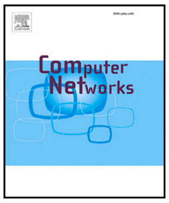
Computer Networks 209 (2022) 108876

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks

Manoj Kumar Somesula [a,*], Rashmi Ranjan Rout [a], D.V.L.N. Somayajulu [a,b]

[a] Department of Computer Science and Engineering, National Institute of Technology Warangal, Telangana, 506004, India
[b] Indian Institute of Information Technology Design and Manufacturing (IIITDM) Kurnool, Andhra Pradesh, 518002, India

ARTICLE INFO

Keywords:
Mobile edge networks
Cooperative caching
Multi-agent deep reinforcement learning
Partially observable Markov decision process
Multi-agent deep deterministic policy gradient

ABSTRACT

Caching the most likely to be requested content at the base stations in a cooperative manner can facilitate direct content delivery without fetching content from the remote content server and thus alleviate the user-perceived latency, reduce the burden on backhaul and minimize the duplicated content transmissions. Content popularity plays a vital role, and it drives caching on edge. In the literature, earlier works considered the content popularity either known earlier or obtained on prediction. However, the content popularity is time-varying and unknown in reality, so the above assumption makes it less practical. Therefore, this paper considers the cooperative cache replacement problem in a realistic scenario where the edge nodes are unaware of the content popularity in mobile edge networks. To address this problem, the main contribution of this paper is to design an intelligent content update mechanism using multi-agent deep reinforcement learning in dynamic environments. With the goal of maximizing the saved delay with deadline and capacity constraints, we formulate the cache replacement problem as Integer linear programming problem. Considering the dynamic nature of the content popularity, high dimensional parameters, and for an intelligent caching decision, we model the problem as a partially observable Markov decision process and present an efficient deep reinforcement learning algorithm by embedding the long short-term memory network (LSTM) into a multi-agent deep deterministic policy gradient formalism. The LSTM inclusion reduces the instability produced by partial observability of the environment. Extensive simulation results demonstrate that the proposed cooperative caching mechanism significantly improves the performance in terms of reward, acceleration ratio and hit ratio compared with existing mechanisms.

1. Introduction

In recent years, advancements in mobile devices and multimedia applications generate a huge volume of content that requires additional resources on the Internet and this leads to the exceptional growth of network traffic and imposes a massive load on the backhaul [1]. According to the Cisco survey, overall mobile data traffic anticipated to rise 7-fold from 2017 to 2022 [2]. To meet user demands and deal with the overwhelming traffic, network densification (deploying the base stations densely) is a fundamental technique in mobile edge networks [3]. However, the dense deployment of base stations does not solve the issue of traffic burden on the backhaul since a significant part of backhaul traffic is the duplicate downloads of some popular content [4]. Thus, mobile edge caching (MEC) is a prominent technique that utilizes the edge nodes as caching nodes to bring the contents near to the users, thus alleviating the core network burden and enhancing user Quality of Experience (QoE) [3]. In MEC, the edge nodes can serve a massive amount of duplicate content requests. It reduces the service delay and content delivery distance. Therefore, mobile edge caching is a promising technique for latency-critical mobile applications in a mobile edge computing framework.

The limited cache size at the base station restricts the mobile edge caching performance in the real world [5]. The caching performance can be improved by designing an efficient content placement mechanism using user preference and content popularity [6]. In the literature, earlier works presented proactive caching schemes by considering the content popularity which is known in advance [7] or can be predicted. The prediction requires user association, and user preferences which may vary in different contexts, such as personal information, topology, location, etc [8,9]. Practically, the content popularity is time-varying, so the above assumption (known in advance) makes it less practical. Futuristic content popularity information may not be available for taking the caching decisions. Hence, in this work we have designed

* Corresponding author.
E-mail addresses: smanoj@student.nitw.ac.in (M.K. Somesula), rashrr@nitw.ac.in (R.R. Rout), soma@nitw.ac.in (D.V.L.N. Somayajulu).

https://doi.org/10.1016/j.comnet.2022.108876
Received 10 August 2021; Received in revised form 22 December 2021; Accepted 28 February 2022
Available online 11 March 2022

