Joint Cache Placement and Delivery Design using Reinforcement Learning for Cellular Networks

Mohsen Amidzadeh*, Hanan Al-Tous*, Olav Tirkkonen* and Junshan Zhang†
*Department of Communications and Networking, Aalto University, Espoo, Finland
†Arizona State University, Tempe Arizona
{mohsen.amidzade, hanan.al-tous, olav.tirkkonen}@aalto.fi, Junshan.Zhang@asu.edu

***Abstract*—We consider a reinforcement learning (RL) based joint cache placement and delivery (CPD) policy for cellular networks with limited caching capacity at both Base Stations (BSs) and User Equipments (UEs). The dynamics of file preferences of users is modeled by a Markov process. User requests are based on current preferences, and on the content of the user's cache. We assume probabilistic models for the cache placement at both the UEs and the BSs. When the network receives a request for an un-cached file, it fetches the file from the core network via a backhaul link. File delivery is based on network-level orthogonal multipoint multicasting transmissions. For this, all BSs caching a specific file transmit collaboratively in a dedicated resource. File reception depends on the state of the wireless channels. We design the CPD policy while taking into account the user Quality of Service and the backhaul load, and using an Actor-Critic RL framework with two neural networks. Simulation results are used to show the merits of the devised CPD policy.**

***Index Terms*—Wireless caching, cache placement and delivery policy, multipoint multicast transmission, reinforcement learning, Actor-Critic, neural network.**

I. Introduction

The unprecedented traffic growth in cellular networks has led to network congestion problems. Caching at the network edge is a promising method to alleviate traffic congestion problems [1]. In the literature, several models have been studied for cache deployment, placement, and delivery. In [2], [3], the authors exploit stochastic geometry and Poisson Point Process (PPP) for the deployment of caches across the network. In [3], [4], a probabilistic caching model is considered to store the files at Base-stations (BS). In [5], the authors consider an orthogonal transmission where bandwidth is allocated to requesting User Equipments (UE) that are served by their nearest BS. To avoid inter-cell interference, multipoint multicasting for cache delivery is considered in [6].

In recent years, reinforcement learning (RL) has been considered for cache policy design [7]–[10]. In [7], a RL framework is used for cache placement in a network with dynamically changing file preferences. Q-learning agents located at individual BSs are used to find the optimum cache policy. In [8], coded caching at fog access points is considered, and a deep RL (DRL) approach is developed to optimize cache placement for static file preferences. In [9], an Actor-Critic DRL algorithm is exploited to jointly find an optimum policy for user scheduling and cache placement in a heterogeneous network with static file preferences. In [10], a DRL-based method is leveraged to achieve optimum caching in term of average transmission delay for a cellular network with cooperative BSs. All of these works consider a *deterministic* caching model, where the individual cache contents of nodes are acted on.

To model the dynamics of user preferences, [11] assumes that popular files change according to a Markov model. In [7], [12], [13], a Markov process with an unknown transition matrix is used to model popularity dynamics.

In this paper, we apply caching both in the network *and* at UEs, assuming *probabilistic models* for cache placement. We model UE and BS deployment as two independent homogeneous PPP. A finite state Markov process is used to model the dynamics of user preferences. Each UE requests files based on its current preference and the status of files stored in its cache. Requested files are delivered by network-level orthogonal multipoint multicasting (OMPMC) transmissions. In this scheme, the network transmits a file towards requesting UEs in the same frequency resources. To transmit different files, network-wide disjoint resources are reserved. The cache placement and delivery (CPD) policy for the whole network is learned based on a RL framework, aiming to maximize Quality of Service (QoS) experienced by UEs, and minimize backhaul load. Neural Networks (NNs) are used to represent the agent, which observes the system state and interacts with the environment through following the NN-provided policy.

The main contribution of this paper is as follows. In contrast to prior works [7], [9], [12], we utilize a probabilistic approach to store the files in limited caches of BSs and UEs that merely includes overall probability vectors and no parameters corresponding to individual users. An NN-based RL framework is exploited to jointly optimize the cache placement and delivery. For this, the CPD problem is modeled as a Markov decision process (MDP). We concentrate on A2C [14] to find the optimal CPD policy. The optimum policy is obtained in term of file delivery and backhaul loads. To enforce the constraints related to the caching policy and trade off between exploitation and exploration, we modify the actor network of A2C algorithms by applying Dirichlet distributions.

The remainder of this paper is organized as follows. In Section II, the system model and in Section III, the problem formulation are introduced. The RL-based CPD method is presented in Section IV. Simulations are shown and discussed in Section V. Section VI concludes the paper.

