Optimal Dynamic Proactive Caching via Reinforcement Learning

Alireza Sadeghi, Fatemeh Sheikholeslami, and Georgios B. Giannakis
Dept. of Elec. & Comput. Engr. and Digital Technology Center, University of Minnesota, USA
{sadeghi, sheik081, georgios}@umn.edu

***Abstract*—Storage of popular reusable data at the edge of a heterogeneous wireless cellular network (HetNet) offers the premise of shifting the load on low-rate, unreliable backhaul links during peak traffic hours to off-peak periods. In order to intelligently capitalize on the limited available caching capacity, a *content-agnostic* small base station (SB) needs to *proactively* learn *what* and *when* to cache. An important challenge in a realistic network scenario is the *spatio-temporal dynamics*, inherent to the *unknown* content popularity profiles. To cope with such dynamics, local and global Markov processes are exploited to model user demands, whose structure and transition probabilities are assumed unknown. A reinforcement learning framework is put forth, through which a cache control unit (CCU) at the SB can continuously *learn*, *track*, and possibly *adapt* to the underlying dynamics of user demands. A Q-learning algorithm is developed to solve the proposed reinforcement learning task, unraveling the *optimal* caching policy in an online fashion. Simulated tests demonstrate the effectiveness of the proposed proactive caching scheme under spatio-temporal dynamic demands.**

***Index Terms*—Proactive caching, dynamic popularity profile, dynamic user demand, reinforcement learning.**

I. Introduction

The advent of smart phones, tablets, mobile routers and a massive number of devices connected through the Internet of Things (IoT) have led to an unprecedented growth in data demand. Increased number of users undergoing behavioral changes towards video streaming, web browsing, social networking and online gaming, have urged providers to pursue new service schemes under which acceptable quality of experience (QoE) can be provided. Among the emerging solutions, one promising technique is to densify the network by deploying small pico- and femtocells, each of which will be serviced by a low-power, low-coverage small base station (SB). In this structure, known as heterogeneous cellular networks (HetNet), SBs are connected to the backbone network by a cheap 'backhaul' link. While boosting the networking capacity per area by substantial reuse of scarce resources, e.g., frequency, the HetNet architecture is restrained by its low-rate, unreliable and highly-delayed backhaul links [1].

During peak traffic periods with increased electricity prices, weak backhaul links can easily become congested, leading to low QoE for end users. One approach for tackling this limitation is to shift the excess load from peak traffic to the off-peak periods. *Caching* in particular achieves this goal by fetching the "anticipated" popular *contents*, i.e., reusable data, during off-peak periods, *storing* this data in memory-enabled SBs, and *reusing* them during peak traffic hours [2], [3]. In order to utilize the caching capacity intelligently, a *content-agnostic* SB must exploit available observations to learn *what* and *when* to cache. To this end, machine learning tools can provide Next-G cellular networks with *proactive caching*, in which a "smart" caching control unit (CCU) can *learn*, *track*, and possibly *adapt* to the dynamics of user demands [2], [4].

Existing efforts in proactive caching are mainly focused on enabling SBs to learn unknown static content *popularity profiles*, and cache the most popular ones accordingly. Multi-armed bandit formulation [5], its distributed and convexified versions [6], and utilization of prior information through transfer-learning [7], [8], are among such recent efforts. Nevertheless, static modeling of popularity profiles is unrealistic as it neglects the spatio-temporal dynamics of demands. For instance, the emergence of new contents such as news or the next El Clásico, along with popularity decay of relatively older contents cannot be captured by such models. Furthermore, due to the geographical and temporal variability of cellular data traffic, global popularity profiles may not always be a good representative of local demands. Targeting such considerations, temporal dynamics of user demands have been modeled via Ornstein-Uhlenbeck process in [9] and tackled through a mean-field game-theoretic approach. A context-aware proactive caching is studied in [10], [11], where dynamic user demands are mapped into a pool of prototypical trends.