List of Notations

$\mathcal{M}, \mathcal{F}, \mathcal{U}$	Set of base stations, contents and users
S_i	The cache capacity of ith MEC
S_f, dl_f	The size and deadline of fth content
$t \in T$	Time slot
$W_{i,u}^t$	Requests generated by user u in MEC i at t
$\rho_{i,u,f}^t$	Cumulative requests for content f in MEC i at t
$d_{i,u}, d_{j,u}, d_{h,u}$	Delay from local, neighbour and central server to user u
$x_{f,i}^t$	The content f cached in MEC i at time t
D^t	The expected saved delay
s_i^t, a_i^t, R_i^t	System state, action spaces and reward at MEC i in t
K_i^t, N_i^t, B_i^t	Set of user requests, cache state and deadline of MEC i
$\psi_{f,i}^{t,l}, \psi_{f,i}^{t,a}, \psi_{f,i}^{t,h}$	Low, average and high priority of f in MEC i at t
$c_i, c_{i,j}, c_{i,h}$	Cost of serving f from local, nearby and central server
$r_{i,l}^t, r_{i,j}^t, r_{i,h}^t$	l contents fetched from local, nearby and central server
$\alpha_i, \delta_{f_+,i}^t$	Cost of replacement and number of contents replaced at i
h_a^t, h_c^t	Actor and critic network historical information
$y_{i,j}^t$	Target network
$\emptyset, \theta$	Actor and critic network weight parameters

a cache replacement scheme to handle the dynamic content requests with out considering the content popularity information.

The effective cache utilization is reduced when the individual nodes with limited storage make independent decisions since they may redundantly cache popular content. A practical solution is to facilitate cooperation among edge nodes by sharing the content and which has been an important consideration in this work. Moreover, different edge nodes share their content in cooperative caching, which forms more extensive cache storage and enables cache diversity [10]. However, the inclusion of cooperation among edge nodes in caching decisions may invoke various technical problems. Some of the important problems that are addressed in this work are as follows: (a) to achieve cooperation, edge nodes should be aware of the caching states of the neighbouring nodes by sharing content, which causes considerable burdens [11], (b) efficient cooperation control needs an adaptive and dynamic framework, and (c) the designed caching mechanism has to tackle large-scale information induced by enormous data and information interaction. The conventional optimization mechanisms, such as dynamic programming and integer linear programming, can handle the first two problems [12]. Considering the dynamic nature of the content popularity, high dimensional parameters, and for an intelligent caching decision, the conventional optimization methods will not be suitable [12]. The recent success in Reinforcement Learning (RL) [13] and strong characteristic representation capability of Deep Reinforcement Learning (DRL) [9,14] motivate us to design a learning based caching mechanism to handle the changing nature and complex systems.

In this work, we aim to maximize the saved delay by considering the capacity and deadline constraints for accessing a large volume of data. We consider the content request deadline for generality and practicality, which is reasonable in latency-sensitive mobile and IoT applications. The novelty of this work lies in designing a cache mechanism for a dynamic environment in the absence of content popularity. Since each edge node observes its local state, the cooperative cache replacement problem is modelled as a Partially Observable Markov Decision Process (POMDP) [15]. The modelled multi-agent decision problem optimizes the latency of obtaining content from local MEC, neighbouring MEC and content server. To manage nodes to coordinate the caching decisions, we adopted and presented a multi-agent actor–critic framework based caching mechanism. The contributions of this work are as follows:

- Design an integer linear programming problem for cooperative content caching problem: maximization of saved download delay subject to constraints, namely cache capacity and deadline of the content in mobile edge networks.
- Modelling the formulated cooperative cache update problem as a POMDP based on a multi-agent decision problem to maximize the cumulative reward by ensuring the coordination of the MEC servers.
- Design a multi-agent recurrent deep reinforcement learning-based cooperative caching algorithm by devising the multi-agent actor–critic framework to solve the proposed problem.
- Extensive simulations have been performed to show the efficacy of the proposed multi-agent recurrent cooperative caching algorithm by considering acceleration ratio, hit ratio and caching reward.

The rest of this paper is organized as follows. The summary of related work is discussed in Section 2. The system model and problem formulation is presented in Section 3. In Section 4, the POMDP modelling is discussed. The multi-agent actor–critic framework and proposed caching scheme is presented in Section 5. The simulation results are discussed in Section 6. Concluding remarks are provided in Section 7.