II. System Model

We consider a cellular access network with an associated core network. BSs fetch files from the core network via error-free backhaul links. Both UEs and BSs are equipped with independent storage-limited caches that store files according to probabilistic models. In the delivery phase, users choose files based on file preferences. If a user chooses a file that is stored in its cache, it can directly obtain it, if it is not locally cached, the user requests the file from the network. Based on the aggregate requests from the users, the network updates BS caches, and transmit the files over the air interface. We model UE and BS locations as two independent homogeneous PPPs, which leads to a statistical description of file delivery success. Time slotted operation is assumed; changes in file preferences, file requests by users, file fetching by BSs from the core network, and file delivery towards UEs are modeled on a per-slot basis, with slots indexed by t.

A. File Popularity and User Requests

In each time-slot, users choose files from a library $\mathcal{F}$ containing N files. Without loss of generality, files are assumed to have the same size normalized to 1. The popularity of file n at time-slot t is $p_n(t) \in [0,1]$. This distribution is not assumed to be a priori known, it emerges from the aggregate user behavior at time-slot t. For simulations, a Zipf distribution [15] is used to model file popularity, $p_n(t) = n^{-\tau(t)} / \sum_{j=1}^{N} j^{-\tau(t)}$ for $j = 1, \dots, N$, where $\tau(t)$ stands for the skewness of the Zipf distribution at time-slot t. Moreover, file popularity evolution is modeled as a finite state Markov chain [11] with M states and state set $\mathcal{Q} := \{Q_1, \dots Q_M\}$. The unknown transition probability matrix is denoted by $\mathbf{Q}$.

Note that we use a Markov process and Zipf distribution merely for simulating the popularity profile, following [11]–[13]. The network does not know the popularity distribution; network optimization is based on the realized requests of users in different time slots. The considered caching policy is thus agnostic to the file popularity model.

Caching of files at UEs is based on a probabilistic method. The UE caching probabilities $s_n(t)$ determine the probability that file n is stored at a UE cache in time-slot t. For instance, if $s_n = 1/3$, approximately one third of UEs store file n while others do not store it. We assume that the cache of each UE has maximum capacity L_u, i.e, $\sum_{n=1}^{N} s_n(t) \leq L_u$. A UE preferring a file only requests it from the network if the file has not been stored at its cache. Accordingly, the probability that a given UE requests a file at time-slot t becomes

$$\mathbf{r}^{\text{net}}(t) = \mathbf{p}(t) \odot (\mathbf{1} - \mathbf{s}(t)), \tag{1}$$

this is a vector with one entry per file, $\mathbf{r}^{\text{net}}(t) = [r_1^{\text{net}}(t), \dots, r_N^{\text{net}}(t)]^\top$, and $\mathbf{s}(t) = [s_1(t), \dots, s_N(t)]^\top$ is a vector of the UE caching probabilities, $\mathbf{p}(t) = [p_1(t), \dots, p_N(t)]^\top$ a vector of file popularities, and $\mathbf{1}$ is an $N \times 1$ vector with all elements equal to one. The element-wise vector multiplication of two vectors is denoted as $\odot$, and $\top$ denotes transposition. File request probability decreases with increasing caching probability, and increases with increasing preference probability.

B. Cache Placement

Files are cached at BSs based on a probabilistic model. There is an overall caching probability vector $\boldsymbol{\rho}(t) = [\rho_1(t), \dots, \rho_N(t)]^\top$, where $\rho_n(t)$ indicates the probability that file n is cached at a BS. We assume that the BS caches have a maximum capacity L_c, such that $\sum_{n=1}^{N} \rho_n(t) \leq L_c$.

C. Cache Delivery

After updating the caches of BSs, files are transmitted towards the UEs in dedicated resources with bandwidth $w_n(t)$, using network orthogonal multipoint multicast transmission. Each BS caching the file participates in the multipoint transmission. Note that in this cache delivery principle, the network responds to the aggregate requests of the whole user population by an aggregate transmission of all files, where a Single-Frequency Network [16] is formed by the BSs caching the file. If the signal-to-noise ratio (SNR) associated to a file at a requesting UE is less than a threshold, the request will be in outage. The outage probability depends on the parameters of CPD policy (i.e. the caching probability $\boldsymbol{\rho}(t)$ and allocated bandwidths $\mathbf{w}(t)$).

The outage probability for a network model described by a PPP can be derived by following the analysis of [2]. For a network-level orthogonal multipoint multicast transmission scheme in an environment with path-loss exponent $\beta = 4$, with BSs distributed according to a PPP with intensity λ, with file n from the library $\mathcal{F}$ being cached with probability $\rho_n(t)$ and having allocated bandwidth $w_n(t)$, the probability that file n is in outage at a user is [6]

$$\mathcal{O}_n(t) = \operatorname{erfc}\left(\frac{\pi^2 \lambda \rho_n(t)}{4\sqrt{2\eta_n(t)}}\right). \tag{2}$$

Here $\operatorname{erfc}(\cdot)$ is the complementary error function, and channel gain threshold $\eta_n(t) = \frac{\sigma_0^2}{\bar{p}}(2^{\alpha/w_n(t)} - 1)$ are expressed in terms of a target spectral efficiency α, the average BS transmit power density $\bar{p}$, and the UE noise power spectral density σ_0^2.