The present paper introduces a novel approach to account for spatio-temporal variability of demands casting the task in a reinforcement learning (RL) framework. Under Markovianity of the underlying dynamics, RL-enabled caching can learn the unknown behavior of the network in terms of demand dynamics and time-varying costs, and consequently unravel the optimal "caching policy." The proposed approach is also capable of differentiating between global and local popularity patterns, and adapting the caching policy of the local SBs accordingly. With proper selection of updating stepsize, the framework is also capable of adaptively tracking demands driven by non-stationary Morkov chains.

The rest of the paper is organized as follows. System model and problem formulation are the subject of Section II. Section III casts the problem in an RL framework, in which an online solver based on Q-learning is introduced. Section IV presents simulations for the proposed proactive caching approach, and Section V delivers concluding remarks.

This work was supported by USA NSF grants 1423316, 1508993, 1514056, 1711471.

II. System Model and problem formulation

In order to model a local section of a HetNet, let us consider a single SB with a low-bandwidth, high-delay, unreliable backhaul link. The SB is responsible for providing a high QoE for end users within its coverage area, and is equipped with a storage capacity of M units of content. Furthermore, suppose that the network is a time-slotted system with time intervals $t = 1, 2, \ldots$. At the beginning of each interval, the CCU-enabled SB is to "intelligently" select M files from the total number of $F \gg M$ available ones at the backbone, and cache those for the duration of the upcoming time interval t. For simplicity, here we assume all contents are of unit size. Let us define the $F \times 1$ caching-indicator action of CCU at time interval t by $\mathbf{a}(t) \in \mathcal{A}$, where $\mathcal{A}$ is the set of all feasible actions defined as $\mathcal{A} := \{\mathbf{a} | \mathbf{a} \in \{0,1\}^F, \mathbf{a}^\top \mathbf{1} = M\}$. That is, $a_f(t) = 1$ indicates that file f is cached for the duration of time interval t, and $a_f(t) = 0$ indicates otherwise.

During time interval t, every local user may request a subset of available files. For every requested file, if the SB has stored the content in its cache memory, it will simply transmit the file to the user, in which case the SB incurs no cost. Conversely, if the requested content is not available in cache, the SB must fetch its content through its low-rate backhaul link from the backbone network, thus incurring certain cost.

According to the received requests by the end of time interval t, the CCU can compute the *local* $F \times 1$-vector of content popularity profile $\mathbf{p}_L(t)$, whose f-th entry indicates the expected local demand of file f, that is

$$\left[\mathbf{p}_{\mathrm{L}}(t)\right]_f = \frac{\text{\# of local requests for file } f \text{ at time slot } t}{\text{Total \# of local requests at time slot } t}.$$

Moreover, suppose that the backbone network obtains the $F \times 1$ *global* popularity profile $\mathbf{p}_G(t)$ similarly, and transmits that to all the CCUs.

Let us now define the system state at the end of slot t as

$$\mathbf{s}(t) := \left[\mathbf{p}_G^\top(t), \mathbf{p}_L^\top(t), \mathbf{a}^\top(t)\right]^\top. \tag{1}$$

In the proposed *proactive* caching, the goal is to find the *optimal* action for the next time interval, namely $\mathbf{a}^*(t)$ on the fly, as the current state $\mathbf{s}(t)$ and associated costs are observed. A schematic of the proposed procedure is depicted in Fig. 1, and explicit expression of the incurred cost and analytical formulation of the objective are discussed in the ensuing subsection.

A. Cost functions and caching strategies

Efficiency of a caching algorithm is measured by how well it utilizes the available caching capacity in the local SB to store the most popular files, and how often the demand is met via fetching through the *more expensive* backhaul link. There are multiple types of costs that a CCU may incur during a time slot t.

