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

Adaptive Cache Policy Optimization Through Deep Reinforcement Learning in Dynamic Cellular Networks

Published in:
Intelligent and Converged Networks

DOI:
10.23919/ICN.2024.0007

Published: 01/01/2024

Document Version
Publisher's PDF, also known as Version of record

Intelligent and Converged Networks ISSN 2708-6240
2024, 5(2): 81−99 DOI: 10.23919/ICN.2024.0007

Adaptive cache policy optimization through deep reinforcement learning in dynamic cellular networks

Ashvin Srinivasan*, Mohsen Amidzadeh, Junshan Zhang, and Olav Tirkkonen

Abstract: We explore the use of caching both at the network edge and within User Equipment (UE) to alleviate traffic load of wireless networks. We develop a joint cache placement and delivery policy that maximizes the Quality of Service (QoS) while simultaneously minimizing backhaul load and UE power consumption, in the presence of an unknown time-variant file popularity. With file requests in a time slot being affected by download success in the previous slot, the caching system becomes a non-stationary Partial Observable Markov Decision Process (POMDP). We solve the problem in a deep reinforcement learning framework based on the Advantageous Actor-Critic (A2C) algorithm, comparing Feed Forward Neural Networks (FFNN) with a Long Short-Term Memory (LSTM) approach specifically designed to exploit the correlation of file popularity distribution across time slots. Simulation results show that using LSTM-based A2C outperforms FFNN-based A2C in terms of sample efficiency and optimality, demonstrating superior performance for the non-stationary POMDP problem. For caching at the UEs, we provide a distributed algorithm that reaches the objectives dictated by the agent controlling the network, with minimum energy consumption at the UEs, and minimum communication overhead.

Key words: wireless caching; deep reinforcement learning; advantageous actor critic; long short term memory; non-stationary Partial Observable Markov Decision Process (POMDP)

1 Introduction

Wireless caching in cellular networks is a highly effective method for alleviating traffic congestion problems[1]. A variety of methods have been explored to develop efficient policies for the two phases of this problem, cache placement and cache delivery[2].

In Ref. [3], probabilistic cache placement is considered, focusing on determining how Base Stations (BSs) should store files. This probabilistic approach serves as the foundation for designing an optimal cache policy, which in turn, ensures the highest total hit probability for random network topologies[4].

During the cache delivery phase, it is crucial to differentiate between unicast and multicast methods, as well as between single-point approaches, where a file is delivered by the caching BS with the highest signal power, and multipoint approaches, where a user downloading a file receives simultaneous transmissions from multiple BSs. Single-Point Unicast (SPUC) cache delivery has been analyzed in Refs. [3–11]. In Ref. [5], a dynamic network architecture is considered, where the nearest BS responds to a User Equipment (UE) that demands service. In Ref. [6], BSs are equipped with multiple antennas. Each BS utilizes beamforming transmissions eliminate interference within a chosen group of cooperating BSs. An SPUC scheme is considered in Ref. [8] in a Heterogeneous Network (HetNet) setup, incorporating zero-forcing beamforming BSs and cache-enabled helper-nodes.

Cache delivery based on Single-Point Multicast

• Ashvin Srinivasan, Mohsen Amidzadeh, and Olav Tirkkonen are with the Department of Information and Communications Engineering, Aalto University, Espoo 02150, Finland. E-mail: {ashvin.1.srinivasan, mohsen.amidzade, olav.tirkkonen}@aalto.fi.
• Junshan Zhang is with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, USA. E-mail: jazh@ucdavis.edu.
* To whom correspondence should be addressed.
Manuscript received: 2023-06-01; revised: 2023-07-25; accepted: 2023-11-07

(SPMC) transmission is considered in Refs. [12–16]. The approach involves caching BSs that multicast or broadcast different files to requesting UEs using multiple access techniques. For instance, in Refs. [12, 13], probabilistic caching is considered in a HetNet environment, where each BS multicasts k files using pre-assigned resources that each span $1/k$ of the total available bandwidth. In Ref. [14], optimal random caching designs for perfect, imperfect, and unknown file popularity distributions in a large-scale multi-tier wireless network are considered. In Ref. [16], Peng et al. investigated multicast sparse beamforming based on a deterministic cache placement.

References [17, 18] have examined deterministic caching with on-demand Multipoint Unicast (MPUC) cache delivery. In Ref. [17], caching BSs deliver a cached file to the requesting UE using a distinct resource, with network resources orthogonalized among UEs to avert interference. This scheme does not incorporate collaborative beamforming based on Channel State Information (CSI), and the BSs only use a single antenna. In contrast, Ref. [18] applies Coordinated Multipoint (CoMP) transmission with zero-forcing beamforming. It assumes known CSI between UEs and a group of serving BSs. Here, each UE receives unicast CoMP transmissions from a set of serving BSs, which inevitably leads to multiple access interference.

Here, we use Orthogonal Multipoint Multicast (OMPMC)[19] for cache delivery. OMPMC caters to file requests through a location-independent, content-specific multicast scheme. This approach significantly reduces the complexity of content delivery while ensuring efficient utilization of network resources.

The dynamic nature of network traffic, and user mobility, necessitate the development of more intelligent caching mechanisms. Deep Reinforcement Learning (DRL) has emerged as a promising solution, offering a robust framework for addressing these challenges[20–23]. In Ref. [20], an actor-critic learning method is employed to identify an optimal policy for user scheduling and cache placement in a heterogeneous network with constant file popularity. This approach aims to optimize network performance and user experience simultaneously. In Ref. [21], DRL is utilized to achieve an optimal policy in terms of average transmission delay for a cellular network. This optimization process is crucial for ensuring timely delivery of information and enhancing overall network efficiency. In Ref. [22], Long Short-Term Memory (LSTM)-based DRL is applied for inter-slice resource management in cellular networks. This advanced technique enables dynamic allocation and reallocation of resources, leading to more efficient network performance. A coded caching policy is developed using a DRL algorithm with an LSTM architecture in Ref. [23]. This policy is subsequently optimized in terms of transmission delay and cache replacement cost, resulting in significant improvements in network efficiency and cost-effectiveness. A dynamic cache policy is designed in Ref. [24], incorporating a non-optimal methodology for placing files on UE caches. This approach seeks to strike a balance between performance and complexity.

In this paper, we study a cache policy optimization problem where UEs cache content proactively, in addition to edge caching at BSs; and cache placement at both BSs and UEs is probabilistic. Departing from Ref. [24], we allow the network to control the UE caches, which is essential for maintaining the balance between resource allocation, user experience, and UE power consumption during proactive file downloads. To take into account the time-varying nature of dynamic caching, the file preference distribution is modeled as a non-stationary stochastic process; this is in contrast to Refs. [19, 20, 24], where the dynamics of the network are treated as stationary, and the state space is considered finite. We also assume that the underlying file popularity distribution is unknown to the network, making the problem only partially observable.

We thus formulate the cache policy optimization problem as a Non-stationary Partially Observable Markov Decision Process (N-POMDP), a sophisticated model that considers the uncertainties inherent in cellular networks. To find an optimal policy, we employ an Advantage Actor-Critic (A2C)[24] algorithm, which is supported by an LSTM-based neural

network[25] to handle the non-stationarity. This approach differs from Ref. [23], where a prediction mechanism is applied to learn a cache policy for a system with unknown parameters. An LSTM-based Reinforcement Learning (RL) algorithm to solve the formulated POMDP offers a more rigorous and systematic approach to cache policy optimization.

The remainder of the paper is organized as follows: Section 2 outlines the system and file popularity models. The problem is formulated in Section 3. Section 4 details the UE cache placement procedure, while the deep reinforcement learning framework is introduced in Section 5. Numeric simulation results are provided in Section 6, and Section 7 concludes the paper.

Notation: We use bold-face lower-case letters to indicate vectors, and $\boldsymbol{a}^{\mathrm{T}}$ is the transpose of a vector. We indicate the m-th element of vector $\boldsymbol{b}$ by b_m, and $\{b_m\}_{m=1}^{p}$ collects the components of vector $\boldsymbol{b}$ from $m=1$ to $m=p$. Moreover, $\mathbf{1}$ and $\mathbf{0}$ denote the vector with all elements equal to one and zero, respectively.

2 System model

We examine a cellular access network that consists of cache-equipped BSs, UEs, and a library $\mathcal{F}$ containing N different files. Without loss of generality, each file is assumed to be normalized to a value of 1. The BSs are connected to the core network through error-free backhaul links. In this network, the BSs are responsible for responding to aggregated content requests from UEs. To fulfill these requests, the BSs fetch the required contents and store them in their caches. To determine the placement of contents in the BS caches, a probabilistic approach is employed, as described in Ref. [3]. This approach utilizes a common probability distribution.

The cache delivery process utilizes OMPMC, as in Ref. [19]. According to this scheme, each file is transmitted simultaneously across the network by all BSs that cache that particular file, using a dedicated resource specific to that file. This resource orthogonality ensures that co-channel interference is avoided, as distinct files are transmitted using different resources. UEs request files from the network based on a file popularity distribution $\{p_n\}_{n=1}^{N}$, where p_n represents the probability of file n being preferred. The transmitted files are then stored at UEs' caches based on a probabilistic cache placement strategy.

The network operates in a time-slotted manner, where each time slot is indexed by t. The network operation within each time slot can be divided into three phases. In the first phase, UEs, distributed according to a spatial poisson distribution, request content from the network based on file popularity. In the second phase, the BSs retrieve the requested files from the core network based on the aggregate sum of requests for different files, and update their respective caches. The third phase involves the broadcast of files using OMPMC[26], and the update of UEs' caches. We assume that all these phases occur sequentially on a time-slot basis. The interactions among UEs, BSs, and the core network for a given time slot t are illustrated in Fig. 1.

2.1 Cache placement and delivery

The network applies probabilistic cache placement strategy characterized by a file-specific probability distribution[4], denoted as $\boldsymbol{\rho}(t)=\{\rho_n(t)\}_{n=1}^{N}$, where $\rho_n(t)$ represents the probability of file n being cached at a randomly selected BS. Each BS has a maximum cache capacity of L_b, thus we have $\sum_{n=1}^{N}\rho_n(t)\leqslant L_b$. Without loss of generality, we assume that all files have the same size. If files are of different sizes, they can be segmented into equal-sized chunks, and each segment could be treated as an individual entity in the caching policy. Instead of formulating the popularity distribution of entire files, we would then consider the popularity of the individual segments.