D. UE Caching Policy

The caching probability of the UEs for the next time-slot depends on the current caching probability and the outage probability. The files already cached, and the files that the UE can decode, are candidates for caching. If the number of candidate files exceeds the UE caching capacity L_u, the UE chooses L_u files among the candidates uniformly at random. On the population level, this leads to the caching probability update equation

$$\begin{aligned} \tilde{\mathbf{s}}(t+1) &= \mathbf{s}(t) + (\mathbf{1} - \mathbf{s}(t)) \odot (\mathbf{1} - \boldsymbol{\mathcal{O}}(t)), \\ \mathbf{s}(t+1) &= \min\left\{1, \frac{L_u}{\sum_{n=1}^{N} \tilde{s}_n(t+1)}\right\} \tilde{\mathbf{s}}(t+1), \end{aligned} \tag{3}$$

where $\boldsymbol{\mathcal{O}}(t) = [\mathcal{O}_1(t), \dots, \mathcal{O}_N(t)]^\top$.

E. Network Operation Timing Model

In equations (1)-(3), the problem has been formulated based on probabilistic vectors $\mathbf{r}^{\text{net}}(t)$, $\mathbf{s}(t)$ and $\boldsymbol{\rho}(t)$ instead of vectors related to individual users and BSs. The joint CPD policy update during time-slot t consists of three consecutive phases: the request announcement, cache update and placement and the cache delivery. The interactions between UEs, BSs and the core network during time-slot t are illustrated in Figure 1.

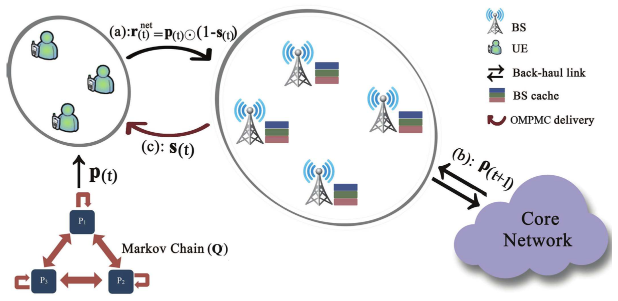
(a): $\mathbf{r}^{\text{net}}_{(t)} = \mathbf{p}_{(t)} \odot (1-\mathbf{s}_{(t)})$
(c): $\mathbf{s}_{(t)}$
$\mathbf{p}_{(t)}$
P₁
P₂
P₃
Markov Chain ($\mathbf{Q}$)
(b): $\boldsymbol{\rho}_{(t+1)}$
Core Network
BS
UE
Back-haul link
BS cache
OMPMC delivery

Fig. 1. The interactions between UEs, BSs and core network for CPD policy. Phase (a): UEs requests. Phase (b): BSs fetch files from the core network and update their caches. Phase (c): OMPMC cache delivery and update of UEs caches.

III. Cache Placement and Delivery Optimization Problem

Users request files based on their preference and cache status. Therefore, the file request probability (1) and the UE caching probability (3) constitute the system state vector.

$$\mathbf{x}(t) = [\mathbf{r}^{\text{net}}(t)^\top,\ \mathbf{s}(t)^\top]^\top. \tag{4}$$

The feasible set for the state vector is

$$\mathcal{X} = \mathcal{R} \times \mathcal{S}, \tag{5}$$

where $\mathcal{S} = \{\mathbf{s} \mid \mathbf{s} \geq \mathbf{0},\ \mathbf{1}^\top \mathbf{s} \leq L_u\}$, and $\mathcal{R} = \{\mathbf{r} \mid \mathbf{r} \geq \mathbf{0},\ \mathbf{1}^\top \mathbf{r} \leq 1\}$. Requesting UEs are responded by the whole network according to the CPD policy. The network action vector is constituted by the caching probability of BSs and the allocated bandwidths:

$$\mathbf{u}(t) = [\boldsymbol{\rho}(t)^\top,\ \mathbf{w}(t)^\top]^\top. \tag{6}$$

The feasible set of actions is constrained by the capacity of BS caches and the total available bandwidth;

$$\mathcal{U} = \{(\boldsymbol{\rho}, \mathbf{w}) | \boldsymbol{\rho} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{\rho} \leq L_c,\ \mathbf{w} \in [0,1]^N, \mathbf{1}^\top \mathbf{w} = 1\}. \tag{7}$$

We are interested in high QoS achieved with small backhaul load. Instead of minimizing the backhaul load with a QoS constraint, or vice versa, we formulate a joint optimization problem.