2. Related work

Content caching has been studied widely in the literature. Practically users request a small number of frequently accessed top-ranked content. Li et al. [16] presented a survey on content placement and delivery mechanisms in various cellular networks. Content pre-fetching that depends on content popularity has been investigated in the literature. Shanmugam et al. [17] proposed a caching mechanism for each helper node to reduce the expected download delay. Poularakis et al. [18] presented an approximation approach for content cache placement in small cell network (SCN) using content popularity knowledge. Collaborative cache placement has been investigated in SCN to handle the limitation of cache capacity at an each node [10,12,19,20]. Sun et al. [19] have studied collaborative caching in multi-cell cooperative networks to minimize the average delay with limited cache storage. Jiang et al. [12] have designed a cooperative content caching and delivery scheme to minimize the average download delay.

The works mentioned above consider the content popularity known in advance. Moreover, popularity prediction based caching strategies were also studied in [8,21–23]. Bharath et al. [21] have proposed a transfer learning scheme to predict the popularity profile using the user request pattern for distributed heterogeneous cellular networks. Chen et al. [8] have presented an echo state network to estimate the content popularity and node mobility to maximize the user QoE in unmanned areal vehicle placement. In [23], authors proposed a learning theoretic perspective for content caching heterogeneous networks with time-varying and unknown popularity profiles. However, the works mentioned above consider the content popularity prediction and ignore the dynamic user requests and environment complexities. In contrast, our work considers reinforcement learning to handle the dynamic environment.

Typically, wireless caching has a high time-varying user requests. To address user requests' time-varying nature, BS with finite cache capacity replaces the content very often. Commonly used content replacement mechanisms are least frequently used (LFU), least recently used (LRU) and first-in first-out (FIFO) [24]. The traditional replacement models cannot capture the changing nature of content popularity because of the real environment's complexity. The conventional replacement mechanisms are suitable for single cache replacement. However, multiple cache replacement mechanisms require coordination among the nodes.

There are various single-agent learning algorithms like DQN, DDPG, and advantage actor–critic (A2C) in the literature [25]. A Q-learning based cache update mechanism presented to model the local and global content popularities as Markov chains in [26]. In [27], the authors presented the proactive caching mechanism based on policy gradient reinforcement learning schemes to minimize the long-term average energy cost by assuming the Poisson shot noise popularity dynamics. A DQN approach is presented in [28] to address the large continuous state–action space. In [29], the authors formulated the cache replacement problem as MDP and presented a caching policy based on the A3C (asynchronous advantage actor–critic) DRL mechanism. A DRL based framework with Wolpertinger architecture is presented in [14] for content caching in a single BS scenario and presented a deep deterministic policy gradient training mechanism for the actor–critic network. However, the works mentioned above consider the DRL mechanism for caching content, which is not a practical solution to the distributed environments where multiple agents are involved in decision making. In contrast, our work considers the multi-agent DRL to handle the distributed nature of the problem.

In reality, the environment is complex and there are several cases where the single-agent cannot deal effectively. In these cases, multiple agent systems are essential. In the multi agent scenario all the agents learn the policy by interacting within a common environment. Therefore, an agent must either compete or coordinate with other agents in the environment to obtain the good results. The cooperative cache network can improve performance [10]. Conventional single agent reinforcement learning mechanisms such as Q-learning or policy gradient is not applicable in multi agent reinforcement learning because as the training progresses each agent policy changes and environment becomes non stationary. To mimic the effect of various agents on the environment, MADDPG employs a centralized value function. Furthermore, every agent has a centralized critic, which may be utilized in various scenarios, including competition and cooperation, to define distinct reward functions for various agents.

There are some studies on caching in a multi-agent system. Jiang et al. [30] have formulated the content caching in D2D networks as multi-armed bandit problem and presented two Q-learning based multi agent learning mechanisms. Q-learning based multi agent learning mechanism maintains the Q-value in memory because the massive state–action space storage of individually BS may exhaust. Zhang et al. [31] have investigated a cooperative edge cache mechanism for IoT networks and designed a centralized MADRL scheme. Chen et al. [32] have discussed variational recurrent neural network and LSTM based cooperative caching mechanism by estimating users' content requests and content access. Wang et al. [33] have presented MacoCache utilizing the MADRL approach for the distributed multi-agent system. Jiang et al. [34] have studied the cache update scheme in a multiple agent system by formulating the replacement problem to improve content delivery. The works discussed above looked at the multi-agent DRL mechanism, which has difficulties transferring information among MECs and these mechanisms are either centralized or partially decentralized mechanisms. Our work considers a fully decentralized multi-agent DRL caching mechanism.