For caching at the BSs, we follow the principle of Ref. [4]. L_b segments of length 1 are used to represent the L_b units of cache memory at the BSs, as shown in Fig. 2. The segments are filled according to the weights $\{\rho_n(t)\}_{n=1}^{N}$. When a segment becomes full and cannot completely accommodate a given weight, the remaining probability weight is filled into the next segment. As a result, each of these L_b segments accommodates a variety of potential files that could be

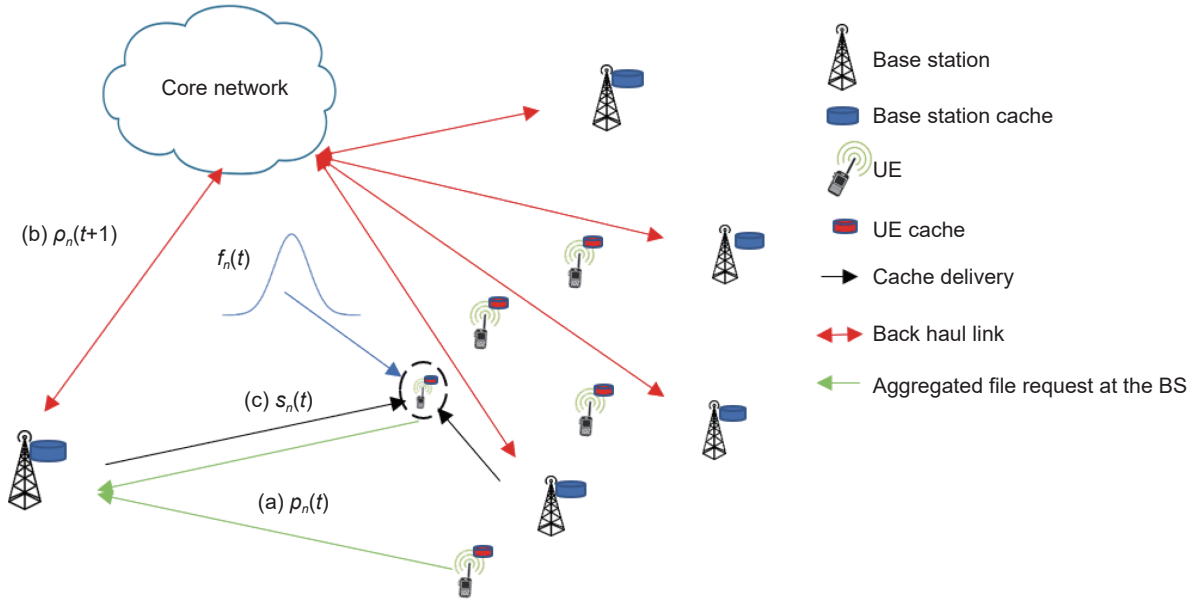

Fig. 1 Communication and coordination among UE, BS, and the core network for cache placement and delivery. (a) File request mechanism at the BS. (b) BS caching probability mechanism. (c) UE caching mechanism provided by the network.

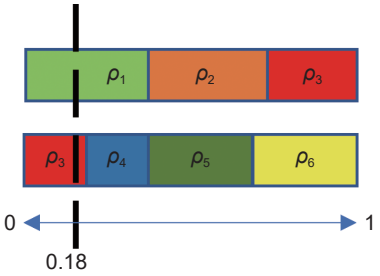

Fig. 2 Example showcasing probabilistic caching for L_b = 2, and N = 6 files with caching probabilities ρ = [0.4, 0.3, 0.5, 0.2, 0.3, 0.3]. A random number U = 0.18 is drawn from the uniform distribution over [0, 1], represented by the vertical line. The line intersects at ρ_1 and ρ_3, indicating that Files 1 and 3 will be cached.

stored in the corresponding portion of the memory. Once all segments have been populated, the BS generates a uniformly distributed random variable U in the range $[0,1]$. The BS then stores the files that coincide with this the position of this random variable in each segment. The probability of file n to be cached becomes precisely ρ_n.

For proactive caching at UEs, we also employ a probabilistic cache placement strategy, based on multicast cache delivery from BSs to UEs. We adopt a one-shot probabilistic UE-caching principle, where the network broadcasts dictating messages to the UEs, and UEs fill their caches accordingly from cache delivery transmissions of the network. The process of filling UE caches, differs from filling BS caches in two essential ways. First, the success of cache delivery from BS caches to UEs depends on randomness of wireless channels. As a consequence, if there is a strict UE cache capability L_u, such that in each UE cache at any time, there is at most L_u files, a UE-cache placement policy which always would fill all UE caches would have to be based on feedback, to mitigate packet losses in wireless transmission. To avoid this, we assume an average cache capacity constraint at UEs; on average, the number of files cached at a UE is not larger than L_u. This can be realized as a service-level agreement, where a fraction of the UEs memory, on average, is allocated to the caching service. Second, to minimize UE power consumption, the amount of files that a UE attempts to decode should be kept at a minimum.

To achieve this, we apply a proactive UE caching method where file decoding attempts at a UE depend on the dictating messages and the cache contents. On the population level, the UE caching probability vector $\boldsymbol{s}(t)$ describes the state of the UE caches. Element $s_n(t)$ represents the probability of file n being cached at a UE at time-slot t. Each UE cache has a restricted capacity of L_u. We thus have

$$\sum_{n=1}^{N} s_n(t) \leqslant L_u \tag{1}$$

At their most general, the dictating messages from the network to the population of UEs consist of two $2^N \times 2^N$ matrices, where for each possible cache content of a UE, there is a probability to attempt to decode certain files, or a probability to flush certain files from memory.

Cache delivery from BSs towards the UEs uses file-specific resources $\{w_n(t)\}_{n=1}^{N}$ through the OMPMC scheme. In this scheme, the network responds to the aggregated UE requests by broadcasting the cached files. Each file is simultaneously broadcasted in resource $w_n(t)$ by all the BSs that cache that particular file[19].

We assume that the users experience block Rayleigh fading in addition to large-scale distance dependent path loss. The instantaneous Signal to Interference plus Noise Ratio (SINR) for UE m receiving file n is $\gamma_{m,n}$. We assume that transmission of the files happen at a rate R_{th}, and that Additive White Gaussian Noise (AWGN) capacity achieving codebooks at this rate are used. Accordingly, R_{th} becomes a threshold rate; if the instantaneous AWGN-capacity of a user receiving a file is larger than R_{th}, the user succeeds in decoding, otherwise the user is in outage. Assuming that all files have the same size, file n is thus in outage at UE m if $w_n W \log_2(1+\gamma_{m,n}) < R_{\text{th}}$, where W is the total bandwidth of the transmission. We can thus define a spectral efficiency threshold $\alpha = R_{\text{th}}/W$, such that at time t, the outage probability for UE m receiving file n in the dedicated fractional resource $w_n(t)$ becomes[19]

$$o_n(t) = P(w_n(t)\log_2(1+\gamma_{m,n}(t)) < \alpha).$$

We utilize two independent homogeneous Poisson Point Processes (PPPs) to model the locations of UEs and BSs in the network. For a network using OMPMC scheme with a propagation environment of path loss exponent $\beta_{\text{pl}} = 4$, with BSs having the average transmission power $\overline{p}$ distributed according to a PPP with intensity λ_{bs}, and UEs having the receiver noise power σ_0^2, the outage probability for file n being cached at BSs with caching probability $\rho_n(t)$, transmitted using frequency resource $w_n(t)$ then becomes[24]

$$o_n(t) = \text{erfc}\left(\frac{\pi^2 \lambda_{\text{bs}} \rho_n(t)}{4\sqrt{\eta_n(t)}}\right) \tag{2}$$

where erfc(·) is the complementary error function and $\eta_n(t) = \frac{\sigma_0^2}{\overline{p}}(2^{\alpha/w_n(t)} - 1)$ is a channel gain threshold.

2.2 File popularity

In each time slot, users request files from the library $\mathcal{F}$. A user either finds the requested file in its cache, or it attempts to decode it from an OMPMC transmission of the network, which may or may not be in outage. We assume that irrespective of whether or not a requested file is in the cache and/or in outage, users inform the network about the request for the time slot. These requests are then aggregated by the network to a file request probability $p_n(t)$ for each file n, forming the file request probability vector $\boldsymbol{p}(t)$. We assume that this is a probability, such that $\sum_{n=1}^{N} p_n(t) = 1$.

There is an underlying dynamic file popularity distribution driving the user's requests, with $f_n(t)$ being the popularity of file n. We model two types of users, patient and impatient ones. If a patient user requests file m in slot $t-1$, if the file is not in the user's cache and the user faces outage, this user will request the same file again in slot t. An impatient user facing this situation, in contrast, selects a file at random in slot t according to $f_n(t)$. We assume that a user is patient with probability ζ. Assume an event O_m, where a user requests file m in slot $t-1$, it is not in the UE cache, and reception from the network fails. The probability of this event is

$$P(O_m) = (1 - s_m(t-1))\, o_m(t-1)\, p_m(t-1) \tag{3}$$

The probability that file n is requested in slot t is then

$$P(n|O_m) = \zeta\, \delta_{nm} + (1-\zeta)\, f_n(t) \tag{4}$$

where δ_{nm} is the Kronecker delta function.

Furthermore, we assume that in the event $\overline{O}_m$ where the user requests file m in slot $t-1$ and either finds it in its cache or successfully receives it, the user will not request the same file in slot t. Otherwise, the user chooses the file freely according to $f_n(t)$:

$$P(n|\overline{O}_m) = \frac{f_n(t)}{1-f_m(t)}(1-\delta_{nm}) \tag{5}$$

The probability of this event is

$$P(\overline{O}_m) = p_m(t-1) - P(O_m) \tag{6}$$

Finally, we assume that there is Gaussian noise $\nu_n(t)$ affecting the file request process, reflecting the randomness of user activation, and differences in real preferences of users. The file request probability dynamics before normalization thus becomes

$$\begin{aligned} p_n(t) =& \nu_n(t) + \sum_{m=1}^{N} P(n|O_m)P(O_m) + P(n|\overline{O}_m)P(\overline{O}_m) = \\ & f_n(t)\sum_{m=1}^{N}(1-\xi_m(t-1))\,p_m(t-1) + \\ & \xi_n(t-1)\,p_n(t-1) \;+\; \nu_n(t) \end{aligned} \tag{7}$$

where

$$\begin{aligned} \xi_n(t) =& \zeta(1-s_n(t))o_n(t) - \\ & \frac{f_n(t+1)}{1-f_n(t+1)}(1-o_n(t)(1-s_n(t))), \end{aligned}$$

arises from the non-stationary driving popularity distribution. The final normalization step adds a mild non-linearity to the problem.

