Srinivasan, Ashvin; Amidzade, Mohsen; Zhang, Junshan; Tirkkonen, Olav

# Cache Policy Design via Reinforcement Learning for Cellular Networks in Non-Stationary Environment

*Published in:*
2023 IEEE International Conference on Communications Workshops

*DOI:*
10.1109/ICCWorkshops57953.2023.10283680

Published: 23/10/2023

*Document Version*
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

# Cache Policy Design via Reinforcement Learning for Cellular Networks in Non-Stationary Environment

Ashvin Srinivasan*, Mohsen Amidzadeh*, Junshan Zhang†, Olav Tirkkonen*
*Department of Information and Communications Engineering , Aalto University, Finland
†College of Engineering, University of California, Davis, California, USA
{ashvin.1.srinivasan, mohsen.amidzade, olav.tirkkonen}@aalto.fi, jazh@ucdavis.edu

***Abstract*—We consider wireless caching both at the network edge and at User Equipment (UE) to alleviate traffic congestion, aiming to find a joint cache placement and delivery policy by maximizing the Quality of Service (QoS) while minimizing backhaul load and User Equipment (UE) power consumption. We assume unknown and time-variant file popularities which are affected by the UE cache content, leading to a non-stationary Partial Observable Markov Decision Process (POMDP). We address this problem in a deep reinforcement learning framework, employing Feed Forward Neural Network (FFNN) and Long Short Term Memory (LSTM) networks in conjunction with Advantageous Actor Critic (A2C) algorithm. LSTM exploits the correlation of the file popularity distribution across time slots to learn information of the dynamics of the environment and A2C algorithm is used due to its ability of handling continuous and high dimensional spaces. We leverage LSTM and A2C tools based on its virtue to find an optimal solution for the POMDP environment. Simulation results show that using LSTM-based A2C outperforms a FFNN-based A2C in terms of sample efficiency and optimality. An LSTM-based A2C gives a superior performance under the non-stationary POMDP paradigm.**

***Index Terms*—Wireless caching, Deep Reinforcement Learning, Advantageous Actor Critic, Long Short Term Memory, Non-Stationary POMDP.**

## I. Introduction

Wireless caching for the cellular networks is regarded as a highly effective technique in alleviating traffic congestion problems [1]. The cache mechanism comprises of two phases namely cache placement and cache delivery.

Several paradigms have been explored for the cache placement and cache delivery policy [2]. A probabilistic cache placement approach is devised in [3] based on which Base-Stations (BSs) store the files. An optimal cache policy is designed based on this probabilistic approach to guarantee a maximal total hit probability for random network topologies [4]. Regarding the cache delivery, Orthogonal MultiPoint MultiCast (OMPMC) is analyzed in [5] for cache-enabled networks. OMPMC configures an content-centric network to satisfy file requests by using a location-independent content-specific multicast scheme.

Considering the intricacy and time-varying nature of cellular networks, there is a need for more intelligent caching mechanisms. Deep Reinforcement Learning (DRL) emerges as a promising solution [6]–[9]. In [6], an Actor critic DRL-based method is utilized to find an optimal policy for user scheduling and cache placement for a heterogeneous network with time-invariant file popularity. In [7], a DRL based method is exploited to achieve an optimal policy in terms of average transmission delay for a cellular network. In [8], a Long Short Term Memory (LSTM)- based DRL is leveraged for inter-slice resource management in cellular networks. In [9], a coded caching policy is designed by developing a DRL algorithm with an LSTM architecture. Then it is optimized in terms of transmission delay and a cache replacement cost. A dynamic cache policy was designed in [10], in which a non-optimal methodology was used to place the files on the User Equipment (UE) caches.

In this paper, we consider a cache policy problem characterized by a probabilistic placement approach for UEs and BSs. We develop a control variable for the network to optimize the UE power consumption during downloading files as well as to prevent the memory of UEs from exceeding their capacity. The cache policy problem is then formulated by a Partial Observable Markov Decision Process (POMDP). To find an optimal policy, we use an Advantage Actor-Critic (A2C) [10] algorithm empowered by an LSTM-based neural network [11]. In contrast to [5], [6], we develop a time-varying cache policy considering the dynamics of network operations. In contrast to [9] which utilizes a prediction mechanism to learn a cache policy for a system with unknown parameters, we formulate the cache policy design as a POMDP and evaluate an LSTM-based RL algorithm. Unlike [10], we enable the network to control the UE caches based on the control variable.

The remainder of the paper is organized as follows. In Section II, we describe the system model and file popularity model, in Section III, UE cache placement procedure is presented. In Section IV, the problem formulation is given. The deep reinforcement learning framework is introduced in Section V. Simulations results are discussed in Section VI and Section VII concludes the paper.

**Notations**: In this paper, we use bold-face lower-case to indicate vectors. Further, $\mathbf{a}^\top$ is vector transpose. We show the $m$-th element of the vector $\mathbf{b}$ by $b_m$, and $\{b_m\}_{m=1}^{p}$ collects the components of vector $\mathbf{b}$ from $m=1$ to $m=p$. Moreover, $\mathbf{1}$ and $\mathbf{0}$ denote the vector with all elements equal to one and zero, respectively.