Zhong et al. [14] have presented a actor–critic framework with deep deterministic policy gradient learning scheme to reduce the overall delay. The content sharing is not considered by Zhong et al. and this leads to cache under utilization. A multi-agent RL mechanism is proposed in [35] to minimize the traffic congestion in the multi-intersection scenario. Huang et al. [36] have discussed a multi-user collaborative task execution problem with limited battery capacity, network and computing scenario and presented a multi-agent deep deterministic policy gradient algorithm. Song et al. [37] have investigated the joint content caching and content sharing in cooperative scenario and addressed the problem in view of multi-armed bandit learning by designing an ADMM. However, the works mentioned above have considered the multi-agent DRL mechanism where they suffer from exchanging information hugely between MECs. As the number of contents and MECs rises, leading to huge exchange overhead, we consider distributed cooperative caching with a recurrent multi-agent DRL mechanism in this work.

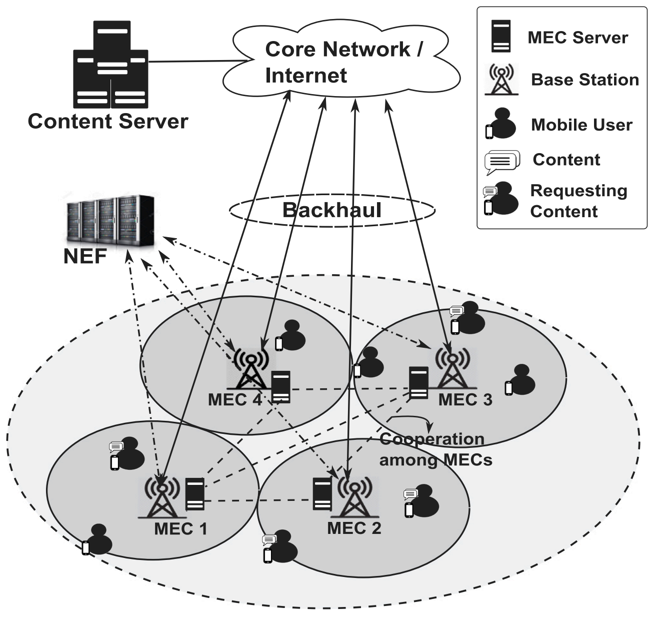


Fig. 1. Illustration of the proposed system model.

3. System model and problem formulation

In this section, the network model and problem formulation are presented in detail.

3.1. Network model

Mobile edge computing improves users' capabilities by providing cache capacity (i.e., storage), network resources and computing near to the users. Consider a mobile edge network containing a set $\mathcal{M}$ of M small base stations (BSs) equipped with a MEC server, a set $\mathcal{U}$ of U mobile users, a content server and a central coordinator NEF (Network Exposure Function) as shown in Fig. 1. Each MEC $i \in \mathcal{M}$ has a limited cache S_i called local storage. The storage of each MEC is used for content caching. The MECs are connected to each other and also to the core network through the backhaul link. The content server acts as an origin server that stores all contents. NEF (is a crucial network element in 5G networks) has a global view and it maintains the content cached at individual MECs and monitors users' content requests at each MEC [38]. A user directly connected to a base station and the user may be in the communication range of more than one BS at any point of time. However, any user can communicate with only one BS at a particular time. Mobile users are attached to the base stations according to a cellular network protocol. The connected base stations are accountable for serving user requests. Each BS receives content requests from multiple users in the communication range without knowing its popularities. The MEC can serve the requests in three ways: (1) local MEC, (2) neighbour MEC, and (3) content (central) server.

Consider a set $\mathcal{F}$ of F contents in the content library located in the content server. Each content f is determined with two features where S_f denotes the size of the content and dl_f denotes maximum allowed access latency to get content f. The time split into slots and each time slot is denoted by $t \in T$. We assume that the content requests are independent. The user can request only one content in time t and user location cannot change in any given time slot. The requests generated by user u at time t is represented by a binary vector $W^t_{i,u} = \{w^t_{i,u,1}, w^t_{i,u,2}, \dots, w^t_{i,u,F}\}$. $w^t_{i,u,f} = 1$ means the user u requests content f in MEC i at time t, $w^t_{i,u,f} = 0$ otherwise. The frequency of content is indicated as $\rho^t_{i,u} = \{\rho^t_{i,u,1}, \rho^t_{i,u,2}, \dots, \rho^t_{i,u,F}\}$, where $\rho^t_{i,u,f}$ denotes the cumulative requests for content f in MEC i at time t. We also assume that in each time slot, the MECs storage is filled with contents.