The first type of cost pertains to fetching the selected files and refreshing the cache, denoted by $c_{1,t}(\mathbf{a}(t), \mathbf{a}(t-1))$. In its most general form, $c_{1,t}$ is a function of the new action

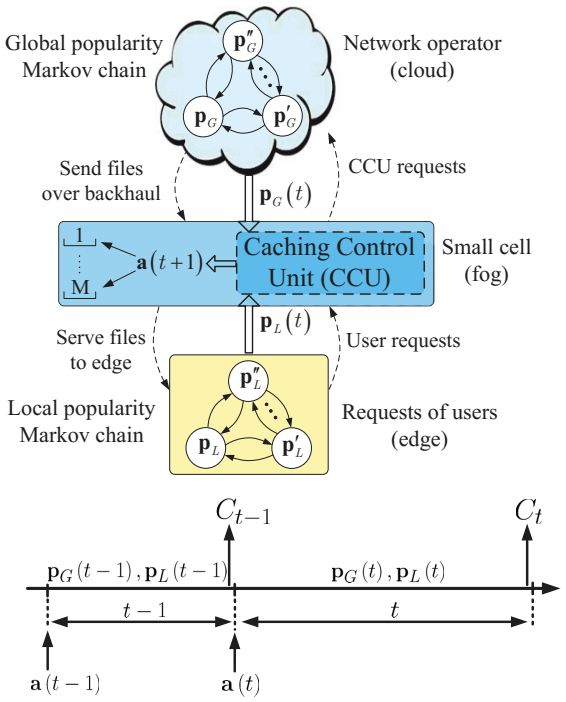

Fig. 1. Schematics of the proposed system model for a caching-enabled SB in a time-slotted network

$\mathbf{a}(t)$, and available contents at the cache, meaning those cached according to the previous caching action $\mathbf{a}(t-1)$. Note that the subscript t is to capture the time-varying nature of file retrieval cost, which can occur due to possible congestion in the backhaul connection or electricity price surges. A viable choice of $c_{1,t}(\cdot)$ would be $c_{1,t}(\mathbf{a}(t), \mathbf{a}(t-1)) := \lambda_{1,t}\mathbf{a}^\top(t)(\mathbf{1} - \mathbf{a}(t-1))$, which takes into account possible overlaps between $\mathbf{a}(t-1)$ and $\mathbf{a}(t)$, and essentially counts the number of files to be fetched and cached prior to time interval t that were not stored according to previous action $\mathbf{a}(t-1)$.

The second type of cost is incurred during the operational phase of time slot t for meeting the users' demands. Denoting this type of cost by $c_{2,t}(\mathbf{s}(t))$, a proper choice must: i) penalize requests for cached files significantly less than the ones for files not stored, and ii) be a non-decreasing function of popularities $[\mathbf{p}_L]_f$. Here for simplicity, let us assume that transmission cost of cached files is negligible, and define $c_{2,t}(\mathbf{s}(t)) := \lambda_{2,t}(\mathbf{1} - \mathbf{a}(t))^\top \mathbf{p}_L(t)$, which solely penalizes the not-cached files proportional to their local popularities.

The third type of cost considered captures how "different" the caching action $\mathbf{a}(t)$ is from the global popularity profile $\mathbf{p}_{\mathrm{G}}(t)$. Indeed, it is prudent to consider that the global popularity of files is a good indication of what the profile will look like in the near future; thus, keeping the caching action close to $\mathbf{p}_{\mathrm{G}}(t)$ can reduce future possible costs of caching. These considerations suggests the third type of cost to be modeled as $c_{3,t}(\mathbf{s}(t)) := \lambda_{3,t}(\mathbf{1} - \mathbf{a}(t))^\top \mathbf{p}_g(t)$, penalizing the not-cached files proportional to their global popularities.

Upon taking action $\mathbf{a}(t)$, and after the global and lo-

cal popularity profiles $\mathbf{p}_G(t)$ and $\mathbf{p}_L(t)$ are revealed, the state of SB is updated as $\mathbf{s}(t)$, and conditional cost $C_t(\mathbf{s}(t-1), \mathbf{a}(t) \mid \mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t))$ is incurred. Considering the peak and off-peak retrieval costs, and time-invariant scenarios $\lambda_{1,t} = \lambda_1$, $\lambda_{2,t} = \lambda_2$, and $\lambda_{3,t} = \lambda_3$, the overall conditional cost is

$$\begin{aligned} &C(\mathbf{s}(t-1), \mathbf{a}(t) \mid \mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t)) \\ &\quad := c_1\left(\mathbf{a}(t), \mathbf{a}(t-1)\right) + c_2\left(\mathbf{s}(t)\right) + c_3(\mathbf{s}(t)) \\ &\quad = \lambda_1 \mathbf{a}^\top(t)(\mathbf{1} - \mathbf{a}(t-1)) + \lambda_2(\mathbf{1} - \mathbf{a}(t))^\top \mathbf{p}_L(t) \\ &\quad + \lambda_3(\mathbf{1} - \mathbf{a}(t))^\top \mathbf{p}_g(t). \end{aligned} \tag{2}$$