We take the availability probability of preferred files at UEs as a measure of QoS. For this, the probability that a requesting user gets the file from the network is considered. Considering all files, the reward should be averaged over the file popularity. This QoS reward function is expressed in terms of the request probability (1) and the outage probability (2) as

$$\begin{aligned} r_{\text{QoS}}(t) &= 1 - \sum_{n=1}^{N} p_n(t)\big[s_n(t) + (1 - s_n(t))(1 - \mathcal{O}_n(t))\big] \\ &= 1 - \sum_{n=1}^{N} r_n^{\text{net}}(t)\mathcal{O}_n(t)\ . \end{aligned} \tag{8}$$

For example, if file n is already cached at all users, based on (8) the reward from this file is p_n, and the outage probability of this file is irrelevant. Conversely, if a file is not cached at any user, and UEs request the file, outage directly reduces the QoS reward from this file.

The backhaul load is related to the number of files fetched from the core network. If $\rho_n(t) - \rho_n(t-1) \leq 0$, no BS needs to fetch file n in time-slot t, and only some BSs may have to delete the file from their cache. If $\rho_n(t) - \rho_n(t-1) > 0$, a corresponding fraction of BS have to fetch the file, using the backhaul network. The probability that a BS has to fetch file n is thus given by $[\rho_n(t) - \rho_n(t-1)]_+$, where $[x]_+ = \frac{1}{2}(x + \text{abs}(x))$. Hence, the aggregate backhaul reward is shaped as

$$r_{\text{bh}}(t) = L_c - \sum_{n=1}^{N} [\rho_n(t) - \rho_n(t-1)]_+\,. \tag{9}$$

A joint optimization problem can be formulated using a Lagrange multiplier, which indicates the relative cost of backhaul and QoS as:

$$r(t) = r_{\text{QoS}}(t) + \lambda_{\text{bh}} r_{\text{bh}}(t), \tag{10}$$

where λ_{bh} is a Lagrange multiplier that regulates between user satisfaction and backhaul load. A discounted average reward starting from time-slot t into the future can then be defined as:

$$R(t) = \sum_{k=t}^{T} \gamma^{k-t} \mathbb{E}[r(k)], \tag{11}$$

where T is number of time-slots in the caching process and $\gamma \in [0,1]$ is the discount factor.

Considering the dynamics caused by the dependencies of actions and states in different time-slots, we are interested in maximizing $R(t)$ starting from any time-slot t. The CPD policy design is formulated as a constrained optimization problem:

$$\begin{aligned} \text{P1}: \quad & \max_{\{\mathbf{w}(k), \boldsymbol{\rho}(k)\}_{k\geq t}} \quad R(t),\ \ 0 \leq t \leq T \\ & \text{s.t.} \begin{cases} \{\mathbf{w}(k), \boldsymbol{\rho}(k)\}_{k\geq t} \in \mathcal{U}, \\ (1)-(3), (8)-(11). \end{cases} \end{aligned} \tag{12}$$

Let $\pi(\cdot|\mathbf{x}(t))$ denote the policy distribution by which the action $\mathbf{u}(t)$ are generated given state $\mathbf{x}(t)$. By following this policy and observing the resulting state at different time-slots, a trajectory is produced with the transition probability function $p\big(\mathbf{x}(t+1)|\mathbf{x}(t), (\mathbf{u}(t)\big)$ that describes the dynamics of the environment. The goal is to learn a policy $\pi\big(\cdot\,|\mathbf{x}(t)\big)$ which maximizes $R(t)$ starting from any time-slot t.

IV. A Reinforcement Learning Framework

We create a RL agent for finding actions according to dynamically changing network state. The objective of the agent is to find an optimal policy $\pi\big(\mathbf{u}(t)|\mathbf{x}(t)\big)$, such that P1 is solved. Although there are many function approximation techniques to represent the policy, we use neural network approach that can circumvent the "curse of dimensionality" [17]. Among RL algorithms such as Policy-Gradient (PG), Advantageous Actor-Critic (A2C), and Proximal Policy Optimization (PPO), we concentrate on A2C [14] to find the optimal CPD policy. The reason is that for our problem, PG fails to correctly learn parameters related to policy representation and PPO often gets stuck in a local optimum. Here, the actor and critic of A2C algorithm are represented by two neural networks. The actor network stands for a RL agent producing policies by which it can interact with the environment. The policy generated by this network is represented by $\pi_{\boldsymbol{\theta}}(\cdot)$, where $\boldsymbol{\theta}$ stands for the parameters of the actor network. On the other side, the critic network gives a representation of the value-function or discounted average reward conditioned on starting from a specific state $\mathbf{x}(t)$. The value-function starting from state $\mathbf{x}(t)$ provides an estimate of

$$V_{\boldsymbol{\phi}}(\mathbf{x}(t)) = \mathbb{E}\left[\sum_{k=t}^{T} \gamma^{k-t} r(k) \mid \mathbf{x}(t)\right], \tag{13}$$

where $\boldsymbol{\phi}$ stands for the parameters of the critic network.