## II. System Model

We consider a cellular access network comprising of cache-equipped BSs, UEs and a library $\mathcal{F}$ containing $N$ different files. Each file is assumed to be normalized to 1 without any loss of generality. The BSs are connected to the core network with a error-free backhaul link. The network employs BSs

to respond to an aggregated content requests received from UEs. As such, BSs fetches the needed contents and store at their caches. A probabilistic approach is used to place the contents at the BS caches [3]. For this, a common probability distribution is used. For the cache delivery, OMPMC [5] is applied. Based on this scheme, each file is simultaneously transmitted across the network by all BSs caching that file in a file-specific resource. Due to the resource orthogonality during transmitting distinct files, co-channel interference is avoided. UEs request files from the network based on a file popularity distribution $\{f_n\}_{n=1}^{N}$, where $f_n$ denotes the probability file $n$ is preferred. The transmitted files are then stored at UEs based on a probabilistic cache placement.

We model the locations of UEs and BSs according to two independent homogeneous Poisson Point Processes (PPPes).

We assume that the operation of whole network performs in a time-slotted fashion indexed by $t$. The network operation of each time-slot can be described by three phases. In the first phase, UEs from a spatial Poisson distribution request contents from the network based on the file popularity. In the second phase, based on the aggregated sum of requests for different files, the BSs fetch them from the core network and updates their respective caches. The third phase involves files being broadcasted by OMPMC [12] and updating cache of UEs. We assume that all the phases occur in a sequential manner modeled in a time-slot basis.

### A. Cache Placement and Delivery

BSs apply a probabilistic cache placement modeled based on a file-specific probability distribution, $\boldsymbol{\rho}(t) = \{\rho_n(t)\}_{n=1}^{N}$, where $\rho_n(t)$ denotes the probability of file $n$ being cached at a BS. Each BS has a maximum cache capacity $L_c$ such that $\sum_{n=1}^{N} \rho_n(t) \leq L_c$.

Once the files are cached at the BSs, files are transmitted to the UEs on file-specific resources $\{w_n(t)\}_{n=1}^{N}$ using OMPMC scheme. For this scheme, the network responds to the aggregated UE requests by transmitting cached files such that each file is broadcasted in resource $w_n(t)$ by all the BSs caching this file [5].

A UE receiving file $n$ is in outage, and correspondingly cannot decode the file, if its experienced Signal-to-Noise-Ratio (SNR) is less than a threshold value. For a network using OMPMC scheme with a propagation environment of path loss exponent $\beta = 4$, serving BSs with the average transmission power $\bar{p}$ distributed according to a PPP with intensity $\lambda_{bs}$, and UEs with the receiver noise power $\sigma_0^2$, the outage probability for file $n$ being cached at BSs with caching probability $\rho_n(t)$, and transmitted using frequency resource $w_n(t)$ is [5]

$$\mathcal{O}_n(t) = \mathrm{erfc}\left(\frac{\pi^2 \lambda_{\mathrm{bs}} \rho_{\mathrm{n}}(\mathrm{t})}{4\sqrt{2\eta_{\mathrm{n}}(\mathrm{t})}}\right), \tag{1}$$

where $\mathrm{erfc}(.)$ is the complimentary error function and $\eta_n(t) = \frac{\sigma_0^2}{\bar{p}}(2^{\alpha/w_n(t)} - 1)$ is the channel gain threshold with $\alpha$ being the spectral efficiency threshold related to a desired transmission rate.

UEs exploit a probabilistic cache placement modeled by a file-specific probability distribution $\mathbf{s}(t) = \{s_n(t)\}_{n=1}^{N}$, where $s_n(t)$ denotes the probability file n is cached at time-slot $t$. Each UE cache has a limited capacity $L_u$ such that $\sum_{n=1}^{N} s_n(t) \leq L_u$.

**Algorithm 1** Cache-Updating Procedure

**if** $c_n(t) > 0$ **then**
  **if** file $n$ is already cached at $t-1$ then do nothing, **else** attempt to decode it with probability $d_n(t)$.
**else**
  **if** file $n$ is not cached at $t-1$ do nothing **else** flush it with probability $d_n(t)$.
**end if**

### B. File Popularity

For every time-slot, users prefer files from the library $\mathcal{F}$ according to a file popularity vector $\mathbf{p}(t)$ with elements $\{p_n(t)\}_{n=1}^{N}$. The file popularity is varying from slot to slot. It is important to note that the dynamics of file popularity is not known a priori to the network. We model the file popularity as arising from three components. First, for each file there is a mean, time-varying file interest dynamic $\{f_n(t)\}_{n=1}^{N}$, collected to the vector $\mathbf{f}(t)$. For simulations, we assume that the interest of files to users develops according to a modified diffusion tipping model [13] as:

$$f_n(t) = \frac{2M_n}{1 + \cosh\left(\frac{C(t - t_{\mathrm{n},0})}{h_n}\right)}, \tag{2}$$