Parameters λ_1 and λ_2, and λ_3 control the proportional significance of different costs on SB. In general we have $\lambda_1 \ll \lambda_2$, while λ_3 is tuned properly to match the interest in (not)following global popularities in the network. For instance, if the goal is to solely keep track of the SB's connected users, one can set $\lambda_3 = 0$, while $\lambda_3 > 0$ is desirable for networks in which users have high mobility and may change SBs rapidly.

Given the random nature of user demands in local and global scales, and possible randomness in $\mathbf{a}(t)$, the introduced cost is a random variable, whose mean is given by

$$\begin{aligned} &\mathbb{E}\left[C\left(\mathbf{s}(t-1), \mathbf{a}(t)\right)\right] \\ &\quad = \mathbb{E}_{\mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t), \mathbf{a}(t)}\left[C_t(\mathbf{s}(t-1), \mathbf{a}(t) \mid \mathbf{p}_{\rm G}(t), \mathbf{p}_{\rm L}(t))\right] \\ &\quad = \lambda_1 \mathbb{E}\left[\mathbf{a}^\top(t)(\mathbf{1} - \mathbf{a}(t-1))\right] + \lambda_2 \mathbb{E}\left[(\mathbf{1} - \mathbf{a}(t))^\top \mathbf{p}_L(t)\right] \\ &\quad + \lambda_3 \mathbb{E}\left[(\mathbf{1} - \mathbf{a}(t))^\top \mathbf{p}_g(t)\right] \end{aligned} \tag{3}$$

where the expectation is taken with respect to the randomness in $\mathbf{p}_L(t), \mathbf{p}_G(t)$, and $\mathbf{a}(t)$.

To outline the proposed Q-learning approach, let us define *policy* function $\pi : \mathcal{S} \to \mathcal{A}$, which maps any given state $\mathbf{s} \in \mathcal{S}$ into the action set. Thus, under policy $\pi(\cdot)$, for any state $\mathbf{s}(\tau)$, caching is carried out via action $\mathbf{a}(\tau + 1) = \pi(\mathbf{s}(\tau))$ dictating what files should be cached in the $(\tau + 1)$-st time slot. Consequently, caching performance is measured via the *state* value function defined as

$$V_\pi\left(\mathbf{s}(\tau)\right) := \lim_{T \to \infty} \mathbb{E}\left[\sum_{t=\tau}^{T} \gamma^{t-\tau} C\left(\mathbf{s}(t), \pi\left(\mathbf{s}(t)\right)\right)\right] \tag{4}$$

which is in fact the total cost incurred in an infinite time horizon, discounted by factor $\gamma \in (0, 1)$. Since taking action $\mathbf{a}(t)$ partially controls the state of SB in the next time slot, future costs always are effected by previous actions. Discount factor γ captures this effect, whose tuning trades off current versus future revenues. Moreover, γ can also capture modeling uncertainties, as well as imperfections, or dynamics. For instance, if there is ambiguity about future rewards, or if the system changes very fast, setting γ to a small value enables one to prioritize current costs, whereas in a stationary setting one may prefer to carefully consider future revenues by a larger γ.

The objective of this paper is to find the *optimal* policy π^* such that the cost of initial state $\mathbf{s}_0$ is minimized, that is,

$$\pi^* = \arg\min_{\pi \in \Pi} V_\pi\left(\mathbf{s}_0\right). \tag{5}$$

where Π denotes the set of all feasible policies. In the ensuing section, we study the Bellman equations which serve as policy optimality conditions, and introduce the proposed Q-learning approach for tackling (5) via reinforcement learning.