3.2. Problem formulation

The cooperative content replacement problem formed as an optimization problem to maximize the saved download delay. A binary indicator $X^t \in \{0,1\}$ denotes the cooperative cache replacement scheme in MEC, $x^t_{f,i} \in X^t$ is 1 if the content f is stored in cache of MEC i at time t, 0 otherwise. The download delay is a typical metric to evaluate the performance in mobile edge networks. First, we find the expected saved delay then formulate the maximizing the saved delay subjective to capacity and deadline constraints.

The delay for getting content f from MEC i to user u is denoted as $d_{i,u}$. The content requested by the user retrieved from the local storage of the corresponding MEC, then the delay is considered as 0. If the requested content is unavailable at the corresponding MEC i, then MEC i forwards the request to neighbouring MEC as per NEFs entry. The delay is considered as the number of hops between user u and MEC j (MEC j is the neighbouring node of MEC i) as $d_{j,u}$. If the requested content is unavailable at any of the MECs, the user fetches the content from the central server $d_{h,u}$ and $d_{i,u} < d_{j,u} < d_{h,u}, \forall j \neq i, j \in \mathcal{M}$.

Definition 3.1 (*Saved Delay*)**.** The difference in delay from the content server and MEC node is defined as the saved delay.

The saved delay depends on the frequency of cached content requests. The saved delay is split into two parts in the proposed model, intra MEC saved delay (local caching) and inter MEC saved delay (cooperative caching). Intra MEC saved delay is attained by locally cached contents.

$$D^t_{i,0} = \sum_{f=1}^{F}\sum_{u=1}^{U} w^t_{i,u,f} \cdot \rho^t_{i,u,f} \cdot S_f \cdot x^t_{f,i} \cdot (d_{h,u} - d_{i,u}) \tag{1}$$

Inter MEC saved delay is attained by the neighbouring MECs sharing the cached contents.

$$D^t_{i,1} = \sum_{f=1}^{F}\sum_{u=1}^{U} w^t_{i,u,f} \cdot \rho^t_{i,u,f} \cdot S_f \cdot (1 - x^t_{f,i}) \cdot z^t_i \cdot (d_{h,u} - d_{j,u}) \tag{2}$$

where $z^t_i = (1 - \prod_{j=1,j\neq i}^{M}(1 - x^t_{f,j}))$. The expected saved delay is

$$D^t = \sum_{i=1}^{M} D^t_{i,0} + D^t_{i,1} \tag{3}$$

We aim to maximize the saved delay by replacing the requested contents at each MEC subjective on deadline and capacity constraints. Hence, the problem is formulated as:

$$\text{Max } \frac{1}{T}\sum_{t=1}^{T} D^t \tag{4}$$

s. t.

$$\sum_{f=1}^{F} S_f \cdot x^t_{f,i} \leq S_i, \qquad \forall i \in \mathcal{M} \tag{5}$$

$$\sum_{f=1}^{F} x^t_{f,i} \leq 1, \qquad \forall i \in \mathcal{M} \tag{6}$$

$$\sum_{i=1}^{V} x^t_{f,i} \leq M, \qquad \forall f \in \mathcal{F}, 1 \leq i \leq M \tag{7}$$

$$D^t \leq dl_f, \qquad \forall f \in \mathcal{F}, \forall i \in \mathcal{M} \tag{8}$$

$$x^t_{f,i} \in \{0,1\}, \qquad \forall f \in \mathcal{F}, i \in \mathcal{M} \tag{9}$$

The objective (4) is the total saved delay of the overall network. Constraints (6) and (7) guarantee that the MEC node is not allowed to cache duplicate content. Constraint (5) provides the finite capacity of each BS. Constraint (8) is the deadline constraint, which ensures that the maximum allowable delay for the response to a request. Thus, the BS can satisfy the users' QoS requirements. Finally, constraint (9) is the non-negativity and integrality of the decision variables.

