F}_C(\mathcal{T}_L)V_{L-1}\Big), \tag{11}$$

where $F_C(\cdot)$ is the cdf of C and $\bar{F}_C(\cdot)$ is its complement. The optimal threshold $\mathcal{T}_L$ satisfies the condition that the derivative $\nabla_{\mathcal{T}_L} V_{L,\mathcal{T}_L} = 0$, whose solution yields

$$\mathcal{T}_L = V_{L-1}. \tag{12}$$

For all possible remaining lifetimes $1 \leq L \leq K_{max}$, the thresholds can be obtained using (11) and (12), and noting that $V_0 = 0$, i.e., an expired content cannot be downloaded.

B. Non-Causal Knowledge of User Behavior

The second lower bound, which we refer to as LB-NCK, assumes that the CM has non-causal knowledge of the exact time slots the user accesses the OSN, in which case, contents that will expire before the user access times are known. Therefore, in every time slot t, and for every *time-to-user access*, only contents (including the M_t new contents generated in time slot t) that will remain relevant by the user access time are considered for download. All such contents are of equal importance, therefore, as many contents as the cache capacity can allow will be downloaded in any time slot that the CM decides to download contents. The decision to download contents also follows from Theorem 1, such that, for any time-to-user access t', there exists a threshold $\mathcal{T}_{t'}$ for which contents are downloaded if $C_t \leq \mathcal{T}_{t'}$. The threshold has the structure: $0 \leq \mathcal{T}_{D_{max}} \leq \cdots \leq \mathcal{T}_1 \leq C_{max}$ (recall that D_{max} is the bound on the user access interval). Obtaining the threshold values follows from (11) (with $p_a = 0$) and (12), and noting that $V_0 = E[C]$.

VI. Performance Evaluation

In this section we apply LRM, i.e., the REINFORCE algorithm, to the LISO policy proposed in [19], and both FDM and LRM together with LFA, considering particular models for the statistical variables in the system model. We evaluate the performances of the algorithms and policies, and compare them with the two lower bounds described in Section V.

A. Simulation Setup

We assume an interference-free channel, and obtain the instantaneous cost C_t using Shannon's capacity formula,

$$R = W \log_2(1 + \text{SNR}),$$

where R is the transmission rate, W is the channel bandwidth, and $\text{SNR} = P_{signal}/P_{noise}$ is the signal-to-noise ratio. We assume a spectral efficiency of $R/W = 2$ bps/Hz. Using the Long Term Evolution (LTE) network model [25]; the noise power is given as

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

where $kT = -174$ dBm/Hz is the noise power spectral density and $NF = 5$ dB is a typical noise figure. To compute the noise power, we assume a fixed (average) bandwidth of 10 MHz in every time slot. The signal power is given by

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

where C_t is the instantaneous cost, G_{TX} and G_{RX} are the transmit and receive antenna gains. We use the values $G_{TX} = 17$ dBi and $G_{RX} = 0$ dBi in our simulations. $PL(d)$ is the path loss, which is a function of the distance d between the user and the serving BS. To compute the path loss, we assume that small scale fading effect is averaged out over the system configuration of interest, and we adopt the 3GPP channel model [26] with the path loss given by

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

where $f_c = 2.5$ GHz 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. In every time slot, the distance d, in metres, is drawn from a uniform distribution; $d \sim \mathcal{U}(50, 250)$, modeling a system in which a mobile user ends up in a different cell at each time slot.

We assume that the random variables M and K, which, in each time slot, generate M_t new contents and $K_{t,i}$ lifetime for each content $i = 1, \ldots, M_t$, are drawn from a uniform distribution over the sets $\{1, \ldots, 8\}$ and $\{5, 10, 15\}$, respectively. We obtain the user behaviour sequence $\{U_t\}$ with i.i.d. probability of access p_a in every time slot. We measure the cache capacity B in number of contents, and we assume the initial states $\mathcal{O}_0 = \mathcal{I}_0 = \emptyset$, and $E_0 = 0$.

For the FDM algorithm, we select the perturbation parameters from $(\Delta\theta_{min}, \Delta\theta_{max}) = (-0.08, 0.08)$, and for the LRM algorithm, we select the slope factor $\eta = -10$. Also, for the FDM algorithm, policy update is done after 100 trajectories, while for the LRM algorithm, policy update is done after 20 trajectories. The duration T of each trajectory is 300 time slots. To test the performances of the algorithms, we use a test data consisting of 100 trajectories, with each trajectory having a duration $T = 5000$ time slots. For each algorithm and system parameters, we carefully select the step size λ.