III. Bellman optimality conditions

By modeling $\mathbf{p}_{\rm L}(t)$ and $\mathbf{p}_{\rm G}(t)$ as stationary Markov processes, and defining $[\mathbf{P}^a]_{\mathbf{s}\mathbf{s}'}$ as the transition probability of going from the current state $\mathbf{s} = \mathbf{s}(\tau)$ to the next state $\mathbf{s}' = \mathbf{s}(\tau + 1)$ under action $\mathbf{a}$, that is,

$$\mathbf{P}(\mathbf{s}'; \mathbf{s}, \mathbf{a}) := \Pr\{\mathbf{s}(\tau + 1) = \mathbf{s}' | \, \mathbf{s}(\tau) = \mathbf{s}, \pi(\mathbf{s}(\tau)) = \mathbf{a}\}$$

the state value function in (4) can be rewritten in a recursive fashion as

$$V_\pi\left(\mathbf{s}\right) = \mathbb{E}\left[C\left(\mathbf{s}, \pi(\mathbf{s})\right)\right] + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \mathbf{P}(\mathbf{s}'; \mathbf{s}, \pi(\mathbf{s})) V_\pi\left(\mathbf{s}'\right) \ , \forall \mathbf{s}, \mathbf{s}'. \tag{6}$$

The set of linear equations in (6) are known as Bellman equations, and express the value of a state as the superposition of immediate cost plus a discounted version of future state values. Were the transition probabilities known, the state value function for a given policy π could be found by solving the equations in (6) with complexity $\mathcal{O}(|\mathcal{S}|^3)$.

In the setting of interest however, the transition probabilities $\mathbf{P}(\mathbf{s}'; \mathbf{s}, \mathbf{a})$ are unknown. The class of adaptive dynamic programming algorithms (ADP) aims at learning the transition probabilities $\mathbf{P}(\mathbf{s}'; \mathbf{s}, \mathbf{a})$ for all $\mathbf{s}, \mathbf{s}' \in \mathcal{S}$ and $\mathbf{a} \in \mathcal{A}$ [12]. Unfortunately, such an approach is often very slow as it estimates a huge number of $|\mathcal{S}|^2 \times |\mathcal{A}|$ parameters, which may not all be necessary. On the other hand, another class of solvers, known as Q-learning algorithms, aim at learning the optimal policy π^* and state-value function in parallel on the fly as new observations become available; see e.g., [12].

In order to utilize Q-learning solvers, let us first define the *action-state value function* $Q(\mathbf{s}, \mathbf{a})$ under a given policy $\pi\left(\cdot\right)$ as

$$Q_\pi\left(\mathbf{s}, \mathbf{a}\right) := \mathbb{E}\left[C\left(\mathbf{s}, \mathbf{a}\right)\right] + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \mathbf{P}(\mathbf{s}'; \mathbf{s}, \mathbf{a}) V_\pi\left(\mathbf{s}'\right) \tag{7}$$

which basically is the instant cost of taking action $\mathbf{a}$ when in state $\mathbf{s}$, followed by the discounted value of the following states given the future actions are taken according to policy π. Defining $Q^*(\mathbf{s}, \mathbf{a}) := Q_{\pi^*}(\mathbf{s}, \mathbf{a})$ and $V^*(\mathbf{s}) := V_{\pi^*}(\mathbf{s})$, and using the results in [12], the Bellman equation for the optimal policy $\pi^*(\cdot)$ implies

$$Q^*\left(\mathbf{s}, \mathbf{a}\right) = \mathbb{E}\left[C\left(\mathbf{s}, \mathbf{a}\right)\right] + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \mathbf{P}(\mathbf{s}'; \mathbf{s}, \mathbf{a}) \min_{\mathbf{a}' \in \mathcal{A}} Q^*\left(\mathbf{s}', \mathbf{a}'\right) \tag{8}$$

and the *optimal* policy is given by

$$\pi^*(\mathbf{s}) = \arg\min_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}), \quad \forall \mathbf{s} \in \mathcal{S}. \tag{9}$$