The cooperative content replacement problem presented in Eq. (4) is an integer linear programming (ILP) problem. We can show that the proposed problem is NP-hard by transforming the knapsack problem (known as NP-hard) into our problem. The problem presented in Eq. (4) can be addressed by finding the optimal decision variables $\{X^t\}$ in the present time slot. Nevertheless, the decision variable present in Eq. (4) is a binary variable and changing dynamically. Addressing the proposed problem requires to gather a huge quantity of network state information. Besides, we consider the practical scenario with an unknown content request pattern in advance. Because of the changing nature of the content popularity and to take an intelligent caching decision, we cannot adopt the conventional optimization methods [12]. The continued advancements and strong characteristic representation capabilities of Deep Learning (DL) [39] have encouraged learning in wireless networks. The learning based mechanism allows an end-to-end solution from predicting the content requests to cache decision. As a branch of AI, reinforcement learning is extensively adapted in several fields (self-driving, robot control, etc.) to solve decision optimization problems.

The content placement is determined mainly based on the present state and the caching decision, which does not depend on the previous states. Therefore, we can model the system states evolution by a Markov process. The MEC has its cache state and current request information while taking the caching decision at time t. In the multiple MEC scenario, each MEC takes the caching decision based on its local cache state and each MEC does not have the caching state information of other (neighbouring) MEC nodes. In reality, MEC cannot observe the complete system information regarding the caching states and content request distribution to take the cache decisions, which motivate us to represent the problem as a partially observable Markov decision process (POMDP) [15]. Then we develop a cooperative content replacement strategy using multi-agent reinforcement learning.

4. Multi-agent deep reinforcement learning model for cooperative caching

We consider that an agent can fully observe the cache state in the Markov Decision Problem (MDP). However, in a multi-agent environment, individual agent can observe its local state, which is partial information about the environment. Therefore, multi agent decision problems are modelled by POMDP. A POMDP is defined by a tuple $\{\mathcal{S}, \mathcal{A}, R, P, \Omega, \mathcal{O}\}$. $\mathcal{S}$ is a set of states, $\mathcal{A}$ set of actions, R is the reward function and P is the transition probability from state s to s', which is defined in the MDP model. Ω is the set of observations and $\mathcal{O}$ is the set observation probabilities. The observation, state, action and reward are defined as follows.

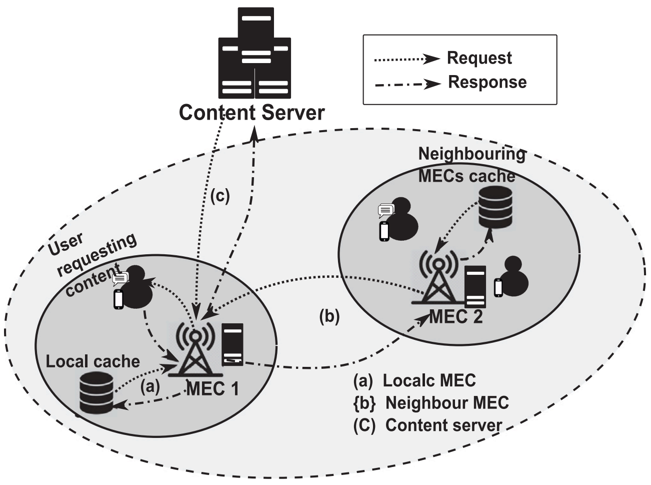


Fig. 2. Illustration of requests served by MEC.

4.0.1. Observation and state space ($\mathcal{S}$)

Let S is the set of system state space where $S = \{s_i | s_i = (N_i^t, K_i^t, B_i^t, \psi_i^t)\}$. In each time slot t, the state s_i^t contains the set of user requests K_i^t, MEC i cache state N_i^t, content delivery deadline B_i^t and priority of content ψ_i^t at MEC i. Where $K_i^t = \{k_{i,1}^t, k_{i,2}^t, \dots, k_{i,U}^t\}$, $k_{i,u}^t$ is the contents requested by user u at MEC i in time t, $B_i^t = \{b_{i,1}^t, b_{i,2}^t, \dots, b_{i,F}^t\}$, $b_{i,f}^t$ is the content delivery deadlines of MEC i for accessing the requested content f in time t. Since the content popularity is not available, the caching decisions are derived depend on the content already cached in MEC and the currently requested content. Therefore, we determine the priority of the content cached in MEC in time t, represented as $\psi_i^t = \{\psi_{f,i}^{t,l}, \psi_{f,i}^{t,a}, \psi_{f,i}^{t,h}\}$. Where $\psi_{f,i}^{t,l} = \sum_{t-\tau_l}^{t} \sum_{u=1}^{U} \rho_{i,u,f}^t$ is low priority, $\psi_{f,i}^{t,a} = \sum_{t-\tau_a}^{t} \sum_{u=1}^{U} \rho_{i,u,f}^t$ is average priority and $\psi_{f,i}^{t,h} = \sum_{t-\tau_h}^{t} \sum_{u=1}^{U} \rho_{i,u,f}^t$ is high priority. The system state space is denoted as