B. Numerical Results

In Figure 2, we plot the average energy cost as a function of the cache capacity. Noting from the result presented in [19] that the average cost converges to the LB-UC bound when the cache capacity $B > 30$, we focus on lower cache capacities in this figure. In general, the performance of both policies, and the respective algorithms, increase with the cache capacity B, approaching the LB-UC bound. Apart from the

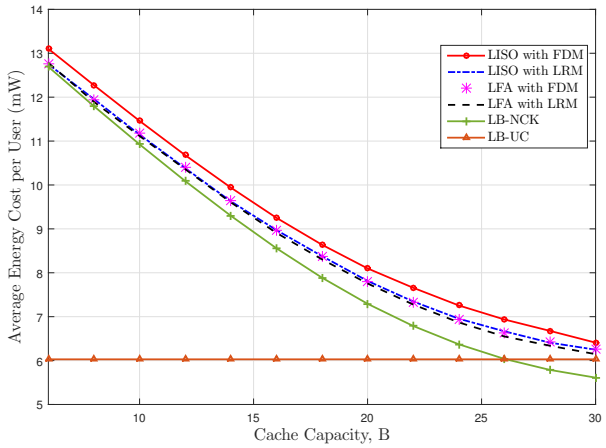

Fig. 2. Average energy cost vs. cache capacity for $K_{max} = 15, M_{max} = 8, D_{max} = 15, p_a = 0.25$.

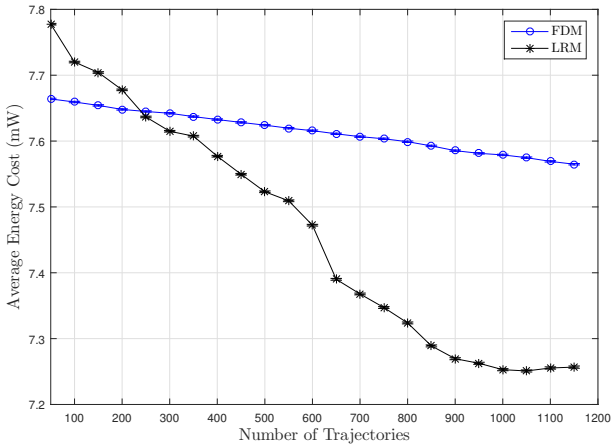

Fig. 3. The evolution of the FDM and the LRM algorithms with respect to the number of trajectories.

LISO policy implemented with FDM, the threshold policy with LFA implemented with both the FDM and the LRM algorithms, and the LISO policy implemented with the LRM algorithm, converge to the LB-NCK bound at $B = 6$, making it clear that the LFA policy and the LRM algorithm perform optimally at low cache capacities.

When implemented with the FDM algorithm, the threshold policy with LFA has a performance improvement of up to 4.4% over the LISO policy. Meanwhile, the LISO policy implemented with the LRM algorithm has up to 4.2% improvement in performance over the FDM algorithm. This may be influenced by the stochasticity of the system. The best performance is obtained when LFA is implemented with the LRM algorithm, with up to 5.6% improvement in performance over the LISO policy implemented with the FDM algorithm. This gain can be attributed to the fact that the proposed threshold policy with LFA takes into account the remaining lifetimes of the contents that are already in the cache when taking caching decisions. However, it only shows a small improvement over LFA implemented with the FDM algorithm. We can argue that the parameterisation of LFA allows better performance for the FDM algorithm as we see a more significant improvement with LFA compared to the LRM algorithm.

In order to compare the convergence rates of the FDM and LRM algorithms, we plot the average energy cost as a function of the number of trajectories in Figure 3. The result shows that the LRM algorithm has a much faster convergence rate, and saturates to its optimal value after approximately 1000 trajectories, whereas the performance of the FDM algorithm improves linearly, at a much slower rate with the number of trajectories.

In Figure 4, we plot the average energy cost as a function of the maximum lifetime of contents. We evaluate the performance of both the LRM and LFA algorithms with respect to K_{max}. Energy cost increases when contents are generated with longer lifetimes because more contents will be consumed by the user at the time of access. However, we observe that the PS with LRM, and LFA both improve the system performance with increasing lifetime compared to LISO implemented with the FDM algorithm. We observe that the LFA, which considers the remaining lifetime of all the contents inside the cache, has a further gain in the energy cost for all the maximum lifetime values considered. We also notice that the gain from LFA may increase with the maximum lifetime. This observation can be noticed in the plot when $K_{max} = 20$ and $B = 20$. We also observe that there is a further improvement in the performance of LFA with LRM algorithm compared to LISO with LRM algorithm when the cache capacity is increased from $B = 20$ to $B = 30$.

VII. Conclusions

We have studied proactive caching of contents into a finite capacity cache memory of a mobile device, with the aim of reducing the long term average energy consumption in content delivery over wireless networks. Content generation is modeled considering an OSN framework, such that a random number of contents are generated by the social connections of the user at each time slot, each with a random lifetime, which represents the time period the content remains relevant for the user. The user accesses the OSN through a mobile app at random time intervals, and requests all the relevant contents at the time of access. Since contents are downloaded through a wireless connection, each download incurs an energy cost that depends on the channel state at the time of download. We have proposed proactive caching in order to reduce the long-term average energy consumption of the system by pushing contents into the cache memory in advance, over more favorable channel states.