Furthermore, the Q-function and the state value function $V(\cdot)$ under the optimal policy π^* are related by

$$V^*(\mathbf{s}) = \min_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}). \tag{10}$$

Algorithm 1: Proactive caching via Q-learning at CCU

Initialize $\mathbf{s}(0)$ randomly and $\hat{Q}(\mathbf{s},\mathbf{a})=0\ \forall \mathbf{s},\mathbf{a}$

for *t=1,2,...* **do**

Take action $\mathbf{a}(t)$ chosen probabilistically by

$$\mathbf{a}(t)=\begin{cases}\arg\min_{\mathbf{a}}\hat{Q}\left(\mathbf{s}(t-1),\mathbf{a}\right) & \text{w.p.} \quad 1-\epsilon_t\\ \text{random } \mathbf{a}\in\mathcal{A} & \text{w.p.} \quad \epsilon_t\end{cases}$$

$\mathbf{p}_L(t)$ and $\mathbf{p}_G(t)$ are revealed

Set $\mathbf{s}(t)=[\mathbf{p}_L(t),\mathbf{p}_G(t),\mathbf{a}(t)]$

Incur cost $C(\mathbf{s}(t-1),\mathbf{a}(t)\,|\,\mathbf{p}_{\mathrm{G}}(t),\mathbf{p}_{\mathrm{L}}(t))$

Update

$$\hat{Q}_t\left(\mathbf{s}(t-1),\mathbf{a}(t)\right)\leftarrow(1-\beta_t)\hat{Q}_{t-1}\left(\mathbf{s}(t-1),\mathbf{a}(t)\right)+\beta_t\cdot\left[C\left(\mathbf{s}(t-1),\mathbf{a}(t)|\,\mathbf{p}_{\mathrm{G}}(t),\mathbf{p}_{\mathrm{L}}(t)\right)+\gamma\min_{\mathbf{a}}\hat{Q}_{t-1}\left(\mathbf{s}(t),\mathbf{a}\right)\right]$$

Utilizing the optimality conditions in (8)-(10), an online Q-learning-based solver for finding $\{\pi^*(\mathbf{s}),Q^*(\mathbf{s},\mathbf{a}),V^*(\mathbf{s})\}$ is introduced in the ensuing subsection.

A. Optimal proactive caching via Q-learning

Q-learning is an online reinforcement learning method to jointly estimate the optimal state-action pair value function $Q^*(\mathbf{s},\mathbf{a})\ \forall \mathbf{s},\mathbf{a}$, and learn the optimal policy π^*. The proposed Q-learning procedure for proactive caching is outlined in the pseudocode tabulated as Algorithm 1. In this algorithm, the agent updates its estimated $\hat{Q}(\mathbf{s}(t),\mathbf{a}(t+1))$ as $C\left(\mathbf{s}(t),\mathbf{a}(t+1)|\,\mathbf{p}_{\mathrm{G}}(t+1),\mathbf{p}_{\mathrm{L}}(t+1)\right)$ is observed. That is, given state $\mathbf{s}(t)$, Q-learning takes action $\mathbf{a}(t+1)$, observes the new state $\mathbf{s}(t+1)$, incurs immediate cost of $C\left(\mathbf{s}(t),\mathbf{a}(t+1)|\,\mathbf{p}_{\mathrm{G}}(t+1),\mathbf{p}_{\mathrm{L}}(t+1)\right)$, and finally updates its estimated $\hat{Q}(\mathbf{s}(t),\mathbf{a}(t+1))$ while keeping the rest of the entries in $\hat{Q}(\cdot,\cdot)$ unchanged. Regarding convergence of the Q-learning algorithm, i.e., $\hat{Q}\left(\cdot,\cdot\right)\to Q^*\left(\cdot,\cdot\right)$, a necessary condition is that all state-action pairs must be continuously updated. Under this assumption and a variant of the usual stochastic approximation conditions, $\hat{Q}\left(\cdot,\cdot\right)$ converges to $Q^*\left(\cdot,\cdot\right)$ with probability 1 [12].