where $C = 4\log(1+\sqrt{2})$, $t_{\mathrm{n},0}$ is a file-specific time-shift that describes the time when interest in the file peaks, $h_n$ is a files-specific half-width of file interest peak, and $M_n = n^{-\tau} / \sum_{i=1}^{N} i^{-\tau}$ is a diffusion amplitude characterizing the peak interest in the file, which we draw from a Zipf distribution with skewness $\tau$ [14]. The file popularity dynamic at the network combines the file interests $\mathbf{f}(t)$ with the UE cache content $\mathbf{s}(t)$ and a random component as

$$\mathbf{p}(t) = \mathbf{p}(t-1) + k\Delta\mathbf{f}(t) + (1-k)\Delta\mathbf{s}(t) + \mathbf{q}(t)\,, \tag{3}$$

where $\Delta\mathbf{f}(t) = \mathbf{f}(t) - \mathbf{f}(t-1)$ is the change in interest in the file, $\Delta\mathbf{s}(t) = \mathbf{s}(t) - \mathbf{s}(t-1)$ is the change in UE cache contents, and $\mathbf{q}(t)$ is additive Gaussian noise with zero mean and diagonal covariance matrix $\boldsymbol{\Sigma}$. Here $k$ is a model parameter characterizing user patience, we assume that a fraction of users are patient and download files from the network according to their interest, while other users are impatient and use files that can be found on their UE caches. The total popularity of files in each slot is normalized to one.

## III. UE Cache Placement

The network leverages an updating mechanism where the cache contents of UEs is controlled to follow a target probability distribution, while minimizing vain decoding attempts of UEs. For file $n$, and time-slot $t$, the network generates the target UE cache probability $s_n(t+1)$ for the next time slot. It then computes the difference

$$c_n(t) = s_n(t+1) - s_n(t). \tag{4}$$

If $c_n(t) > 0$, it implies that at time-slot $t+1$ more UEs should cache file $n$ compared to the previous time-slot $t$ and if $c_n(t) \leq 0$, it indicates that some UEs should remove the file from their caches. To realize the target caching probability,

the population of users follow a procedure as follows. First, define a probability

$$d_n(t) = \begin{cases} \dfrac{s_n(t) - s_n(t-1)}{(1 - s_n(t-1))(1 - \mathcal{O}_n(t))}, & c_n(t) > 0 \\ \dfrac{s_n(t-1) - s_n(t)}{s_n(t-1)}, & c_n(t) \leq 0 \end{cases} \tag{5}$$

The quantity $d_n(t)$ is the probability a UE should decode file $n$, if it is not already cached, or that it should drop a file if it is cached. We call it the *dictating distribution*. The network broadcasts the dictating distribution to the UEs. Each UE now independently follows the update procedure of Algorithm 1. We have

**Proposition 1.** *Consider a population of UEs storing file $n$ at time-slot $t$ with cache probability $s_n(t)$, experiencing an outage during downloading file $n$ at time-slot $t+1$ with probability $\mathcal{O}_n(t)$, and following independently the cache update procedure of Algorithm 1. In the limit of infinite population size, the cache probability after the UE updates is $s_n(t+1)$. Algorithm 1 realizes $s_n(t+1)$ starting from $s_n(t)$ with the minimum number of file decoding attempts per UE.*
*Proof: Not given due to the lack of space*

## IV. Cache Policy Formulation

Here, we intend to formulate a cache policy based on a Non-stationary POMDP. A non-stationary POMDP is described by a tuple $(\mathcal{S}, \boldsymbol{O}, \mathcal{U}, P_T(\cdot;t), P_O(\cdot;t), r(\cdot))$, where $\mathcal{S}$, $\boldsymbol{O} \subset \mathcal{S}$ and $\mathcal{U}$ are the state, observation and action spaces, respectively, $P_T(\cdot, t)$ is the time-varying transition probability describing the system environment, $P_O(\cdot;t)$ is the time-varying observation distribution, and $r(\cdot)$ is the immediate reward function. The system state, observation and action, at time $t$, are denoted by $s(t) \in \mathcal{S}$, $o(t) \in O$ and $u(t) \in \mathcal{U}$, respectively. The transition probability $P_T(s(t+1)|s(t), u(t); t)$ shows the time-variant probability that being in state $s(t)$ and performing action $u(t)$ leads to the next state $s(t+1)$. For a POMDP, the state $s(t)$ is not observable for an environment-interacting agent, instead the agent has access to the observation $o(t)$ that is drawn from the distribution $P_O(\cdot|s(t+1);t)$.