A. Learning via A2C

An episodic trajectory is produced by the actor network as follows. For the current state, the actor network produces an action based on the parameters $\boldsymbol{\theta}$. By interacting with the environment using this action, an immediate reward and the next state are generated. The critic network then gives an estimate of the value-function using the current state, and based on the parameters $\boldsymbol{\phi}$. Afterwards, the produced action, the immediate reward and the value-function are stacked in a buffer to be used in the future to update the parameters of actor and critic networks. By repeating this procedure, we can buffer information until time-slot T.

After initialization, the parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$ of the actor and critic networks are updated via A2C algorithm [14] as follows

$$\begin{aligned}
\boldsymbol{\phi} &\leftarrow \boldsymbol{\phi} + \delta\left[\sum_{t}^{T} A_{\boldsymbol{\phi}}(\mathbf{u}(t), \mathbf{x}(t))\ \nabla_{\boldsymbol{\phi}} V_{\boldsymbol{\phi}}(\mathbf{x}(t))\right], \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \psi\left[\sum_{t}^{T} A_{\boldsymbol{\phi}}(\mathbf{u}(t), \mathbf{x}(t))\ \nabla_{\boldsymbol{\theta}} \log\left(\pi_{\boldsymbol{\theta}}(\mathbf{u}(t)|\mathbf{x}(t))\right)\right. \\
&\qquad \left. + \beta_{\text{ent}} \underbrace{\sum_{t}^{T} \nabla_{\boldsymbol{\theta}} H\left(\pi_{\boldsymbol{\theta}}(\mathbf{u}(t)|\mathbf{x}(t))\right)}_{\text{Entropy term}}\right],
\end{aligned} \tag{14}$$

where the advantage value is given by $A_{\boldsymbol{\phi}}(\mathbf{u}(t), \mathbf{x}(t)) = r(t+1) + V_{\boldsymbol{\phi}}(\mathbf{x}(t+1)) - V_{\boldsymbol{\phi}}(\mathbf{x}(t))$. The term with the entropy function $H(\cdot)$ is added in order to trade off exploration against exploitation by discouraging premature convergence to suboptimal deterministic policies [18]. The parameters ψ and δ are the learning rates corresponding to update of actor and critic networks, respectively, and β_{ent} is the entropy regularization term. The Figure 2 illustrates the A2C algorithm. As Figure 2 and (14) imply, the parameters of both actor and critic network are updated using the advantage value generated by the critic network.

Assuming that cache placement and cache delivery policies can be independently updated, we have

$$\begin{aligned}
\log\left(\pi_{\boldsymbol{\theta}}(\mathbf{u}(t)|\mathbf{x}(t))\right) = &\log\left(\pi_{\boldsymbol{\theta}}(\boldsymbol{\rho}(t)|\mathbf{x}(t))\right) \\
&+ \log\left(\pi_{\boldsymbol{\theta}}(\mathbf{w}(t)|\mathbf{x}(t))\right).
\end{aligned} \tag{15}$$

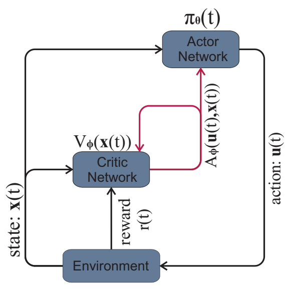

Fig. 2. The diagram of A2C algorithm. The red arrows shows the update process.

B. Structure of Actor and Critic Networks

The actor and critic are designed using two separate neural networks. One hidden layer is used for the critic network and the rectified linear unit (ReLU) activation function is used for each neuron. Figure 3 depicts the structure of the neural network representing the actor agent. It has one hidden layer with hyperbolic tangent activation functions. The output layer is acted on by a *softplus* function. The standard actor network of the A2C considers a Gaussian distribution to generate the action vector. However, to enforce the constraints in (7) over the action vector, we modify the actor network and consider Dirichlet distributions to generate the action vector as:

$$\mathbf{u}(t) \sim \begin{bmatrix} L_c \cdot \text{Dirichlet}\left(\boldsymbol{\rho}'(t, \boldsymbol{\theta})\right) \\ \text{Dirichlet}\left(\boldsymbol{w}'(t, \boldsymbol{\theta})\right) \end{bmatrix},$$

where $\boldsymbol{\rho}'(t, \boldsymbol{\theta})$ and $\boldsymbol{w}'(t, \boldsymbol{\theta})$ are the outputs of the actor network parametrized by $\boldsymbol{\theta}$, and $\text{Dirichlet}(\cdot)$ stands for the multivariate Dirichlet distribution. The reason behind selecting this structure is that the elements of any random vector generated from a Dirichlet distribution lie between zero and one, and the sum of its elements equals to one. By this modification, the constraints in (7) over the action vector at each time-slot are enforced and the entropy term in (14) is computed without additional computational complexity.