For concreteness, we model the underlying file popularity distribution in terms of a modification of the diffusion model applied in Ref. [27]:

$$f_n(t) = \frac{2m_n}{1+\cosh\left(\dfrac{e(t-t_{n,0})}{h_n}\right)} \tag{8}$$

where $t_{n,0}$ is a file-specific time-shift that describes the time instance when the interest in the file peaks, h_n is a file-specific half-width of the file interest peak in units of time slots, 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 τ[28], and $e = 4\ln(1+\sqrt{2})$ is a constant.

It is important to note that the model developed in Eq. (7) is not confined to a specific form of $f_n(t)$. Rather, it exhibits a degree of flexibility, allowing it to be applied across various contexts and systems. We assume that the stochastic dynamics in Eq. (7) is not a priori known to the network. The network does not know the underlying popularity distribution $f_n(t)$, while it does know the realized file request probabilities $p_n(t)$.

3 UE cache placement

The network employs an updating mechanism in order to manage the cache contents of UEs to adhere to a target probability distribution $\boldsymbol{s}$. For this, the network broadcasts dictating messages. The UEs act as independent agents. Depending on the cache content of an individual UE, and the dictating messages, the UEs attempt to decode files and/or flush files. In this section, we shall find an optimal distributed procedure that the UE agents follow, such that the aggregate action of the UE agents leads to the state desired by the network with minimum energy consumption. We assume that UE energy consumption for the caching policy directly depends on the number of file decoding attempts. As discussed in Section 2.1, we have Formula (1) on the average cache size of the UEs.

In time-slot t, the network generates the target UE cache probabilities $\{s_n(t+1)\}$ for the files for the next time slot. It then broadcasts dictating messages to the UEs, with the objective of changing $s_n(t)$ to the target values.

First, we observe that while the continuous variables $\boldsymbol{s}$ describe the probabilities to find files in the user caches, when considering the cache of a given user, the probabilities of finding different files are not independent. This is directly seen in an extreme case, where $\boldsymbol{s}$ is such that the same set of L_s files is cached in every UE. The probability space describing possible UE cache contents thus is 2^N-dimensional. A priori, any one of the N files may or may not be cached at a given UE. Similarly, there are 2^N different combinations of files that a UE may attempt to decode in a time slot, and in principle 2^N combinations of files that a UE may flush. Despite this rather complicated setting, we find that for an energy consumption minimizing UE caching policy with the average cache size constraint, correlations between probabilities of files being cached do not need to be taken into account. It is sufficient for each UE to treat each file independently.

If $s_n(t+1) > s_n(t)$, it implies that at time-slot $t+1$, more UEs should cache file n as compared to time-slot t while if $s_n(t+1) \leqslant s_n(t)$, some UEs should discard the file from their caches. To realize the target caching probability, the population of users follow a procedure as follows. Based on the caching probablity changes

$$\Delta_n(t+1) = s_n(t+1) - s_n(t),$$

we define the dictating variables:

$$d_n(t) = \begin{cases} \dfrac{\Delta_n(t+1)}{(1-s_n(t))(1-o_n(t))}, & \Delta_n(t+1) > 0; \\ \dfrac{\Delta_n(t+1)}{s_n(t)}, & \Delta_n(t+1) \leqslant 0 \end{cases} \tag{9}$$

This number is positive if more caching is needed, and negative if less is needed. The quantity $|d_n(t)|$ is the probability a UE should decode file n, if it is not already cached, or that it should discard it if it is cached. The network broadcasts $d_n(t)$ to the UEs. Each UE now independently follows the update procedure of Algorithm 1. We have

Proposition 1 Consider a population of UEs storing files at time-slot t with aggregate cache probability $\{s_n(t)\}$, experiencing an outage during downloading file n with probability $o_n(t)$, and following independently the cache update procedure of Algorithm 1. In the limit of an infinite population, Algorithm 1 leads to cache probability $\{s_n(t+1)\}$ with a minimum number of file decoding attempts per UE.

Proof As a shorthand, we denote variables in slot $t+1$ with a "+", and variables in slot t without. The target cache probability vector $\boldsymbol{s}^+$, and $\boldsymbol{s}$ then represents a global view of the network on UE cache contents. The cache content of an individual UE at time t is given by the set $\boldsymbol{c}$, a set of decoding attempts performed by an individual UE is denoted by $\boldsymbol{a}$, and a set of files flushed by a UE is denoted by $\boldsymbol{b}$. These can be interchangeably thought of as N-dimensional vectors, or subsets of F. These represent the local view of a UE of its cache content, and the actions it may take. The probability of a UE having a given cache content can be summarized in a 2^N-dimensional vector $\boldsymbol{p_c}$, while the probabilities of decoding attempt and file flushing events are $\boldsymbol{p_a}$ and $\boldsymbol{p_b}$, respectively.

Algorithm 1 Cache-updating procedure

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

The conditional probability for a combination of file decoding attempts given a UE cache content is $\boldsymbol{P_{a|c}}$, and the conditional probability for a combination of files being flushed given a UE cache content is $\boldsymbol{P_{b|c}}$. Both can be understood as $2^N \times 2^N$ matrices. A generic probabilistic UE cache update policy can be described in terms of these two conditional probability matrices.

First, it is worth to observe that a file cannot be flushed if it is not cached, thus the matrix elements of the conditional flushing probability fulfill $p_{\boldsymbol{b}|\boldsymbol{c}} = 0$ if $\boldsymbol{b} \not\subset \boldsymbol{c}$. Also, if a file is already cached, downloading it does not change the cache content and only consumes energy, thus we require that $p_{\boldsymbol{a}|\boldsymbol{c}} = 0$ if $\boldsymbol{a} \cap \boldsymbol{c} \neq \varnothing$.

To move between the 2^N-dimensional probability space of UE cache contents and the N-dimensional space of $\boldsymbol{s}$, we use an $N \times 2^N$ constant indicator matrix $\boldsymbol{J}$. This matrix has entries in $\{0,1\}$. Each column is a possible cache content, and the ones indicate the files in the cache.

With these notations, the probability that a file in the library is cached at an arbitrary UE is given by the vector

$$\boldsymbol{s} = \boldsymbol{J}\,\boldsymbol{p_c} \tag{10}$$

while the expected number of decoding attempts u_n of file n by a UE and the expected number of times v_n that file n is flushed by a UE, collected to N-dimensional vectors, are

$$\boldsymbol{u} = \boldsymbol{J}\,\boldsymbol{P_{a|c}}\,\boldsymbol{p_c}\,;\ \ \boldsymbol{v} = \boldsymbol{J}\,\boldsymbol{P_{b|c}}\,\boldsymbol{p_c} \tag{11}$$

Note that here we use matrix notation, such that summation over $\boldsymbol{c}, \boldsymbol{a}$, and $\boldsymbol{b}$ is implicitly understood.

As the decoding attempts face channel uncertainty in terms of packet loss, the UE-cache update equation becomes

$$\boldsymbol{s}^+ - \boldsymbol{s} = (\boldsymbol{D}_o)\boldsymbol{u} - \boldsymbol{v} \tag{12}$$

where the diagonal matrix $\boldsymbol{D}_o$ has the packet decoding success probabilities $1 - o_n$ on the diagonal.

The objective is to minimize UE power consumption spent on decoding attempts, i.e., the expected number of file decoding attempts given by the one-norm $\|\boldsymbol{u}\|_1$. As both $\boldsymbol{u}$ and $\boldsymbol{v}$ have non-negative entries, we find the rather obvious fact that, assuming that none of the files has outage probability 1, the optimum expectations for decoding attempts and flushing fulfill

$$\boldsymbol{u}^* = \boldsymbol{D}_o^{-1}\left[(\boldsymbol{s}^+ - \boldsymbol{s})\right]_+ ;\ \boldsymbol{v}^* = \left[(\boldsymbol{s} - \boldsymbol{s}^+)\right]_+ \tag{13}$$

where the non-negative part of a number is given by $[x]_+ = \frac{1}{2}(x + |x|)$. This can be achieved by defining the conditional decoding attempt probabilities based on independent probabilities d_n of Formula (9) as

$$p_{\boldsymbol{a}|\boldsymbol{c}} = \begin{cases} \prod_{m\in\boldsymbol{a}} d_m \prod_{k\in\bar{\boldsymbol{c}}\setminus\boldsymbol{a}} (1-d_k), & \text{if } \boldsymbol{a}\cap\boldsymbol{c} = \varnothing; \\ 0, & \text{else} \end{cases} \tag{14}$$

where $\bar{\boldsymbol{c}}$ is the complement of $\boldsymbol{c}$ in the file index set, and \ denotes set subtraction. With this conditional probability, we have for the probability of attempting file download combination $\boldsymbol{a}$:

$$p_{\boldsymbol{a}} = \sum_{\boldsymbol{c};\boldsymbol{c}\cap\boldsymbol{a}=\varnothing} p_{\boldsymbol{a}|\boldsymbol{c}}\, p_{\boldsymbol{c}} = \left(\prod_{m\in\boldsymbol{a}} d_m\right) \sum_{\boldsymbol{c};\boldsymbol{c}\cap\boldsymbol{a}=\varnothing} p_{\boldsymbol{c}} \prod_{k\in\bar{\boldsymbol{c}}\setminus\boldsymbol{a}} (1-d_k) \tag{15}$$

and the probability that a UE attempts downloading file n is

$$\begin{aligned} u_n = \sum_{\boldsymbol{a}\ni n} p_{\boldsymbol{a}} &= d_n \sum_{\tilde{\boldsymbol{a}}\subset\tilde{\mathcal{F}}} \prod_{m\in\tilde{\boldsymbol{a}}} d_m \sum_{\boldsymbol{c}\subset\bar{\tilde{\boldsymbol{a}}}} p_{\boldsymbol{c}} \prod_{k\in\bar{\tilde{\boldsymbol{a}}}\setminus\boldsymbol{c}} (1-d_k) = \\ &d_n \sum_{\boldsymbol{c}\subset\tilde{\mathcal{F}}} p_{\boldsymbol{c}} \sum_{\tilde{\boldsymbol{a}}\subset\bar{\boldsymbol{c}}} \prod_{m\in\tilde{\boldsymbol{a}}} d_m \prod_{k\in\bar{\boldsymbol{c}}\setminus\tilde{\boldsymbol{a}}} (1-d_k) = \\ &d_n \sum_{\boldsymbol{c}\subset\tilde{\mathcal{F}}} p_{\boldsymbol{c}} \;=\; d_n(1-c_n) \end{aligned} \tag{16}$$