For the cache policy problem of this paper, we define the observation vector based on the file popularity (3) and the UE caching probability as

$$\mathbf{o}(t) = [\mathbf{p}(t)^\top, \mathbf{s}(t)^\top]^\top. \tag{6}$$

The observation space is given as

$$\mathbf{O} = \left[ (\mathbf{p}, \mathbf{s}) \middle| \mathbf{p} \geq \mathbf{0},\ \mathbf{1}^\top \mathbf{p} = 1,\ \mathbf{s} \geq \mathbf{0},\ \mathbf{1}^\top \mathbf{s} \leq L_u \right].$$

The state vector is defined as

$$\mathbf{x}(t) = [\mathbf{p}(t)^\top, \mathbf{s}(t)^\top, \Delta\mathbf{f}(t)^\top]^\top. \tag{7}$$

The network is equipped with three control variables: the BS cache probability $\{\rho_n(t)\}_{n=1}^N$, resource allocation $\{w_n(t)\}_{n=1}^N$ being used by the network to employ the BSs to operate OMPMC scheme, and the future UE cache probability $\{s_n(t+1)\}_{n=1}^N$ through which it directs the UE to update their caches. Accordingly, we define the action vector as

$$\mathbf{u}(t) = [\boldsymbol{\rho}(t)^\top, \mathbf{w}(t)^\top, \mathbf{s}(t+1)^\top]^\top. \tag{8}$$

The action space is given by

$$\mathcal{U} = \left[ (\boldsymbol{\rho}, \mathbf{w}, \mathbf{s}) \middle| \boldsymbol{\rho} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{\rho} \leq L_c, \mathbf{w} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{w} = 1, \quad \mathbf{s} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{s} \leq L_u \right] \tag{9}$$

The control variables (8), state vector definition (6), and $\Delta\mathbf{f}(t)$ being time-varying and unknown to the network leads to a non-stationary POMDP.

In this paper, we are interested in three metrics to measure the network performance. First a Quality-of-Service (QoS) metric, where the probability of a requesting UE being satisfied by the OMPMC networking is considered. It can be expressed as

$$c_{\text{qos}}(t) = \sum_{n=1}^{N} p_n(t)(1 - s_n(t))\mathcal{O}_n(t). \tag{10}$$

From (10), if the file is not cached then the cost is directly proportional to the outage probability when minimized the QoS metric improves.

Second, we consider a backhaul load pertaining to the load for files being fetched by the BSs from the core network. If $\rho_n(t) - \rho_n(t-1) \leq 0$, then there is no backhaul load as no files are fetched, otherwise some BSs needs to cache the file $n$. Thus, the backhaul cost function is defined as [10]

$$c_{\text{bh}}(t) = \sum_{n=1}^{N} [\rho_n(t) - \rho_n(t-1)]_+, \tag{11}$$

where $[x]_+ = \frac{1}{2}(x + |x|)$.

Third, we consider a power consumption metric due to UEs updating their caches. For file $n$, each UE updates its cache with probability $d_n(t)$ (5) and only if $c_n(t) > 0$. Hence, a power consumption metric is formulated as:

$$c_{\text{uep}}(t) = \sum_{c_n(t) > 0} \min(d_n(t), 1). \tag{12}$$

We then formally define a weighted reward $r_{\text{wt}}(t)$ to be optimized as

$$r_{\text{wt}}(t) = -\big(c_{\text{qos}}(t) + \lambda_{\text{bh}} c_{\text{bh}}(t) + \lambda_{\text{uep}} c_{\text{uep}}(t)\big), \tag{13}$$

where $\lambda_{\text{bh}}$, $\lambda_{\text{uep}}$ are the Lagrange multipliers for backhaul and UE power consumption costs. Considering the dynamics introduced by difference equation (4) and stochastic difference equation (3), we formulate a discounted cumulative reward starting from time-slot $t$ till a time horizon $T$,

$$R_{\text{ac}}(t) = \sum_{k=t}^{T} \gamma^{k-t} r_{\text{wt}}(k), \tag{14}$$

where $\gamma \in [0, 1)$ is the discount factor and $T$ is the total number of time-slots for the caching policy design. We maximize $R_{\text{ac}}(t)$ considering the interplay between the state and action reflecting the UE aggregated sum requests and accordingly network cache policy. Thus, the cache policy design can be formulated as a constrained optimization problem:

$$\begin{aligned} P_1: \quad & \max_{\{\pi(\cdot|\mathbf{o}(k))\}_{k \geq t}} R_{\text{ac}}(t),\ 0 \leq t \leq T \\ & \text{s.t.} \begin{cases} \{\mathbf{w}(k), \boldsymbol{\rho}(k), \mathbf{s}(k)\}_{k \geq t} \in \mathcal{U}, \\ (1) - (3), (10) - (14). \end{cases} \end{aligned} \tag{15}$$

where $\pi\big(\cdot\,|\mathbf{o}(k)\big)$ is a observation-dependent policy distribution through which action $\mathbf{u}(t)$ is drawn. We intend to design a dynamic cache policy by solving $P_1$.

## V. Deep Reinforcement Learning Framework

We consider a RL agent to find the optimal policy $\pi^*(\cdot|\mathbf{o}(t))$ such that $P_1$ is solved. Neural network is considered for functional approximation based on the fact that they are universal approximators.

We consider A2C algorithm over other RL algorithms, like DQN, due to faster convergence and better stability, especially in non-stationary, dynamic environments. A2C also uses computational resources efficiently as an on-policy algorithm that updates policy and value functions simultaneously [10], unlike the off-policy DQN algorithm.