Algorithm 1 shows the pseudo code for the proposed RL-based CPD policy based on modified A2C. In the pseudo-code, $E_{\max}$ is the maximum number of episodes for which the training process is performed and N_u is the number of time-slots after which the actor and critic networks are updated.

V. Simulation and Discussion

To investigate RL-based CPD, extensive simulations have been conducted. For this, we consider a cellular network, with

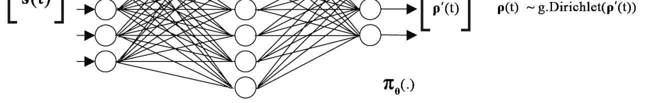
state: $\mathbf{x}(t)$
$\begin{bmatrix}\mathbf{p}(t)\\ \mathbf{s}(t)\end{bmatrix}$
$\boldsymbol{\theta}_{xh}$
$F_h(.) = \tanh(.)$
$\boldsymbol{\theta}_{hu}$
$F_u(.)$=softplus(.)
Output
$\mathbf{w}'(t)$
$\boldsymbol{\rho}'(t)$
action: $\mathbf{u}(t)=[\boldsymbol{\rho}(t),\mathbf{w}(t)]$
$\mathbf{w}(t) \sim$ Dirichlet($\mathbf{w}'(t)$)
$\boldsymbol{\rho}(t) \sim$ g.Dirichlet($\boldsymbol{\rho}'(t)$)
$\boldsymbol{\pi}_{\boldsymbol{\theta}}(.)$

Fig. 3. The structure of the actor neural network $\pi_{\boldsymbol{\theta}}(.)$.

the intensity of BSs $\lambda = 50$ and average BS transmission power density $\bar{p} = 1$. The UE noise power is $\sigma_0^2 = 1$ and the target spectral efficiency $\alpha = 1/5$ as in [6]. Like [7], [19], we consider a Markov process to model the preference profile. The popularity profile is modeled by a four-state Markov chain with transition matrix

$$\mathbf{Q} = \begin{pmatrix} 0.9 & 0.033 & 0.033 & 0.033 \\ 0.2 & 0.7 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.85 & 0.05 \\ 0.1 & 0.1 & 0.1 & 0.7 \end{pmatrix}, \tag{16}$$

and the corresponding state set contains Zipf distributions with skewness values $\tau = \{2, 2.5, 3, 3.5\}$. We apply four different popularity ranks of the N files, one for each state of the Markov process. Hence, file popularity and rank change during time-slots. Note that the RL-based approach for the cache policy is not restricted by the model of popularity we considered here, and we use a Markov process and Zipf distribution as a typical model. We have $N = 40$ files in the library, the capacity of BSs caches are $L_c = 6$, and the capacity of UEs caches are $L_u = 3$. The total number of time-slots is $T = 256$.

Several RL algorithms are tested and the best results are obtained by the modified A2C algorithm. For this algorithm, the hyperparameters are tuned as follows. For the actor and critic networks, the number of neurons in the hidden layer is 64. Increasing the number of neurons does not give better performance in terms of accumulative reward, while it needs more data for tuning the parameters of the networks. We set $E_{\max} = 2.5 \times 10^4$ and $N_u = 128$. We use $\lambda_{\text{bh}} = 0.05$ to regulate the cost of backhaul, $\gamma = 0.98$ for the discount factor and $\beta_{\text{ent}} = 10^{-2}$ for regularization of the entropy term. The learning rates of the actor and critic networks are $\psi = 7 \times 10^{-3}$ and $\delta = 7 \times 10^{-3}$.

Figure 4 shows the training performance of the RL agent in terms of the accumulative reward $R_{\text{Ac}}(t) = \sum_{k=1}^{t} r(k)$, and QoS and backhaul terms of accumulative reward, i.e. $R_{\text{Ac,QoS}}(t) = \sum_{k=1}^{t} r_{\text{QoS}}(k)$ and $R_{\text{Ac,bh}}(t) = \sum_{k=1}^{t} r_{\text{bh}}(k)$. As this figure illustrates, the RL agent is trained after approximately 1.5×10^4 episodes for the aforementioned cellular network parameters.