Here the sum in the first expression is over all sets $\boldsymbol{a}$ that have n as an element, $\tilde{\boldsymbol{a}}$ is the set $\boldsymbol{a}$ with the element n removed, $\tilde{\mathcal{F}}$ is the file index set with n removed, and $\bar{\tilde{\boldsymbol{a}}}$ is the complement of $\tilde{\boldsymbol{a}}$ in $\tilde{\mathcal{F}}$, i.e., it coincides with $\bar{\boldsymbol{a}}$ inside the sum. Furthermore, $\bar{\boldsymbol{c}}$ denotes the complement of $\boldsymbol{c}$ in $\tilde{\mathcal{F}}$. The second line follows from changing the order of summations; we divide $\tilde{\mathcal{F}}$ to the disjoint sets $\tilde{\boldsymbol{a}}$, $\boldsymbol{c}$, and their complement. The last line follows from the fact that for a sum over the power set P of a set S of indices one has

$$\sum_{\boldsymbol{a}\in P} \prod_{m\in\boldsymbol{a}} d_m \prod_{k\in S\setminus\boldsymbol{a}} (1-d_k) = 1 \tag{17}$$

which is a direct consequence of the multinomial theorem. The final equality follows from the fact that the sum of the probability over all cache contents where n is not included is $1-c_n$ by definition. We thus have found that

$$\boldsymbol{u} = \mathrm{diag}\left([d_n]_+\right) \boldsymbol{J}\, \boldsymbol{p}_{\boldsymbol{c}} = \mathrm{diag}\left([d_n]_+\right) (\boldsymbol{1} - \boldsymbol{s}).$$

Using the values of d_n in Formula (9), one sees that this realizes the first part of Eq. (13). A similar argument for $\boldsymbol{P}_{\boldsymbol{b}|\boldsymbol{c}}$ leads to the second part. This policy thus is a minimum energy solution. Each UE executes Algorithm 1 on one sample $\boldsymbol{c}$. In the asymptotic limit of an infinite population, the sample expectation coincides with the probabilistic one, and $\boldsymbol{s}^+$ is realized. ■

Note that there may be a continuum of UE cache update policies that would realize $s_n(t+1)$ starting from $s_n(t)$ with the same energy consumption. Algorithm 1 is set apart by its communication complexity—instead of dictating messages consisting of large-dimensional matrices, only an N-dimensional vector is needed. Note that this is the minimum overhead—due to the outage probabilities, the dictating variables do not sum up to 0.

4 Cache policy formulation

With the probabilistic caching UE-cach update policy define, we concentrate on determining the cache policy of the network. The objective is to maximize Quality of Service (QoS) while minimizing the network backhaul load and UE power consumption under the time-varying file request dynamics Eq. (7).

4.1 State, observation, and action

For the cache policy optimization problem, the state of the system is defined as the vector

$$\boldsymbol{x}(t) = \left[\boldsymbol{p}(t)^{\mathrm{T}}, \boldsymbol{s}(t)^{\mathrm{T}}, \boldsymbol{f}(t)^{\mathrm{T}}\right]^{\mathrm{T}} \tag{18}$$

The system state consists of the realization of the stochastic request process Eq. (7), the content of the UE caches, and the time-varying file popularity distribution.

While the network has access to the file requests $\boldsymbol{p}(t)$

and the UE cache contents $\boldsymbol{s}(t)$, the underlying file popularity distribution $\boldsymbol{f}(t)$ remains unknown to the network. Consequently, we define an observation vector as

$$\boldsymbol{q}(t) = [\boldsymbol{p}(t)^{\mathrm{T}}, \boldsymbol{s}(t)^{\mathrm{T}}]^{\mathrm{T}} \tag{19}$$

Accordingly, the observation space is given as

$$\mathcal{Q} = \{(\boldsymbol{p}, \boldsymbol{s}) \mid p_n \geqslant 0,\ \mathbf{1}^{\mathrm{T}}\boldsymbol{p} = 1,\ s_n \geqslant 0,\ \mathbf{1}^{\mathrm{T}}\boldsymbol{s} \leqslant L_u\}.$$

The network is equipped with three control variables: the BS cache probability $\{\rho_n(t)\}_{n=1}^{N}$, the resource allocation $\{w_n(t)\}_{n=1}^{N}$ used by the network when transmitting files by OMPMC, and the dictating messages $\{d_n(t)\}_{n=1}^{N}$. We then define the action vector

$$\boldsymbol{u}(t) = [\boldsymbol{\rho}(t)^{\mathrm{T}}, \boldsymbol{w}(t)^{\mathrm{T}}, \boldsymbol{d}(t)^{\mathrm{T}}]^{\mathrm{T}} \tag{20}$$

The action space is given by

$$\begin{aligned}\mathcal{U} =& \{(\boldsymbol{\rho}, \boldsymbol{w}, \boldsymbol{d}) \mid \rho_n \geqslant 0,\ \mathbf{1}^{\mathrm{T}}\boldsymbol{\rho} \leqslant L_b,\\ & w_n \geqslant 0,\ \mathbf{1}^{\mathrm{T}}\boldsymbol{w} = 1,\ -1 \leqslant d_m \leqslant 1\}\end{aligned} \tag{21}$$

Accordingly, the process of file retrieval is controlled by the variables provided by the RL agent. Using the optimal policy, it determines which files are essential for proactive caching by the BSs from the core network, hence maintaining the consistency and reliability of the system.

4.2 POMDP

The file request dynamics in Eq. (7) involves the outage probability $o_n(t)$, which is a function of resource allocation $w_n(t)$ as given in Eq. (2). Since $w_n(t)$ is an action variable, the dynamics leads to a Markov Decision Process (MDP). As the underlying file popularity $f_n(t)$, modeled in Eq. (8), is time-varying we have a non-stationary MDP. Furthermore, since $f_n(t)$ is unknown to the agent, our cache policy formulation leads to a N-POMDP. The objective of this work is thus to formulate a cache policy based on this N-POMDP.

An N-POMDP is characterized by a tuple $(\mathcal{X}, O, \mathcal{U}, P_T(\cdot;t), P_O(\cdot;t), r(\cdot))$, where $\mathcal{X}$ represents the state space, $O \subset \mathcal{X}$ denotes the observation space, and $\mathcal{U}$ signifies the action space. The time-varying transition probability $P_T(\cdot;t)$ describes the system environment, while the time-varying observation distribution $P_O(\cdot;t)$ and the immediate reward function $r(\cdot)$ provide additional information about the environment and the agent's performance.

The system state at time t is represented by $\boldsymbol{x}(t) \in \mathcal{X}$. The observation and action at the same time instance are denoted by $\boldsymbol{q}(t) \in O$ and $\boldsymbol{u}(t) \in \mathcal{U}$, respectively. The transition probability $P_T(\boldsymbol{x}(t+1) \mid \boldsymbol{x}(t), \boldsymbol{u}(t); t)$ indicates the time-variant probability that being in state $\boldsymbol{x}(t)$ and performing action $\boldsymbol{u}(t)$ will result in the next state $\boldsymbol{x}(t+1)$. It is important to note that in our model, the transition probability changes over time, reflecting the dynamic nature of the environment.

In a POMDP framework, the state $\boldsymbol{x}(t)$ is not directly observable for an agent interacting with the environment. Instead, the agent has access to the observation $\boldsymbol{q}(t)$, which is drawn from the distribution $P_O(\cdot \mid \boldsymbol{x}(t+1); t)$. This means that the agent must make decisions based on incomplete information of the environment state, adding a layer of complexity to the decision-making process.

In the context of file popularity, this framework is adept at modeling environments with non-stationary dynamics, relevant for situations with time-variant file popularities and UE caching probabilities. More specifically, it captures the time-varying nature of transition probability, observation distribution, and reward function, accurately modeling the intricate cache policy dynamics. The uncertainty and incomplete information of the cache policy problem is modelled based on an N-POMDP problem. By leveraging learning techniques robust against N-POMDP environment such as LSTM-based reinforcement learning, we can find effective solutions for cache policies[29]. The approach facilitated by the POMDP encourages the development of joint cache placement and delivery policies that maximize system performance, while also revealing the underlying structure and dynamics of the cache policy problem for a deeper understanding of influential factors.

4.3 Optimization objective

This paper focuses on three metrics for evaluating network performance. The first metric is QoS, which measures the likelihood of a requesting UE being satisfied by the OMPMC networking. This metric can

be quantified as the probability of successful requests compared to the total number of requests made by UEs.

$$c_{\text{qos}}(t) = \sum_{n=1}^{N} p_n(t)\left(1 - s_n(t)\right) o_n(t) \tag{22}$$

From Eq. (22), if the file is not cached then the cost is directly proportional to the outage probability when minimized the QoS metric improves. It is important to note that the QoS metric depends on $p_n(t)$, which is a part of the state vector.

Next, we examine the backhaul load associated with the retrieval of files by the BSs from the core network. When the difference between the file load at time t and the previous time $t-1$, i.e., $\rho_n(t) - \rho_n(t-1)$ is less than or equal to zero, it indicates that there is no backhaul load since no files are being fetched. However, if the difference is greater than zero, it implies that certain BSs are required to cache file n due to the presence of a backhaul load. This makes the backhaul load depends on the action vector. The backhaul cost function is defined as[24]

$$c_{\text{bh}}(t) = \sum_{n=1}^{N} \left[\rho_n(t) - \rho_n(t-1)\right]_+ \tag{23}$$

Note that this assumes that the BSs are conservative in filling their caches, they do not. This backhaul function can be realized, e.g., by BSs following the method of Section 2.1, such that each BS keeps it random variable U determining the cache content static, while the probability weights of the files change from time instant to next.

Thirdly, we investigate a power consumption metric that arises from UEs downloading files according to their preferences, and for updating their caches. We assume that discarding files does not cost energy. As each UE not caching file n attempts to download it with probability $d_n(t)$, the power consumption metric depends both on the state and the action vector:

$$c_{\text{uep}}(t) = \sum_{n=1}^{M} \left(1 - s_n(t)\right)\left(p_n(t) + \left[d_n(t)\right]_+\right) \tag{24}$$

This measures the average number of file download attempts that users perform both to fulfill their requests, and for proactive caching.