To meet the former condition, Q-learning utilizes a probabilistic *exploration-exploitation* approach in action selection. That is, at time t, exploitation happens with probability $1-\epsilon_t$ where action $\mathbf{a}(t)=\arg\min_{\mathbf{a}\in\mathcal{A}}\hat{Q}(\mathbf{s}(t-1),\mathbf{a})$ as the *anticipated* optimal action is chosen, and the exploration happens with probability ϵ_t where SB takes a random exploratory action $\mathbf{a}\in\mathcal{A}$. Parameter ϵ_t tunes exploration versus exploitation, guaranteeing the necessary condition for convergence. During initial iterations or when the CCU observes considerable shifts in content popularities, setting ϵ_t high promotes exploration in order to learn the underlying dynamics. On the other hand, in stationary settings and once "enough" observations are made, small values of ϵ_t promote exploiting the learned $\hat{Q}(\cdot,\cdot)$ by taking the estimated optimal action $\arg\min_{\mathbf{a}}\hat{Q}\left(\mathbf{s}(t),\mathbf{a}\right)$.

TABLE I
COST PARAMETERS

Scenario	λ_1	λ_2	λ_3
(s1)	10	600	1000
(s2)	600	10	1000
(s3)	10	10	1000
(s4)	0	1000	0
(s5)	0	0	1000

IV. NUMERICAL TESTS

In this section, the performance of the proposed proactive caching algorithm is assessed via numerical tests. We have simulated a setup with the total of $F=10$ contents and a caching capacity of $M=3$ at the local SB. Moreover, global popularity profile $\mathbf{p}_G(t)$ is modeled via a two-state Markov chain, represented by two popularity profiles $\mathbf{p}_G^{(1)}$ and $\mathbf{p}_G^{(2)}$, each modeled with a Zipf distribution with parameters $\eta_1^G=1$ and $\eta_2^G=1.5$, respectively. That is, for state $i\in\{1,2\}$, the F contents are assigned a random ordering of popularities and then sorted accordingly in a descending order. Given this ordering and the Zipf distribution parameter η_i^G, the popularity of the f-th content is equal to

$$\left[\mathbf{p}_{\mathrm{G}}^{(i)}\right]_f=\frac{1}{f^{\eta_i^G}\sum_{l=1}^{F}\frac{1}{l^{\eta_i^G}}}\quad\text{for }\ i=1,2.$$

The summation term in the denominator normalizes the components into a valid probability mass function, and parameter $\eta_i^G\geq 0$ controls the skewness of the popularities. Specifically, $\eta_i^G=0$ yields a uniform spread of popularity among the contents, while a large value of η_i^G generates more skewed popularities.

The Markov transition probabilities are given by the transition matrix

$$\boldsymbol{\tau}:=\begin{bmatrix}\tau_{1,1} & \tau_{1,2}\\ \tau_{2,1} & \tau_{2,2}\end{bmatrix}=\begin{bmatrix}0.6 & 0.4\\ 0.45 & 0.55\end{bmatrix}$$

where $\tau_{i,j}$ indicates transition probability form state i to j, $i,j\in\{1,2\}$.

Similarly, we consider a two-state Markov chain with Zipf parameters $\eta_1^L=1.2$ and $\eta_2^L=1.7$ to model the local content popularities. The state transition matrix $\boldsymbol{\tau}'$ for local popularity profile is set as

$$\boldsymbol{\tau}':=\begin{bmatrix}\tau'_{1,1} & \tau'_{1,2}\\ \tau'_{2,1} & \tau'_{2,2}\end{bmatrix}=\begin{bmatrix}0.35 & 0.65\\ 0.75 & 0.25\end{bmatrix}.$$

In the utilized Q-learning algorithm, β_t is set to a constant $\beta=0.8$, and the exploration-exploitation parameter $\epsilon_t=0.05$. The proposed algorithm is run with different cost parameters reported in table. I and the performance is averaged over 1000 independent realizations of the system for each setting.