In order to compare and benchmark our result, we compare to the so-called "Least Recently Used" (LRU) and "Least Frequently Used" (LFU) caching policies from the literature

Algorithm 1 The Modified A2C for CPD design.

```
1:  for $episode = 1$ to $E_{\max}$ do
2:      Given the initial system state vector $\mathbf{x}(1)$, actor and critic
        networks parametrized with $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.
3:      for $t = 1$ to $T$ do
4:          Select an action $\mathbf{u}(t)$ following $\pi_{\boldsymbol{\theta}}(\mathbf{x}(t))$ and interact with
            the environment.
5:          Critic network provides estimate of value-function $V_{\boldsymbol{\phi}}(\mathbf{x}(t))$.
6:          Observe new state $\mathbf{x}(t+1)$ and immediate reward $r(t)$.
7:          Buffer $V_{\boldsymbol{\phi}}(\mathbf{x}(t))$, $\mathbf{x}(t)$, $r(t)$, $\log\left(\pi_{\boldsymbol{\theta}}(\mathbf{u}(t)|\mathbf{x}(t))\right)$, and
            $H\left(\pi_{\boldsymbol{\theta}}(\mathbf{u}(t)|\mathbf{x}(t))\right)$.
8:          Update parameters of actor and critic networks as in (14)
            every $N_u$ time-slot.
9:      end for
10: end for
```

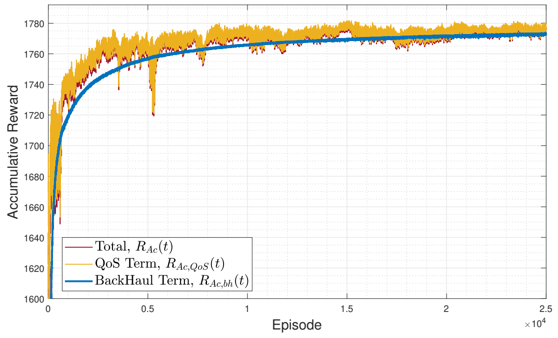

Fig. 4. Training performance of the RL agent in term of the accumulative reward.

[20], as well as to the optimal method. LRU is based by each BS tracking the most recent request and removing the least recently used file from the cache when it is full, while LFU is based on keeping the most frequently requested files in the cache of BSs. To find the optimal method, we use the *interior-point* algorithm to solve the optimization problem. For all of these methods, the accumulative reward is computed to obtain the performance benchmark.

Figure 5 shows the comparison of test performance between RL-based proposed CPD policy method, LRU, LFU and "*Optimum*" for 128 Episodes in term of the accumulative reward. As this figure shows, the performance of the RL-based CPD policy method outperforms other cache policy approaches. The "*Optimum*" policy slightly outperforms RL-based policy, at a considerable complexity cost. The RL-based method can be immediately leveraged whenever the actor and critic networks are fully trained. For this, it suffices for the agent to feed the measured system state into the actor network and read its output as the optimum cache policy.

The experimental cumulative distribution function (CDF) of the instantaneous QoS reward (8) across time-slots is plotted in Figure 6. This shows the statistics of QoS experienced by users during the network dynamics after the agent is fully trained. Note that the instantaneous QoS reward indicates the probability that a user gets the needed file. The closer it is to one, the more probable it is that users have either cached,

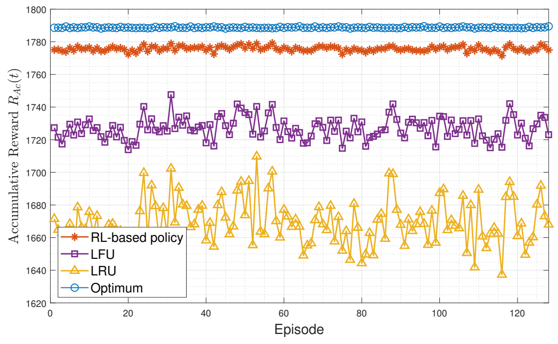

Fig. 5. Performance comparison between the RL-based CPD, LFU, LRU and "*Optimum*" cache policies in term of the accumulative reward.

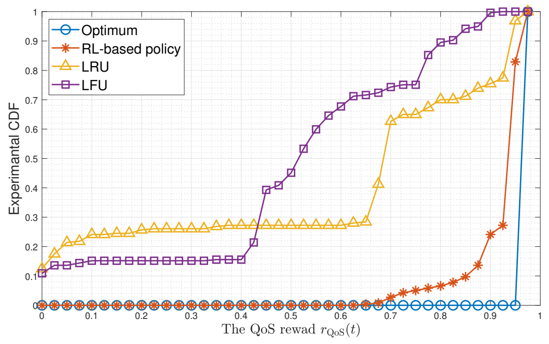

Fig. 6. Experimental CDF of the QoS reward during network dynamics.

or have successfully received their files of interest in a given slot. For the RL-based and "Optimum" policies, the CDF is aggregated on values greater than 0.9, meaning that users are satisfied with high probability. However, for the LRU and LFU policies the CDF is distributed over a greater range meaning that in some slots the network fails to fulfil user requests.

The experimental CDF corresponding to the other variable of the trade-off, instantaneous backhaul reward (9), is sketched in Figure 7. The smaller it is, the more costly it is to fetch the files from the core network. For the RL-based and "Optimum" policies, the CDF is concentrated near the maximum reward $L_c = 6$, implying that in each time-slot these policies have proactively cached at BSs most files that will be requested by users in the next time-slot, and the network will not need to fetch many files. However, for the LRU and LFU policies, the files fail to be cached at BSs proportional to user requests which causes backhaul load for the network.