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

$$r_{\text{wt}}(t) = -\left(c_{\text{qos}}(t) + \lambda_{\text{bh}} c_{\text{bh}}(t) + \lambda_{\text{uep}} c_{\text{uep}}(t)\right) \tag{25}$$

where λ_{bh} and λ_{uep} are the Lagrange multipliers for backhaul and UE power consumption costs. Considering the dynamics introduced by the stochastic difference Eq. (7), we formulate a discounted cumulative reward starting from time-slot t until a time horizon T,

$$R_{\text{ac}}(t) = \sum_{k=t}^{T} \gamma^{k-t} E\left[r_{\text{wt}}(k)\right] \tag{26}$$

where $E(\cdot)$ is the expectation operator, $\gamma \in [0,1)$ 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.

The objective is to find a probabilistic cache policy $\pi(\boldsymbol{u}|\boldsymbol{q})$, which defines the probability of action $\boldsymbol{u}(t)$ given the observation $\boldsymbol{q}(t)$. This policy is obtained by addressing the constrained optimization problem:

$$\begin{aligned} P_1: \ & \max_{\{\pi(\cdot|\boldsymbol{q})\}} R_{\text{ac}}(t),\ 0 \leqslant t \leqslant T, \\ & \text{s.t.},\ \ (\boldsymbol{\rho}, \boldsymbol{w}, \boldsymbol{d}) \in \mathcal{U} \end{aligned} \tag{27}$$

The utility cost function is defined in Eqs. (22)−(26), while the underlying file request dynamics are defined in Eqs. (2)−(7).

5 Reinforcement learning framework

We employ an RL agent to obtain the optimal policy $\pi^*(\cdot|\boldsymbol{q}(t))$ that solves problem P_1. To facilitate this, we utilize neural networks for functional approximation, leveraging their universal approximation capabilities[30].

Among several RL algorithms, we use a policy gradient algorithm: A2C. A2C offers several benefits over Deep Q-Network (DQN) in various reinforcement learning scenarios. A2C directly optimizes the policy using the actor network, enabling it to learn stochastic policies that can be advantageous in non-deterministic environments[31] or where exploration is crucial. In contrast, DQN optimizes the action-value function.

Consequently, A2C stochastic policy can lead to more efficient exploration and faster convergence to an optimal policy compared to DQN's epsilon-greedy strategy. A2C method, as on-policy algorithm, is particularly adept at handling dynamic environments, where file requests are non-stationary. It optimizes policy based on ongoing experiences, providing an effective exploration-exploitation trade-off, as the actor learns the best policy and the critic evaluates it, often aided by an entropy term to encourage further exploration[24]. A2C continually learns from new experiences, a feature less prominent in algorithms like DQN which rely on a replay buffer of past experiences.

Furthermore, A2C is better suited for tasks with continuous action spaces as it can handle them naturally, while DQN requires discretization which can lead to loss in precision[32]. The parallelism and scalability of A2C make it an attractive choice for large-scale problems, as it can be easily parallelized using multiple workers. Finally, A2C advantage function estimates the value of taking an action, reducing the variance of the policy gradient and resulting in more stable learning and potentially better convergence properties.

The A2C algorithm consists of two neural networks. The actor network, embedded within the RL agent, provides a policy distribution $\pi_\theta(\cdot)$ parameterized by θ, facilitating interaction with the environment. Additionally, the critic network approximates the state-value function, which is defined as

$$V(\boldsymbol{q}(t)) = E\left[\sum_{k=t}^{T} \gamma^{k-t} r_{\mathrm{wt}}(k) \,\middle|\, \boldsymbol{q}(t)\right] \tag{28}$$

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

5.1 LSTM cell

LSTM networks, a specialized subset of recurrent neural networks, are crafted to tackle challenges in learning long-term dependencies within sequential data. Central to the LSTM is the cell state, acting as a parallel memory unit that facilitates prolonged information retention. The internal architecture features three pivotal gates—input, forget, and output—regulating the influx, relevance, and output of information. This enables LSTM to selectively update and employ information, effectively addressing the vanishing gradient problem inherent in traditional recurrent networks. Additionally, the LSTM includes a hidden state for short-term memory and incorporates feedback connections[22, 33] spanning various time steps, allowing it to adeptly capture dependencies across extended sequences. This dynamic interplay, coupled with the adaptive updates of the cell state through input and forget gates, enables LSTM to distinguish and retain essential information while discarding extraneous details. We incorporate LSTM architecture to address the N-POMDP environment of study formulated in Section 4.

5.2 Actor-critic and state processing network

The A2C architecture comprises of three distinct networks: a state processing network, an actor network, and a critic network as in Fig. 3. To address the non-stationarity of the environment as formulated in Section 4, we incorporate an LSTM architecture for state processing. An LSTM cell includes feedback connections[22, 23], a distinctive feature absent in Feed-Forward Neural Networks (FFNN). The state processing network incorporates an LSTM layer and a fully connected layer. With LSTM, the network processes a sequence of observation vectors ranging from two to six time-slots, generating input for the fully connected layer. For example, LSTM with six time-slots processes the observation vectors $[\boldsymbol{q}(t-5), \boldsymbol{q}(t-4), \ldots, \boldsymbol{q}(t)]$ and is denoted as $\mathrm{LSTM}_{6\mathrm{s}}$. The output of the fully connected layer serves as input for both the actor and critic networks.

We bench mark LSTM-based state processing with a two time slot FFNN state processing architecture, denoted by $\mathrm{FF}_{2\mathrm{s}}$. It processes observation vectors $[\boldsymbol{q}(t-1), \boldsymbol{q}(t)]$, and consists of two fully connected layers.

The actor and critic network take input from the state processing network. The actor network produces a probabilistic caching policy, while the critic network provides an estimate of the value function. Each of the actor and critic networks consists of an independent

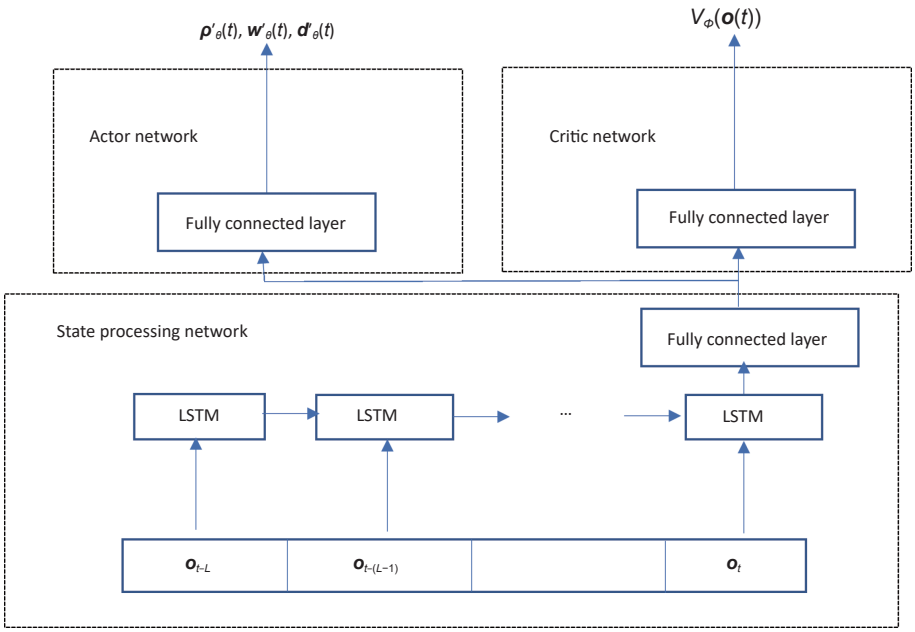

Fig. 3 LSTM-based A2C architecture.

fully connected layer, receiving input from the state processing unit.

To guarantee normalized outputs, we draw the action vector at random from two independent Dirichlet distributions such that

$$\boldsymbol{u}(t) \sim \left[L_b \,\mathrm{Dirich}(\boldsymbol{\rho}'_\theta(t))^{\mathrm{T}}, \mathrm{Dirich}(\boldsymbol{w}'_\theta(t))^{\mathrm{T}}, \boldsymbol{d}'_\theta(t)^{\mathrm{T}}\right]^{\mathrm{T}},$$

where $\boldsymbol{\rho}'_\theta(t)$, $\boldsymbol{w}'_\theta(t)$, and $\boldsymbol{d}'_\theta(t)$ are the outputs of the actor network parameterized by θ, and Dirich(·) stands for the multivariate Dirichlet distribution. Utilizing Dirichlet distributions offers notable benefits, such as improved exploration-exploitation balance due to providing random actions with learnable parameters controlling the stochasticity. This approach allows the model to capture uncertainty across multiple actions as well as ensures compliance with control parameter constraints Eq. (21).

5.3 A2C algorithm

The RL actor network generates an action for a given state based on the parametric policy distribution $\pi_\theta(\cdot|\boldsymbol{q}(t))$. When the agent interacts with the environment, it receives an immediate reward and progresses to the next state. The critic network produces an estimate of the state-value function $V_\phi(\boldsymbol{q}(t))$ for the current observation. The immediate reward, action vector, and estimated state-value function is stored in a buffer, which is used to update the actor and critic parameters after T time-slots. This entire process forms a single episodic trajectory, with multiple trajectories needed for the training process. The parameter of the actor and critic networks are changed as[34]

$$\begin{aligned}\Delta\phi =& \alpha_\phi \sum_{t=1}^{T} A_\phi(\boldsymbol{q}(t),\boldsymbol{u}(t))\,\nabla_\phi V_\phi(\boldsymbol{q}(t)),\\ \Delta\theta =& \alpha_\theta \sum_{t=1}^{T} [A_\phi(\boldsymbol{q}(t),\boldsymbol{u}(t))\,\nabla_\theta \log(\pi_\theta(\boldsymbol{u}(t)|\boldsymbol{q}(t)))+\\ & \beta\nabla_\theta H(\pi_\theta(\boldsymbol{u}(t)|\boldsymbol{q}(t)))]\end{aligned} \tag{29}$$

where α_ϕ and α_θ are the learning rates of the critic and actor agents, respectively,

$$A_\phi(\boldsymbol{q}(t),\boldsymbol{u}(t)) = r_{\mathrm{wt}}(\boldsymbol{q}(t),\boldsymbol{u}(t)) + \gamma V_\phi(\boldsymbol{q}(t+1)) - V_\phi(\boldsymbol{q}(t)) \tag{30}$$

is an estimate of the accumulated reward Eq. (26), and $H(\cdot)$ is the entropy term used with regularization factor β for trading off between exploration and exploitation in order to prevent A2C from converging to sub-optimal policies.