The optimal policy for this problem is a threshold-based policy which swaps relevant contents that are not inside the cache with those in the cache according to a threshold on the channel condition which depends on the system state. However, characterising the optimal policy requires identifying a prohibitively

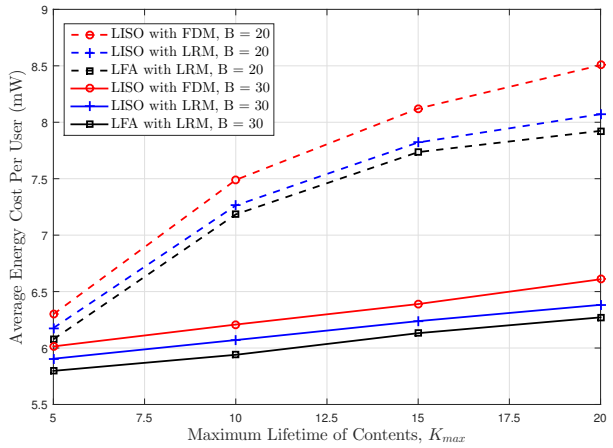

Fig. 4. Average energy cost vs. maximum lifetime of contents for cache capacities B = 20 and B = 30 when $M_{max} = 8, D_{max} = 15, p_a = 0.25$.

large set of threshold values. Instead, we propose suboptimal parameterised caching policies. In particular, we considered a stochastic policy, which allowed us to implement the LISO policy in [19] using policy search with the REINFORCE algorithm. We then proposed a new parameterisation of the problem using LFA, where the threshold values are obtained as linear functions of a frequency vector, which represents the distribution of the remaining lifetimes of the contents inside the cache. Numerical results show an improved performance for the proposed threshold policy with LFA compared with the simplified LISO policy in [19]. While the improvement is limited in the particular scenario considered in this paper, in which the content generation and lifetime processes are i.i.d. over time, we believe that the gains from LFA will be more significant when the involved stochastic processes have memory, which will be explored in a future work.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers for their insightful comments. This research was supported in part by The Petroleum Technology Development Fund (PTDF), and the European Research Council (ERC) through Starting Grant BEACON (agreement #677854).

REFERENCES

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

[2] F. Malandrino *et al.*, "Proactive seeding for information cascades in cellular networks," in *2012 IEEE Conference on Computer Communications (INFOCOM)*, March 2012, p. 17191727.

[3] Z. Chang *et al.*, "Context-aware data caching for 5g heterogeneous small cells networks," in *IEEE Int'l Conf. on Comms*, June 2016.

[4] V. A. Siris, X. Vasilakos, and G. C. Polyzos, "Efficient proactive caching for supporting seamless mobility," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–6.

[5] N. Golrezaei *et al.*, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE INFOCOM*, Mar. 2012, pp. 1107–1115.

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

[7] E. Batu, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2015 13th International Symposium on*, May 2015, pp. 161–166.

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

[9] A. Sengupta *et al.*, "Learning distributed caching strategies in small cell networks," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE, 2014, pp. 917–921.

[10] S. Müller *et al.*, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *IEEE Int'l Conference on Communications (ICC)*, June 2016.

[11] S. Zhou *et al.*, "GreenDelivery: Proactive content caching and push with energy-harvesting-based small cells," *Communications Magazine, IEEE*, vol. 53, no. 4, pp. 142–149, April 2015.

[12] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE Journal on Selected Areas in Comms*, vol. 34, no. 1, pp. 176–189, Jan 2016.

[13] L. Zhang *et al.*, "Efficient scheduling and power allocation for d2d-assisted wireless caching networks," *IEEE Transactions on Communications*, vol. 64, no. 6, pp. 2438–2452, June 2016.

[14] S. W. Jeon *et al.*, "Caching in wireless multihop device-to-device networks," in *IEEE Int'l Conf. on Comms.*, Jun. 2015, pp. 6732–6737.

[15] M. Mohammadi Amiri, Q. Yang, and D. Gündüz, "Coded caching for a large number of users," in *Proc. Information Theory Workshop (ITW)*, Cambridge, UK, Sep. 2016, pp. 171–175.

[16] K. Poularakis *et al.*, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. on Wireless Comms.*, vol. 15, no. 4, pp. 2995–3007, Apr. 2016.

[17] A. C. Gungor and D. Gündüz, "Proactive wireless caching at mobile user devices for energy efficiency," in *Proc. IEEE Int'l Symp. on Wireless Comm. Systems (ISWCS)*, Brussels, Belgium, Sep. 2015, pp. 171–175.

[18] M. Gregori *et al.*, "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.

[19] S. O. Somuyiwa, A. György, and D. Gündüz, "Energy-efficient wireless content delivery with proactive caching," in *Proc. Workshop on Content Caching and Del. Wireless Nets.*, 2017.

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

[21] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[22] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Found. Trends Robot*, vol. 2, pp. 1–142, Aug. 2013. [Online]. Available: http://dx.doi.org/10.1561/2300000021

[23] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 2219–2225.

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

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

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