A2C algorithm comprises of two neural networks. The actor network, as a part of the RL agent, gives a $\theta$-parametrized policy distribution $\pi_\theta(\cdot)$ through which it can interact with the environment. Additionally, The critic network which approximates the state-value function defined as:

$$V(\mathbf{o}(t)) = \mathbb{E}\Big[\sum_{k=t}^{T} \gamma^{k-t} r(k)|\mathbf{o}(t)\Big]. \tag{16}$$

For this, the critic network is parameterized by $\phi$ and denoted by $V_\phi(\cdot)$.

### A. Actor-Critic and State processing networks

For the actor and critic networks, we apply two different architectures. In the first type, a simple Feed-Forward Neural Network (FFNN) is used in which the observation vector is fed as an input and the output is generated based on a fully connected layer. In the second type, we consider an additional extra unit namely *state-processing neural network*. For the state-processing unit, we exploit Long Short Term Memory (LSTM) cells with the input observation length varying from two till six time-slots. The output of the last LSTM cell is then given as an input to a fully connected layer. Finally, the output of state-processing network is fed into another fully-connected layer. For the sake of simplicity, we call the whole architecture of second type as LSTM. Fig.1 shows the architecture of second type neural network for actor and agent networks. Notice that, the sequential model of LSTM provides feedback connections unlike FFNNs [8], [9].

Notice that we incorporate LSTM architecture to address the non-stationary POMDP environment of study formulated in section IV.

The two architectures might use multiple observation vectors as inputs, for example, FFNN with two time-slots includes the vector $[\mathbf{o}(t-1), \mathbf{o}(t)]$ and is denoted by FF_2$s$, and LSTM with six time-slots has the vector $[\mathbf{o}(t-5), \ldots, \mathbf{o}(t)]$ and is denoted by LSTM_6$s$.

We modify the standard actor network output and consider that the action vector is drawn from three independent Dirichlet distributions such that

$$\mathbf{u}(t) \sim \Big[L_c \,\mathrm{Dirich}\big(\boldsymbol{\rho}'_{\boldsymbol{\theta}}(t)\big)^\top,\ \mathrm{Dirich}\big(\mathbf{w}'_{\boldsymbol{\theta}}(t)\big)^\top,\ L_u \,\mathrm{Dirich}\big(\mathbf{s}'_{\boldsymbol{\theta}}(t)\big)^\top\Big]^\top,$$

where $\boldsymbol{\rho}'_{\boldsymbol{\theta}}(t)$, $\mathbf{w}'_{\boldsymbol{\theta}}(t)$, and $\mathbf{s}'_{\boldsymbol{\theta}}(t)$ are the outputs of the actor network parameterized by $\boldsymbol{\theta}$, and $\mathrm{Dirich}(\cdot)$ stands for the multivariate Dirichlet distribution. Note that Dirichlet distribution is chosen in order to satisfy the constraints in (9).

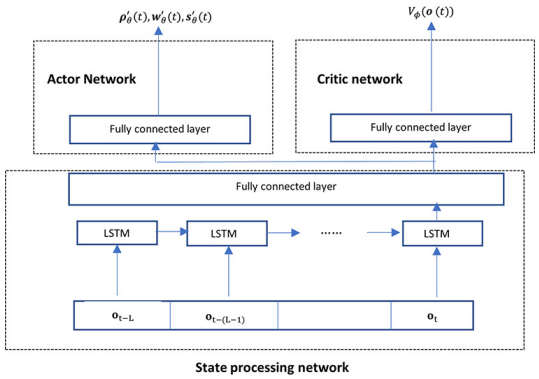


**Fig. 1:** LSTM architecture based A2C architecture

### B. A2C Algorithm

The RL actor network produces an action for a given state based on the parametric policy distribution $\pi_\theta(\cdot|\mathbf{o}(t))$. An immediate reward and the next state is obtained based on the agent interacting with the environment. The critic network simultaneously generates an estimate of the state-value function, $V_\phi(\mathbf{o}(t))$ for the current observation. Tuple of the immediate reward, action vector and estimated state-value value function are stored in a buffer and are used to update the actor and critic parameters after $T$ time-slots. This whole operation constitutes a single episodic trajectory. Several trajectories are needed for the training process. The update equations for the actor and critic networks [15] are given by

$$\begin{aligned}\phi &\leftarrow \phi + \alpha_\phi \Big[\sum_{t=1}^{T} A_\phi(\mathbf{o}(t), \mathbf{u}(t))\, \nabla_\phi V_\phi(\mathbf{o}(t))\Big] \\ \theta &\leftarrow \theta + \alpha_\theta \Big[\sum_{t=1}^{T} A_\phi(\mathbf{o}(t), \mathbf{u}(t))\, \nabla_\theta \log\big(\pi_\theta(\mathbf{u}(t)|\mathbf{o}(t))\big) \\ &\qquad + \beta \sum_{t=1}^{T} \nabla_\theta H\big(\pi_\theta(\mathbf{u}(t)|\mathbf{o}(t))\big)\Big],\end{aligned} \tag{17}$$

where $\alpha_\phi$ and $\alpha_\theta$ are the learning rates of critic and actor agents respectively, $A_\phi(\boldsymbol{o}(t), \boldsymbol{u}(t)) = r_{wt}(\boldsymbol{o}(t), \boldsymbol{u}(t)) + \gamma V_\phi(\boldsymbol{o}(t+1)) - V_\phi(\boldsymbol{o}(t))$, $H(.)$ is the entropy term used for trade off between exploration and exploitation in order to prevent A2C from converging to sub-optimal policies, and $\beta$ is the entropy regularization term.