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.

Algorithm 2 Modified A2C algorithm

Input: A θ-parametrized policy distribution $\pi_\theta(\boldsymbol{u}(\cdot)|\boldsymbol{q}(\cdot))$, ϕ-parametrized approximation of state-value function $V_\phi(\boldsymbol{q}(\cdot))$.
Output: optimal solution θ of problem P_1.
for update = 1 **to** $E_{\max}$ **do**
  if update == 1 **then**
    Initialize observation vector $\boldsymbol{q}(1)$ to some random point.
  else
    Set $\boldsymbol{q}(1) = \boldsymbol{q}_T$ based on previous update.
  end if
  for $t = 1$ **to** T **do**
    Draw action $\boldsymbol{u}(t)$ from policy $\pi_\theta(\boldsymbol{u}(t)|\boldsymbol{q}(t))$.
    Fit value function, $V_\phi(\boldsymbol{q}(t))$ from the critic network.
    Evaluate the advantage function $A_\phi(\boldsymbol{q}(t),\boldsymbol{u}(t)) = r(t+1)+\gamma V_\phi(\boldsymbol{q}(t+1)) - V_\phi(\boldsymbol{q}(t))$.
    Get new observation vector $\boldsymbol{q}(t+1)$, and immediate reward $r(t+1)$.
    Buffer $V_\phi(\boldsymbol{q}(t))$, $\boldsymbol{q}(t)$, $r(t+1)$, $\log(\pi_\theta(\boldsymbol{u}(t)|\boldsymbol{q}(t)))$, $H(\pi_\theta(\boldsymbol{u}(t)|\boldsymbol{q}(t)))$.
  end for
  Update parameters for actor and critic networks based on the updating rules Eq. (29).
  Set $\boldsymbol{q}_T = \boldsymbol{q}(T)$.
end for

6 Simulation and discussion

In order to assess the performance of the proposed DRL cache policy, we examine a cellular network consisting of BSs and UEs positioned according to two independent PPPs.

In our experimental setup, we follow the urban Non-Line of Sight (NLOS) conditions defined by 3GPP[35], with carrier frequency 2 GHz and BS transmission power of 28 dBm. The BS antenna gain is 8 dBi, while UE antenna gain is 0 dBi. The noise power spectrum density is −174 dBm, the noise figure of the UEs is 9 dB, and the bandwidth is 2 MHz. We consider the path-loss exponent $\beta_{\text{pl}} = 4$, with the path loss given by $L = 128 + \beta_{\text{pl}} \lg d$, with distance d measured in kilometers. Hence, the reference Signal to Noise Ratio (SNR) at the reference distance of 1 km is $\frac{\overline{p}}{\sigma_0^2} \approx 1$. We set the BS density to $\lambda_{\text{bs}} = 300$ BSs per km^2.

For the diffusion model in Eq. (8), we employ a Zipf distribution with skewness $\tau = 0.6$, while the probability for a user to patient, used in Eq. (4) is $\zeta \in \{0.1, 0.9\}$. The system parameters are summarized in Table 1.

Table 1 Simulation parameters.

System parameter	Value
BS capacity, L_b	6
UE capacity, L_u	3
Number of files, N	40
Total number of updates, $E_{\max}$	10 000
Cost regularization parameters, $\lambda_{\text{bh}}, \lambda_{\text{uep}}$	0.05
Entropy regularization term, β	0.005
File skewness, τ	0.6
Target spectral efficiency, α	0.1
BS intensity, λ_{bs}	300
RL update after T slots	256
Probability of UE patience, ζ	0.1, 0.9

In the state-processing unit of the LSTM architecture, LSTM output is linked to a fully connected hidden layer of 128 neurons with tanh activation function. The actor and critic networks consist of one fully connected hidden layer of 128 neurons activated by a tanh(·) function. The output activation functions for the actor and critic networks are Relu(·) and Softplus(·), respectively. The learning rates are set to $\alpha_\theta = \alpha_\phi = 10^{-3}$, and Adam optimizer is used.

For the hyperparameters of Algorithm 2, we set the total number of updates to $E_{\max} = 10^4$, the Lagrange multipliers of the weighted reward in Eq. (25) to $[\lambda_{\text{bh}}, \lambda_{\text{uep}}] = [0.05, 0.05]$, and the entropy regulation term to $\beta = 0.005$. The system parameters used in the simulations are configured accordingly.

Figure 4 presents the training performance of FFNN and LSTM architectures when they process the observations of two time-slots. Performance is measured in terms of normalized discounted cumulative costs of QoS, backhaul and UE power consumption;

$$c_{\text{qos}} = \frac{1}{T}\sum_{t=1}^{T}\gamma^{t-1}c_{\text{qos}}(t),$$

$$c_{\text{bh}} = \frac{1}{T}\sum_{t=1}^{T}\gamma^{t-1}c_{\text{bh}}(t),$$

$$c_{\text{uep}} = \frac{1}{T}\sum_{t=1}^{T}\gamma^{t-1}c_{\text{uep}}(t),$$

plotted as a function of update samples. The results indicate that LSTM outperforms FFNN in all cost

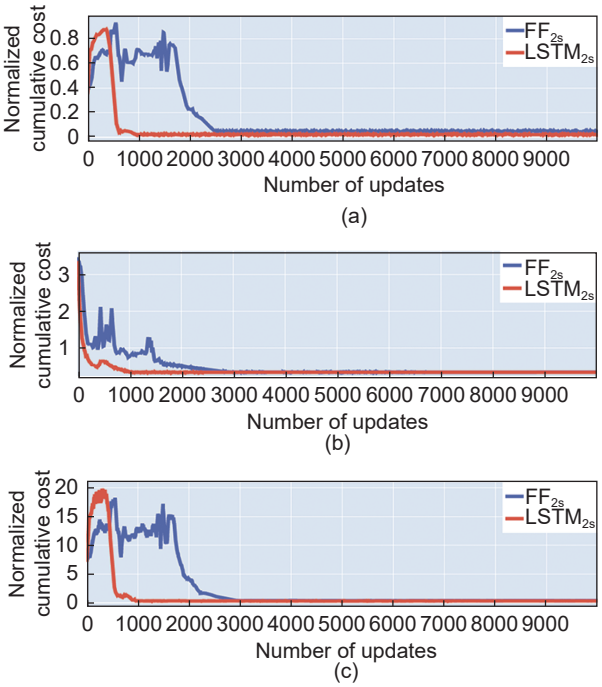

Fig. 4 Training performance of LSTM and FFNN with two states for γ = 0.98. (a) Normalized cumulative QoS cost. (b) Normalized cumulative backhaul cost. (c) Normalized cumulative UE power consumption cost.

metrics from a sample-efficiency standpoint. LSTM converges in approximately one-third of the number of samples as compared to FFNN.

The LSTM architecture is thus better at capturing the non-stationarity of the POMDP environment than conventional FFNN. This can be attributed to the feedback connections of the LSTM network, which play a crucial role in learning the evolution of file popularity $\{f_n(t)\}_{n=1}^{N}$ that is latent in the observation vector.

In order to establish a benchmark for the RL-based cache policy, we consider a static cache solution obtained using an interior-point algorithm. The policy is independently optimized in each time-slot based on the immediate reward rather than a cumulative reward. We refer to this solution as "Static".

Figures 5 and 6 showcase the test performance of the static optimization solution in comparison to dynamic RL-based solutions with various Neural Network (NN) architectures. In Fig. 5, impatient users with $\zeta = 0.1$ are consdered, while in Fig. 6, the users are patient, with

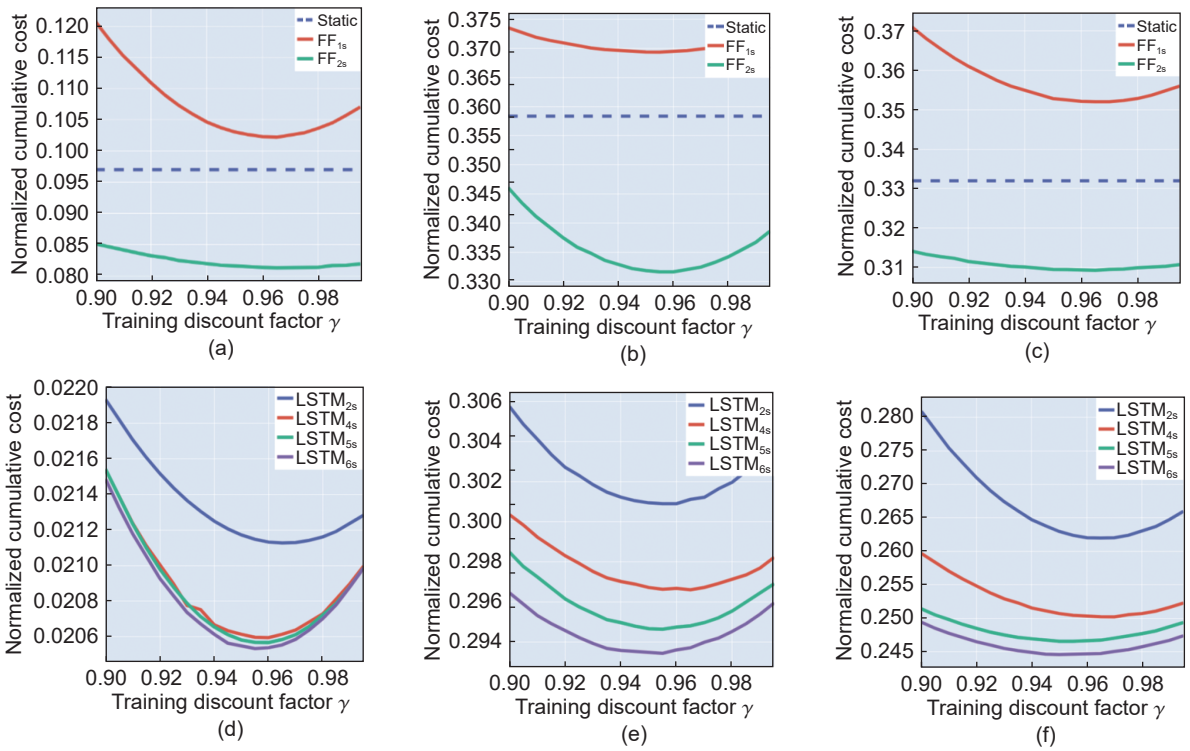

Fig. 5 Test performance for impatient users with ζ = 0.1 for static and RL-based solutions with different NN architectures as a function of discount factor. (a) and (d) show QoS cost. (b) and (e) show backhaul cost. (c) and (f) show UE power consumption cost.