$$s_i^t = (N_i^t, K_i^t, B_i^t, \psi_i^t) \tag{10}$$

4.0.2. Action space ($\mathcal{A}$)

Let A is the set of actions. Each MEC determines whether to keep or replace the content. The challenge in the MADRL is that we need to replace multiple contents in each time slot. Since the system environment is multi-agent, each MEC serves multiple users. Therefore, different MECs get a different number of content requests.

If some user requests are missing from corresponding MEC and neighbouring MECs, then replace the missed content with appropriate content by fetching it from the content server. Otherwise (all the user requests miss), MEC replaces contents comprises of newly obtained content from the server with the suitable MECs and its cache. $A = \bigcup_{t\in T} a_i^t, \forall s_i \in S$ represents the action space analogous to the state space S. Upon receiving the content requests, MEC calculates each content's priority and determines whether the content to keep or replace based on priority by satisfying constraints (5) and (8).

4.0.3. Reward function (R^t)

Our work aims to maximize the saved transmission delay by obtaining the desired content at a low transmission delay within the fetching deadline. Each MEC node replaces the cached content at local storage cooperatively. In the multi-agent cooperative environment, based on the availability of the content, either neighbouring MECs or the central server, serves the local MECs requests. The MEC associated with the user called local MEC, the nearby MECs, is called neighbouring MECs (see Fig. 2).

1. Suppose the content requested by the user is available at the local MEC. The content can be delivered immediately with low latency (see case-(a) from Fig. 2). The cost of delivering content from local MEC is denoted as c_i. Let us consider that the MEC i fetches l number of contents from its local storage in time t is indicated as $r_{i,l}^t$. Therefore, the cost of the local MEC service is represented as $c_i r_{i,l}^t$.
2. Suppose some of the contents requested by the user are unavailable at the corresponding MEC i and the content is available at neighbour MEC j, then the content is served by j to the user via MEC i (see case-(b) from Fig. 2). The cost of fetching content from MEC j to MEC i is denoted as $c_{i,j}$. Assume l number of contents are fetched from j to i in time t is denoted as $r_{i,j}^t$. Therefore, the cost of neighbouring MEC service is represented as $\sum_{j\in\mathcal{M},j\neq i} c_{i,j} r_{i,j}^t$.
3. Suppose the content requested by the user is unavailable at any of the MECs, then the corresponding MEC obtains the content from the content server (see case-(c) from Fig. 2). Let us consider the cost to get the content from the content server to the user via MEC i denoted as $c_{i,h}$. Assume l number of contents are fetched from content server h to i in time t is denoted as $r_{i,h}^t$. Therefore, the cost of content server service is represented as $c_{i,h} r_{i,h}^t$.

The overall cost of the service in time t is represented as

$$c_i r_{i,l}^t + \sum_{j\in\mathcal{M},j\neq i} c_{i,j} r_{i,j}^t + c_{i,h} r_{i,h}^t \tag{11}$$

The content server serves the content which is missed at local MEC and neighbouring MEC. Hence, the MEC replaces the newly fetched content with less priority content. Therefore, the cost should contain the replacement cost along with the delivery cost. The cost of replacing content at MEC i is denoted as α_i. The number of contents replaced by MEC i at time t is indicated as $\delta_{f_+,i}^t = f_i - (x_{f,i}^t \cap x_{f,i}^{t-1})$ where $x_{f,i}^t$ indicates content f cached in MEC i in time t, $x_{f,i}^{t-1}$ indicates the content f cached in MEC i at time $t-1$ and f_i indicates the content requests at MEC i. Therefore, the replacement cost is defined as

$$\sum_{f\in\mathcal{F}} \alpha_i \delta_{f_+,i}^t \tag{12}$$

The total cost is represented as sum of (11) and (12). That is

$$c_i r_{i,l}^t + \sum_{j\in\mathcal{M},j\neq i} d_{i,j} r_{i,j}^t + c_{i,h} r_{i,h}^t + \sum_{f\in\mathcal{F}} \alpha_i \delta_{f_+,i}^t \tag{13}$$

We consider that each content should be satisfied within the specified deadline of the content. If the content does not get within the deadline, the penalty cost should be included in the reward. The penalty cost of the system is represented as $\rho_{i,f}^t b_{i,f}^t$, where $b_{i,f}^t$ is the deadline of content f in MEC i and $\rho_{i,f}^t$ is the content frequency.