Algorithm 2 shows the pseudo code for the modified A2C algorithm used in this paper. Note that $E_{max}$ is the total number of updates involved in the training process and *update* is the iteration number after which the model parameters are updated.

## VI. Simulation and Discussion

To evaluate the proposed DRL cache policy, we consider a cellular network with BSs and UEs located based on two independent PPPes. The deployment intensity of BSs is 300 points/km$^2$, their average transmission power is 1 and the BS cache capacity is 6. The UE noise power is set to 1 and the target spectral efficiency is set to $\alpha = 0.1$. We consider a library containing 40 files. For the diffusion model (3), we use a Zipf distribution with skewness, $\tau = 0.6$, $k = 0.9$, and $\mathrm{Diag}(\Sigma) = \Delta\mathbf{f}(\mathrm{t})$.

For the LSTM and FFNN architectures, the fully-connected layer of actor and critic networks consists of a single hidden

**Algorithm 2** Modified A2C algorithm

**Input:** A $\theta$-parametrized policy distribution $\pi_\theta(\mathbf{u}(t)|\mathbf{o}(t))$, $\phi$-parametrized approximation of state-value function $V_\phi(\mathbf{o}(t))$.
**Output:** optimal solution $\boldsymbol{\theta}$ of problem $P_1$.
**for** $update = 1$ **to** $E_{max}$ **do**
  **if** update == 1 **then**
    Initialize observation vector $\mathbf{o}(1)$ to some random point.
  **else**
    Set $\mathbf{o}(1) = \mathbf{o}_T$ based on previous update.
  **end if**
  **for** $t = 1$ **to** $T$ **do**
    Select action $\mathbf{u}(t)$ under policy $\pi_\theta(\mathbf{u}(t)|\mathbf{o}(t))$.
    Fit value function, $V_\phi(\mathbf{o}(t))$ from the critic network.
    Evaluate the advantage function $A_\phi(\mathbf{o}(t), \mathbf{u}(t)) = r(t+1) + \gamma V_\phi(\mathbf{o}(t+1)) - V_\phi(\mathbf{o}(t))$.
    Get new state vector $\mathbf{o}(t+1)$, and immediate reward $r(t+1)$.
    Buffer $V_\phi(\mathbf{o}(t))$, $\mathbf{o}(t)$, $r(t+1)$, $log(\pi_\theta(\mathbf{u}(t)|\mathbf{o}(t)))$, $H(\pi_\theta(\mathbf{u}(t)|\mathbf{o}(t)))$.
    Update parameters for actor and critic networks based on the updating rules (17).
  **end for**
  Set $\mathbf{o}_T = \mathbf{o}(T)$.
**end for**

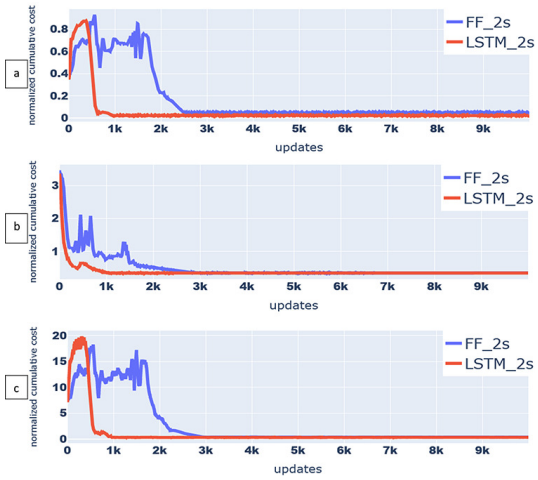


**Fig. 2:** Train performance between LSTM and FFNN with two states for $\gamma = 0.98$. (**a**) denotes the normalized cumulative QoS cost, (**b**) is the normalized cumulative backhaul cost, (**c**) is the normalized cumulative UE power consumption cost.

layer with 256 neurons, each being activated by a $\mathrm{Tanh}(\cdot)$ function. The output activation functions for actor and critic networks, are $\mathrm{Relu}(\cdot)$ and $\mathrm{Softplus}(\cdot)$, respectively. For the state-processing unit of LSTM architecture, we use a fully-connected layer with a single hidden layer of 128 neurons, each being activated by a $\mathrm{Tanh}(\cdot)$ function. The output of LSTM network is connected to a Fully Connected layer with 128 neurons which is subsequently given as an input to the actor-critic networks. The learning rates $\alpha_\theta$ and $\alpha_\phi$, both are set to $10^{-3}$ and optimizer is *Adam* with default settings.