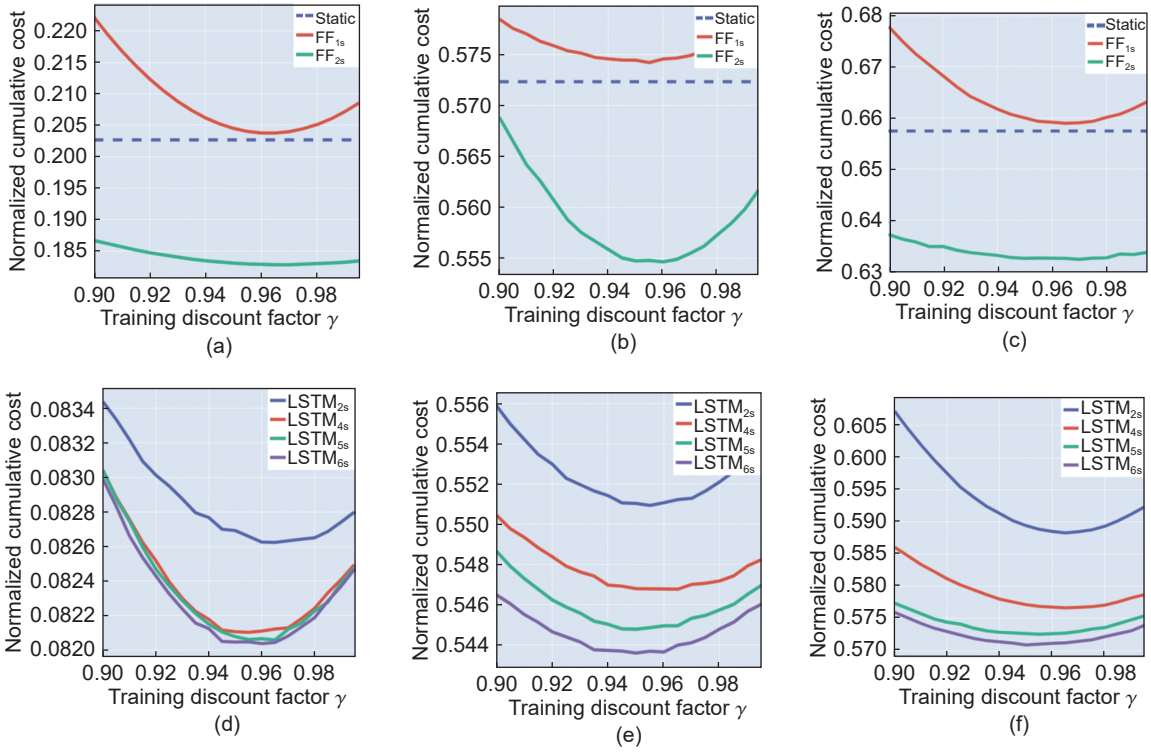

Fig. 6 Test performance when for patient users with $\zeta = 0.9$ for static and RL-based solutions with different NN architectures as a function of discount factor. (a) and (d) show QoS cost. (b) and (e) show backhaul cost. (c) and (f) show UE show power consumption cost.

$\zeta = 0.9$. For the RL-based solutions, the agent is trained for different values of the discount factor $\gamma \in [0.9, 1)$, after which the agent is evaluated during a test scenario with discount factor $\gamma = 1$. The cumulative costs c_{qos}, c_{bh}, and c_{uep} averaged over the testing time duration T are shown.

Figures 5 and 6 show that the LSTM architecture outperforms FFNN in terms of optimality. Additionally, increasing the observation length of LSTM cells leads to a decrease in all cumulative costs. This implies that incorporating a longer observation history improves the RL agent's performance due to an enhanced ability to uncover latent information. The static solution performs better than the dynamic RL solution of an FFNN with one state. However, when compared to the dynamic RL solutions that exploit multiple time slots, regardless of whether FFNN or LSTM is used, the static solution performs poorly. This outcome validates the use of cumulative reward optimization as in P_1 for dynamic cache policy design. Also, this demonstrates that the developed A2C algorithm can find a near optimal solution for the formulated POMDP problem. Comparing Figs. 5 and 6, we see that for patient users with $\zeta = 0.9$, all costs are higher. This is understandable; if a user insists on getting a rare file, the network is forced to use resources for that file, which reduces the performance related to more popular files.

In order to assess how the partial observability influences performance, we consider an MDP environment, where the state vector $\boldsymbol{x}(t)$ from Eq. (18) is utilized instead of the observation vector $\boldsymbol{q}(t)$. Figures 7 and 8 compare the test performance in POMDP and MDP environments in terms of average cumulative costs with $\gamma = 1$. The POMDP approaches are shown for training discount factors in the range $\gamma \in [0.9, 1)$. The MDP solutions are trained for $\gamma = 0.98$, and are represented by dotted lines. Figure 7 shows patient users ($\zeta = 0.9$) and Fig. 8 impatient ones ($\zeta = 0.1$).

Figures 7 and 8 show that the difference between cumulative costs for MDP and POMDP is significantly

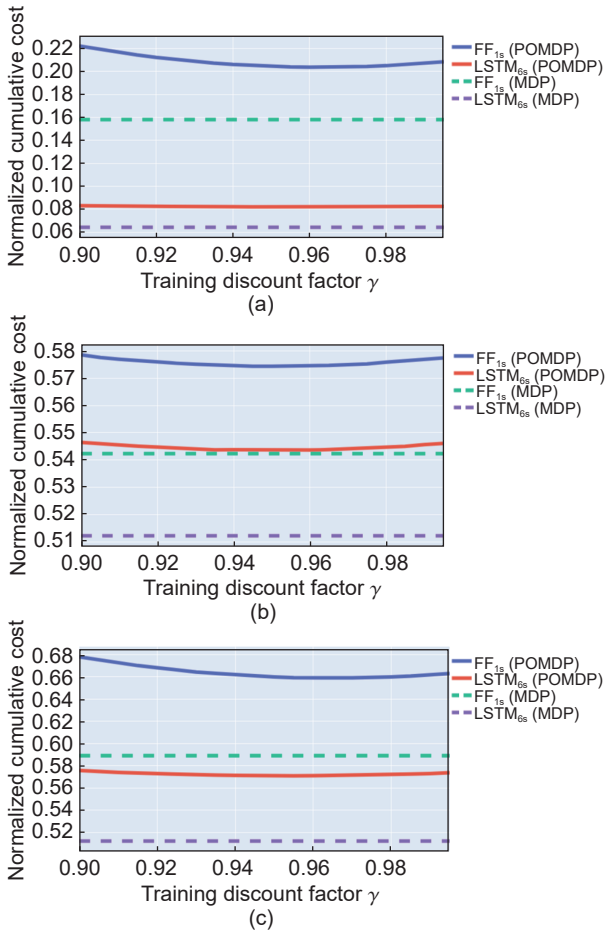

Fig. 7 Test performance for patient users with ζ = 0.9 for POMDP and MDP environments. (a) Normalized cumulative QoS cost. (b) Normalized cumulative backhaul cost. (c) Normalized cumulative UE power consumption cost.

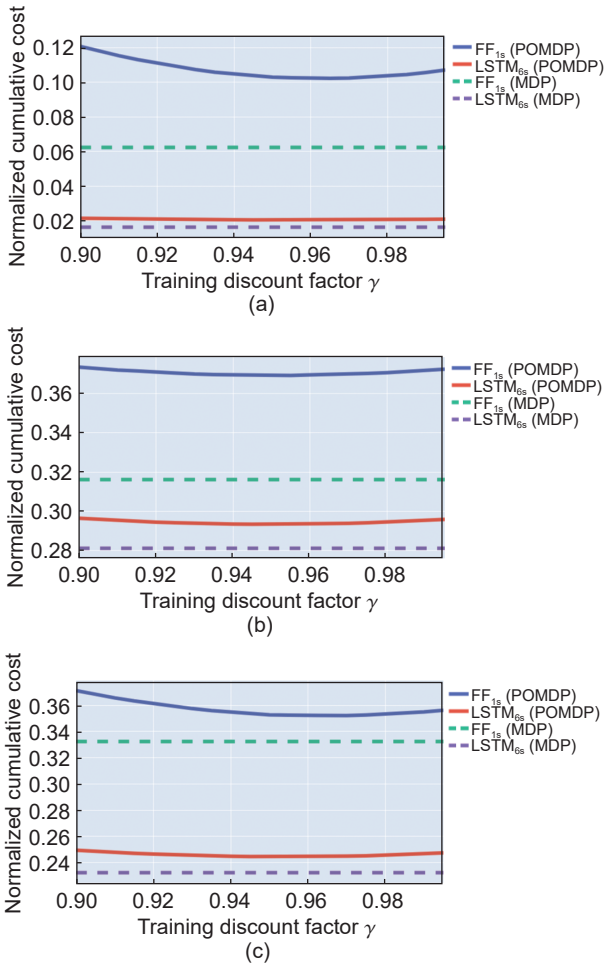

Fig. 8 Test performance when ζ = 0.1 for POMDP and MDP environments. (a) Normalized cumulative QoS cost. (b) Normalized cumulative backhaul cost. (c) Normalized cumulative UE power consumption cost.

smaller for the LSTM architecture as compared to the FFNN architecture, suggesting that LSTM is more capable of handling the complexity and uncertainties associated with POMDP settings.

7 Conclusion

In this study, we present a dynamic cache placement and delivery problem optimized by an RL algorithm. Files are proactively cached both at the BSs and at UEs. Our objective is to optimize the quality of service, backhaul load, and UE power consumption by leveraging the A2C algorithm. We first provide a distributed UE cache placement algorithm, where the population of UEs achieves a target distribution of cached files, which minimizes the UE energy consumption in terms of file decoding attempts. Next, we investigate the RL agents at the network side for optimizing the cache placement and delivery parameters in an orthogonal multipoint multicasting networking scenario. Two types of architectures for the RL agent are considered: FFNN and LSTM. The motivation behind exploring these two architectures is to identify the most suitable approach for addressing the considered problem. Our simulation results not only provide justification for utilizing the POMDP for problem formulation but also demonstrate that the proposed LSTM-based A2C surpasses the FFNN-based A2C in terms of sample efficiency and optimality. Moreover, our findings indicate that the LSTM-based A2C can deliver significantly improved performance in a POMDP environment compared to its FFNN

counterpart. This highlights the potential of using LSTM-based methods in reinforcement learning scenarios where the problem formulation requires consideration of the POMDP framework.