VI. Conclusion

In this paper, we formulated the cache placement and cache delivery policy design problem based on a reinforcement learning framework. We shaped a reward function aiming to optimize the user QoS and backhaul load. We utilized an Actor-Critic algorithm and neural networks to learn optimal cache placement and delivery policy. Simulation results showed that the proposed RL-based method outperforms other caching approaches and at the same time its difference with the optimum method is negligible. In future work, we shall consider more involved UE caching strategies and a more realistic model for popularity profile simulation.

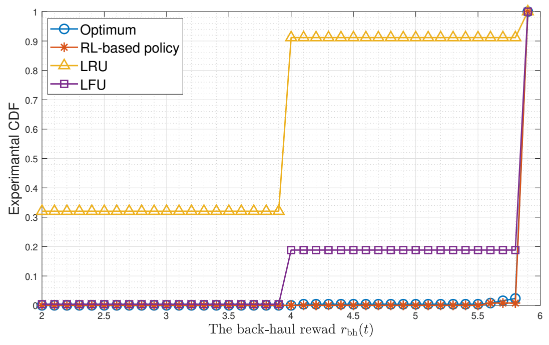

Fig. 7. Experimental CDF of the backhaul reward during network dynamics.

Acknowledgement

This work was funded in part by the Academy of Finland (grant 319058). We thank Prof. Ville Kyrki and his team for giving advises for the simulation matters.

References

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

[2] J. G. Andrews, A. K. Gupta, and H. S. Dhillon, "A primer on cellular network analysis using stochastic geometry," eprint arXiv 1604.03183, 2016.

[3] B. Serbetci and J. Goseling, "On optimal geographical caching in heterogeneous cellular networks," in *IEEE Wireless Commun. and Networking Conf. (WCNC)*, March 2017, pp. 1–6.

[4] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *IEEE Internat. Conf. on Commun. (ICC)*, June 2015, pp. 3358–3363.

[5] D. Cao, S. Zhou, and Z. Niu, "Optimal base station density for energy-efficient heterogeneous cellular networks," in *IEEE Internat. Conf. on Commun. (ICC)*, June 2012, pp. 4379–4383.

[6] M. Amidzade, H. Al-Tous, O. Tirkkonen, and G. Caire, "Cellular network caching based on multipoint multicast transmissions," in *IEEE Globecom*, Nov. 2020, pp. 1–6.

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

[8] Y. Zhou, M. Peng, S. Yan, and Y. Sun, "Deep reinforcement learning based coded caching scheme in fog radio access networks," in *IEEE/CIC Internat. Conf. on Commun. in China (Workshops)*, Aug 2018, pp. 309–313.

[9] Y. Wei, Z. Zhang, F. R. Yu, and Z. Han, "Joint user scheduling and content caching strategy for mobile edge networks using deep reinforcement learning," in *IEEE Internat. Conf. on Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[10] D. Li, Y. Han, C. Wang, G. Shi, X. Wang, X. Li, and V. C. M. Leung, "Deep reinforcement learning for cooperative edge caching in future mobile networks," in *IEEE Wireless Commun. and Networking Conf. (WCNC)*, 2019, pp. 1–6.

[11] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Networking*, vol. 24, no. 2, pp. 836–845, April 2016.

[12] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Trans. Cognitive Commun. and Networking*, pp. 1–1, 2019.

[13] N. Garg, M. Sellathurai, and T. Ratnarajah, "Content placement learning for success probability maximization in wireless edge caching networks," in *IEEE Internat. Conf. on Acoustics, Speech and Sign. Proc. (ICASSP)*, May 2019, pp. 3092–3096.

[14] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[15] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *IEEE Internat. Conf. on Computer Commun., INFOCOM*, 1999, pp. 126–134.

[16] H. Sari, G. Karam, and I. Jeanclaude, "Transmission techniques for digital terrestrial TV broadcasting," *IEEE Commun. Mag.*, vol. 33, no. 2, pp. 100–109, Feb. 1995.

[17] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," eprint arXiv 1206.5538, 2012.

[18] R. J. Williams and J. Peng, "Function optimization using connectionist reinforcement learning algorithms," *Connection Science*, vol. 3, no. 3, pp. 241–268, 1991.

[19] L. Lu, Y. Jiang, M. Bennis, Z. Ding, F. Zheng, and X. You, "Distributed edge caching via reinforcement learning in fog radio access networks," in *IEEE Vehicular Technology Conf. (VTC-Spring)*, 2019, pp. 1–6.

[20] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, and S. Niccolini, "Analyzing the performance of LRU caches under non-stationary traffic patterns," eprint arXiv 1301.4909, 2013.