Regarding the hyper-parameters of Algorithm 2, we set total number of updates $E_{max} = 10^4$, the Lagrange multipliers of the weighted reward (13) $[\lambda_{\mathrm{bh}}, \lambda_{\mathrm{uep}}] = [0.05, 0.05]$, and the entropy regulation term $\beta = 0.005$.

Fig.2 shows the training performance of FFNN and LSTM architectures with two time-slots of observations, i.e., FF_2s and LSTM_2s, in terms of normalized discounted cumulative costs of QoS $c_{\mathrm{qos}} = \sum_{t=1}^{T} \gamma^{t-1} c_{\mathrm{qos}}(t)/T$, backhaul $c_{\mathrm{bh}} = \sum_{t=1}^{T} \gamma^{t-1} c_{\mathrm{bh}}(t)/T$ and UE power-consumption $c_{\mathrm{uep}} = \sum_{t=1}^{T} \gamma^{t-1} c_{\mathrm{uep}}(t)/T$, as a function of update samples. It can be seen that for all the costs, LSTM outperforms FFNN from sample-efficiency perspective. It shows that the LSTM architecture can more efficiently provide a cache policy for the considered POMDP than the conventional feed forward neural networks. Furthermore, it can be seen that LSTM takes approximately one-third the number of samples to converge compared to FFNN architecture which strongly indicates the sample-efficiency of the LSTM architecture. This suggests that feedback connections within the LSTM network is instrumental in learning the file popularity evolution that is latent in the observation vector.

In order to benchmark the RL-based cache policy, we consider a static cache solution obtained by an *interior-point algorithm*. For this static solution, we consider $T$ time-slots of the network operations but each time-slot is independently optimized based on its immediate reward, instead of a cumulative reward. Therefore, in contrast to the dynamic solution of RL algorithm, a static solution is obtained. As such, we call it "Static". Fig.3 illustrates the test performance of static solution and RL-based solutions with different NN architectures. For the RL-based solutions, the agent is trained for different values of discount factor $\gamma \in [0.9, 1)$, the model hyper-parameters are saved, and then the agent is evaluated during a test scenario with discount factor $\gamma = 1$. The cumulative costs considered during the test time evaluation are $c_{\mathrm{qos}}, c_{\mathrm{bh}}, c_{\mathrm{uep}}$ with $c_{\mathrm{qos}} = \sum_{t=1}^{T} c_{\mathrm{qos}}(t)/T$, $c_{\mathrm{bh}} = \sum_{t=1}^{T} c_{\mathrm{bh}}(t)/T$, $c_{\mathrm{uep}} = \sum_{t=1}^{T} c_{\mathrm{uep}}(t)/T$. From Fig.3 it can be seen that the LSTM outperforms FFNN from the optimality perspective. Furthermore, increasing the observation length of LSTM cells decreases all cumulative costs of QoS, backhaul and UE power consumption metrics. It suggests that including more history of the observation improves the RL agent performance due to increase in the ability of discovering the latent information. The static solution performs better than the RL dynamic solution of FFNN with one state. However, it performs poorly compared to the RL dynamic solution when multiple time-slots are exploited irrespective of FFNN or LSTM. This justifies the usage of cumulative reward optimization as mentioned in $P_1$ to design a dynamic cache policy as well as that the developed A2C algorithm can find an optimal solution for the formulated POMDP problem.

To specify how the latent information in the observation vector might impact on the policy performance of different Neural Network (NN) architectures, we consider a tentative case with MDP environment. For the MDP case, the state vector $\mathbf{x}(t)$ of (7) is used instead of the observation vector $\mathbf{o}(t)$. Fig.4 compares the test performance of different approaches for the POMDP and MDP environments in terms of cumulative costs with $\gamma = 1$ as a function of different values of training discounted factor, $\gamma \in [0.9, 1)$. Solutions of MDP are shown by dotted lines for $\gamma = 0.98$. It can be seen that the difference between cumulative cost of MDP and POMDP for LSTM is remarkably smaller than the difference of FFNN architecture. It indicates that LSTM exhibits better performance for a POMDP environment.

## VII. Conclusion

In this paper, we formulated a cache placement and delivery problem based on a reinforcement learning framework. We aimed at jointly optimizing the quality of service, backhaul and