Fig. 2 plots the evolution of cost versus iteration index for the proposed approach, demonstrating its convergence to the cost of optimal offline policy derived with *known* transition probabilities. Slower convergence under (s1) is because of

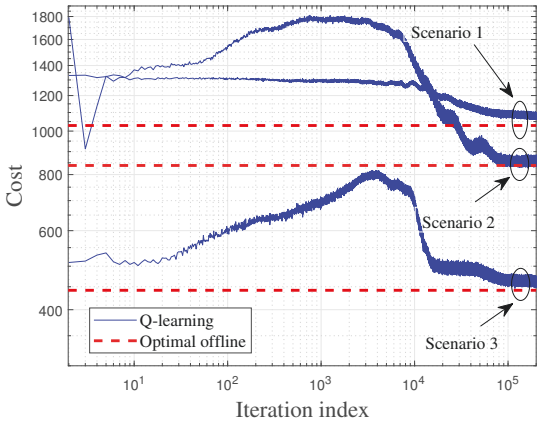

Fig. 2. Evolution of the overall cost versus iteration index

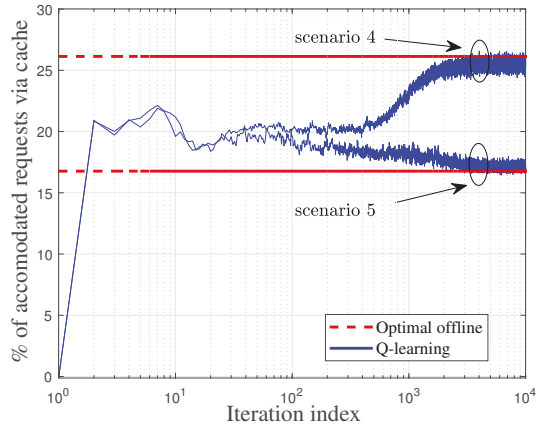

Fig. 3. Percentage of accommodated requests via cache vs. iteration index

the relatively high local and global mismatch cost parameters which necessitates learning both the global and local popularity dynamics in order to find the optimal policy. In contrast, under (s2) λ_2 corresponding to local popularity is low, thus influence of local popularity on optimal policy is reduced, and faster convergence is obtained. Finally setting $\lambda_1 = 10$, further reduces cache refreshing cost and more importance falls on learning global popularity Markov chain, and a slightly faster convergence is obtained compared to (s2).

Furthermore, Fig. 3 plots the percentage of requests served directly via the locally-available cached contents. Observe that penalizing local-popularity mismatches in (s4) forces the caching policy to adapt to local request dynamics, while (s5) prioritizes tracking global popularities, leading to a lower cache hit rate.

All in all, the conducted simulation tests illustrate the significance and effectiveness of the proposed online RL-enabled caching, implemented locally in the SBs. To endow the algorithm with scalability, and thus enable implementation in real network scenarios involving large-scale Markov chains where the size of contents can be very large, function approximation schemes can be considered instead. An extended version of this work focusing on linear function approximation can be found in [13].

V. Conclusion

The present work addresses caching in Next-G cellular networks, in which local and global content popularity profiles exhibit spatio-temporal dynamics. In this context, proactive caching is accommodated by casting the problem in a reinforcement learning framework, for which an online solver with optimality guarantees is proposed. Distributed and scalable approaches for tackling the curse of dimensionality, e.g., parametric and non-parametric techniques for Q-function approximation, are among the future directions we will pursue.

References

[1] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed, "Femtocells: Past, present, and future," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 497–508, April 2012.

[2] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, August 2016.

[3] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, April 2013.

[4] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[5] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *IEEE Inte. Conference on Communications*, June 2014, pp. 1897–1903.

[6] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. of 11th Inte. Symposium on Wireless Communications Systems*, August 2014, pp. 917–921.

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

[8] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, April 2016.

[9] H. Kim, J. Park, M. Bennis, S.-L. Kim, and M. Debbah, "Ultra-dense edge caching under spatio-temporal demand and network dynamics," *arXiv preprint arXiv:1703.01038*, 2017.

[10] S. Mller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, February 2017.

[11] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, December 2016.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[13] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, February 2018.