Acknowledgment

This work was supported in part by the Academy of Finland (No. 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 Trans. Inf. 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, no. 15, p. 5033, 2021.

[3] B. Serbetci and J. Goseling, On optimal geographical caching in heterogeneous cellular networks, in *Proc. IEEE Wireless Communications and Networking Conf.* (*WCNC*), San Francisco, CA, USA, 2017, pp. 1–6.

[4] B. Blaszczyszyn and A. Giovanidis, Optimal geographic caching in cellular networks, in *Proc. IEEE Int. Conf. Communications* (*ICC*), London, UK, 2015, pp. 3358–3363.

[5] Y. Chen, M. Ding, J. Li, Z. Lin, G. Mao, and L. Hanzo, Probabilistic small-cell caching: Performance analysis and optimization, *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 4341–4354, 2017.

[6] X. Xu and M. Tao, Modeling, analysis, and optimization of caching in multi-antenna small-cell networks, *IEEE Trans. Wirel. Commun.*, vol. 18, no. 11, pp. 5454–5469, 2019.

[7] M. Choi, A. F. Molisch, D. J. Han, D. Kim, J. Kim, and J. Moon, Probabilistic caching and dynamic delivery policies for categorized contents and consecutive user demands, *IEEE Trans. Wirel. Commun.*, vol. 20, no. 4, pp. 2685–2699, 2021.

[8] J. Wu, B. Chen, C. Yang, and Q. Li, Caching and bandwidth allocation policy optimization in heterogeneous networks, in *Proc. IEEE 28th Annual Int. Symp. on Personal, Indoor, and Mobile Radio Communications* (*PIMRC*), Montreal, Canada, 2017, pp. 1–6.

[9] J. Wen, K. Huang, S. Yang, and V. O. K. Li, Cache-enabled heterogeneous cellular networks: Optimal tier-level content placement, *IEEE Trans. Wirel. Commun.*, vol. 16, no. 9, pp. 5939–5952, 2017.

[10] K. Li, C. Yang, Z. Chen, and M. Tao, Optimization and analysis of probabilistic caching in *N*-tier heterogeneous networks, *IEEE Trans. Wirel. Commun.*, vol. 17, no. 2, pp. 1283–1297, 2018.

[11] J. Wu, C. Yang, and B. Chen, Proactive caching and bandwidth allocation in heterogenous networks by learning from historical numbers of requests, *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 4394–4410, 2020.

[12] Z. Wang, Z. Cao, Y. Cui, and Y. Yang, Joint and competitive caching designs in large-scale multi-tier wireless multicasting networks, in *Proc. GLOBECOM 2017—2017 IEEE Global Communications Conf.*, Singapore, 2017, pp. 1–7.

[13] Y. Cui and D. Jiang, Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks, *IEEE Trans. Wirel. Commun.*, vol. 16, no. 1, pp. 250–264, 2017.

[14] C. Ye, Y. Cui, Y. Yang, and R. Wang, Optimal caching designs for perfect, imperfect, and unknown file popularity distributions in large-scale multi-tier wireless networks, *IEEE Trans. Commun.*, vol. 67, no. 9, pp. 6612–6625, 2019.

[15] M. Bayat, R. K. Mungara, and G. Caire, Achieving spatial scalability for coded caching via coded multipoint multicasting, *IEEE Trans. Wirel. Commun.*, vol. 18, no. 1, pp. 227–240, 2019.

[16] X. Peng, Y. Shi, J. Zhang, and K. B. Letaief, Layered Group sparse beamforming for cache-enabled green wireless networks, *IEEE Trans. Commun.*, vol. 65, no. 12, pp. 5589–5603, 2017.

[17] W. Sun, Y. Li, C. Hu, and M. Peng, Joint optimization of cache placement and bandwidth allocation in heterogeneous networks, *IEEE Access*, vol. 6, pp. 37250–37260, 2018.

[18] F. Zhou, L. Fan, N. Wang, G. Luo, J. Tang, and W. Chen, A cache-aided communication scheme for downlink coordinated multipoint transmission, *IEEE Access*, vol. 6, pp. 1416–1427, 2018.

[19] M. Amidzadeh, H. Al-Tous, G. Caire, and O. Tirkkonen,

Caching in cellular networks based on multipoint multicast transmissions, *IEEE Trans. Wirel. Commun.*, vol. 22, no. 4, pp. 2393–2408, 2023.

[20] 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 *Proc. IEEE Int. Conf. Communications Workshops* (*ICC Workshops*), Kansas City, MO, USA, 2018, pp. 1–6.

[21] 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 *Proc. IEEE Wireless Communications and Networking Conf.* (*WCNC*), Marrakesh, Morocco, 2019, pp. 1–6.

[22] 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 Commun. Lett.*, vol. 24, no. 9, pp. 2005–2009, 2020.

[23] Z. Zhang and M. Tao, Deep learning for wireless coded caching with unknown and time-variant content popularity, *IEEE Trans. Wirel. Commun.*, vol. 20, no. 2, pp. 1152–1163, 2021.

[24] M. Amidzadeh, H. Al-Tous, O. Tirkkonen, and J. Zhang, Joint cache placement and delivery design using reinforcement learning for cellular networks, in *Proc. IEEE 93rd Vehicular Technology Conf.* (*VTC2021-Spring*), Helsinki, Finland, 2021, pp. 1–6.

[25] T. Ni, B. Eysenbach, and R. Salakhutdinov, Recurrent model-free RL is a strong baseline for many POMDPs, arXiv preprint arXiv: 2110.05038, 2021.

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

[27] M. Chiang, *Networked Life*. Cambridge, UK: Cambridge University Press, 2012.

[28] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, Web caching and Zipf-like distributions: Evidence and implications, in *Proc. IEEE Annual Joint Conference*: *INFOCOM, IEEE Computer and Communications Societies*, New York, NY, USA, 1999, pp. 126–134.

[29] M. Kumar, R. Rout, and D. Somayajulu, Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks, *Comput. Netw.*, vol. 209, p. 108876, 2022.

[30] E. Paluzo-Hidalgo, R. Gonzalez-Diaz, and M. A. Gutiérrez-Naranjo, Two-hidden-layer feed-forward networks are universal approximators: A constructive approach, *Neural Netw.*, vol. 131, pp. 29–36, 2020.

[31] A. Baisero and C. Amato, Unbiased asymmetric reinforcement learning under partial observability, arXiv preprint arXiv: 2105.11674v2, 2022.

[32] S. Nath and J. Wu, Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems, *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.

[33] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] 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. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, 2012.

[35] 3GPP, UMTS Universal Mobile Telecommunications System, RF system scenarios (3GPP TR 25.942 version 14.0. 0), Tech. Rep. ETSI TR 125 942, 3GPP, 2017.

Mohsen Amidzadeh received the BSc and MSc degrees in electronic engineering from Sharif University of Technology, Tehran, Iran in 2010 and 2012, respectively. He is currently pursuing the PhD degree at the Department of Information and Communications Engineering, Aalto University, Finland. He completed a sabbatical program at University of Alberta, Edmonton, Canada, in 2018. His current research interests include next-generation cellular networks, wireless communications, machine learning, optimization problems, and estimation theory.

Ashvin Srinivasan received the BEng degree in communications engineering from VTU, Belgaum, India in 2009, and the MSc degree in communications engineering from Aalto university, Finland in 2012. He is currently pursuing the PhD degree at the Department of Information and Communications Engineering, Aalto University, Finland. His current research interests include wireless communications and machine learning.

Junshan Zhang received the PhD degree from Purdue University, USA in 2000. He was on the faculty of the School of Electrical, Computer and Energy Engineering, Arizona State University, USA from 2000 to 2021. He is currently a professor at the Department of Electrical and Computer Engineering, University of California, Davis, USA. He was the recipient of the ONR Young Investigator Award in 2005 and the NSF CAREER award in 2003. He received the IEEE Wireless Communication Technical Committee Recognition Award in 2016. His papers have won a few awards, including the Best Student Paper Award at WiOPT 2018, the Kenneth C. Sevcik Outstanding Student Paper Award of ACM SIGMETRICS/IFIP Performance 2016, the Best Paper Runner-up Award of IEEE INFOCOM 2009 and IEEE INFOCOM 2014, and the Best Paper Award at IEEE ICC 2008 and ICC 2017. He has co-founded Smartiply Inc., a fog computing startup company delivering boosted network connectivity and embedded artificial intelligence. He served as the editor-in-chief for *IEEE Transactions on Wireless Communications* during 2019–2022, and is a senior editor for *IEEE/ACM Transactions on Networking*. He was Technical Program Committee (TPC) co-chair for a number of major conferences in communication networks, including IEEE INFOCOM 2012 and ACM MOBIHOC 2015. He was the general chair for ACM/IEEE SEC 2017 and WiOPT 2016. He is a fellow of IEEE. He was a distinguished lecturer of the IEEE Communications Society.

Olav Tirkkonen received the MSc and PhD degrees in theoretical physics from Helsinki University of Technology, Helsinki, Finland in 1990 and 1994, respectively. He is currently a full professor of communication theory at Aalto University, Finland, where he has held a faculty position since 2006. After postdoctoral positions at The University of British Columbia (UBC), Vancouver, Canada, and Nordic Institute for Theoretical Physics (NORDITA), Copenhagen, Denmark, from 1999 to 2010, he was with Nokia Research Center, Helsinki, Finland. From 2016 to 2017, he was the visiting associate professor at Cornell University, Ithaca, NY, USA. He has authored or co-authored 300 papers and is the inventor of some 85 families of patents and patent applications which include 1% of all patents declared essential for the first standardized version of 4G LTE. His research interests include the coding for random access and quantization, quantum computation, and machine learning for cellular networks. He served as the general chair of 2022 IEEE International Symposium on Information Theory, and is a member of the Executive Editorial Committee of IEEE Transactions on Wireless Communications. He is a fellow of IEEE.