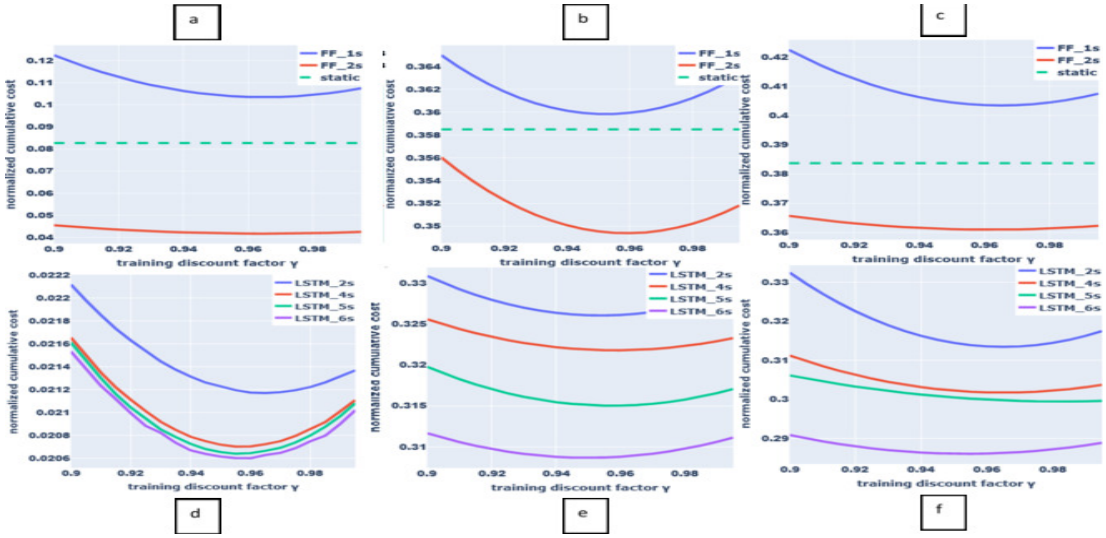


**Fig. 3:** Test performance for static and RL-based solutions with different NN architectures as a function of discount factor. (**a**), and (**d**) denotes QoS cost. (**b**), and (**e**) denotes backhaul cost. (**c**), and (**f**) denotes UE power consumption cost.

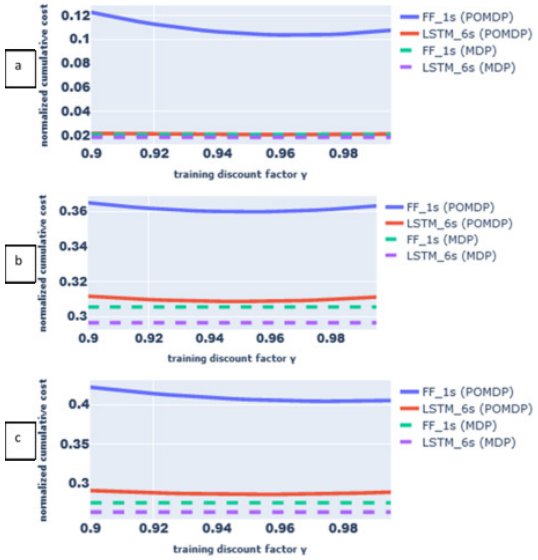


**Fig. 4:** Test performance for POMDP and MDP environments.(**a**): normalized cumulative QoS cost, (**b**): normalized cumulative backhaul cost, (**c**): normalized cumulative UE power consumption cost.

UE power consumption using Advantageous Actor Critic (A2C) algorithm. For the RL agent network, we explored two different types of architectures namely Feed Forward Neural Network (FFNN) and Long Short Term Memory (LSTM). Simulation results justified usage of POMDP for problem formulation and showed that the proposed LSTM-based A2C outperforms FFNN-based A2C in terms of sample-efficiency and optimality. Furthermore, it was also shown that LSTM-based A2C can outperform considerably better for a POMDP environment. For future work, we shall construct a more comprehensive model for file popularity dynamics, and investigate how difference rates of change affect learning efficiency.

## ACKNOWLEDGEMENT

This work was funded in part by the Academy of Finland (grant 345109).

## REFERENCES

[1] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-memory trade-off for caching and delivery of correlated sources," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2219–2251, 2020.

[2] H. Wu, Y. Fan, Y. Wang, H. Ma, and L. Xing, "A comprehensive review on edge caching from the perspective of total process: Placement, policy and delivery," *Sensors*, vol. 21, 2021.

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

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

[5] M. Amidzadeh, H. Al-Tous, G. Caire, and O. Tirkkonen, "Caching in cellular networks based on multipoint multicast transmissions," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2022.

[6] 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 *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[7] 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 *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, 2019, pp. 1–6.

[8] R. Li, C. Wang, Z. Zhao, R. Guo, and H. Zhang, "The LSTM-based advantage actor-critic learning for resource management in network slicing with user mobility," *IEEE Communications Letters*, vol. 24, no. 9, pp. 2005–2009, 2020.

[9] Z. Zhang and M. Tao, "Deep learning for wireless coded caching with unknown and time-variant content popularity," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1152–1163, 2021.

[10] M. Amidzadeh, H. Al-Tous, O. Tirkkonen, and J. Zhang, "Joint cache placement and delivery design using reinforcement learning for cellular networks," in *Vehicular Technology Conference*, 2021, pp. 1–6.

[11] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free RL can be a strong baseline for many POMDPs," in *International Conference on Machine Learning*, 2022, pp. 16 691–16 723.

[12] J. Andrews, A. Gupta, and H. Dhillon, "A primer on cellular network analysis using stochastic geometry," 2016.

[13] M. Chiang, "Networked life," Cambridge University Press, 2012.

[14] 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.

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