The cost of utilizing the local MECs cache is higher than without the local cache. Therefore, we need to maximize the saved cost for an effective caching scheme. The reward function of MEC i is denoted as

$$R_i^t = (c_{i,h} - c_i) r_{i,l}^t + \sum_{j\in\mathcal{M},j\neq i} (c_{i,h} - c_{i,j}) r_{i,j}^t - \sum_{f\in\mathcal{F}} (\alpha_i \delta_{f_+,i}^t + \rho_{i,f}^t b_{i,f}^t) \tag{14}$$

Maximizing the reward is maximizing the cost of saved delay. The term $r_{i,j}^t$ depends on the local cache and neighbouring cache. The instant reward of the system is defined as

$$R^t = \sum_{i\in\mathcal{M}} R_i^t \tag{15}$$

In a multi-agent system, each MEC is considered as an agent. Based on the systems states, each agent determines its cache placement. We indicate $\pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ set of all caching strategies, $\pi : S \to A$ is a caching policy, which associates the current system state s to a permissible action a. The optimal caching policy π^* maximizes the long-term reward in the multi-agent system. To maximize the system's long-term reward, each agent needs to work cooperatively because the immediate and long-term rewards impact agent actions. Hence, the cooperative content replacement problem expressed to maximize the cumulative discounted reward. The value function $V^{\pi}(S)$: is defined as

$$\mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^t | s(0) = s, \pi\right] \tag{16}$$

where $0 \le \gamma < 1$ is the discount faction, γ decides the future reward's effectiveness to the present decision. Lower γ values give more weight to the immediate reward. We need to find the optimal caching policy π^* follows Bellman's functions

$$V^{\pi^*}(s) = R(s, \pi^*(s)) + \gamma \sum_{s' \in S} P_{s's} V^{\pi^*}(s') \tag{17}$$

where $P_{s's}$ is the state transition probability. Bellman's functions usually solved by either value or policy iteration methods. However, Bellman's function presented in Eq. (17) is challenging because of the following points.

1. The state transition probability $P_{s's}$ is not known in advance without any prior knowledge. It is difficult to estimate $P_{s's}$ in real environment.
2. The time complexity is high with the traditional value or policy iteration methods because of the vast state and action space, restricting practical cache systems' applicability.
3. Due to the cooperation among the MEC nodes, each MEC node should not cache the cached content at neighbouring MEC nodes. Each MEC can only know its local information and not aware of the full system states and actions of the other MECs.

Hence, to address these issues, we present a MADRL based cooperative caching mechanism. The following section presents a multi-agent DRL mechanism to handle the proposed caching problem.

5. Multi-agent recurrent DRL for cooperative content caching

In real world, the environment has challenging conditions for multi-agent system that demands the cooperation among agents, such as partial observable nature of agents and non-stationary nature. Hence, we come up with a multi-agent actor–critic framework inspired by the success of AI in recent times. Therefore, we present the multi-agent DRL framework for cooperative content replacement in MEN.

5.1. Multi-agent actor–critic framework

Usually, there are two approaches to develop caching decisions, namely decentralized and centralized. In the centralized caching mechanism, the centralized server determines the caching decisions depending on the caching states' global view and assign them to edge nodes. Each edge node is responsible for executing the caching decisions and data storage. In the decentralized caching mechanism, the caching decisions are determined by the edge nodes. Each edge node determines its caching decision based on other nodes' cache state information received from the central server in this mechanism. The central server is responsible for synchronization and cache state information interaction. However, both approaches induce new problems. A centralized mechanism causes additional delay since the central server determines the cache decisions. Decentralized mechanism suffers from the cache state information exchanging problem, and it has a severe effect on cooperation among the edge nodes.

We propose cooperative caching by adopting centralized training with distributed execution framework to alleviate the problems mentioned earlier, providing the policies to utilize additional knowledge to simplify the training. In general, Q-learning cannot carry various information at the training and testing phase; therefore, making additional assumptions about the environment's structure is unnatural. Hence, we propose a Multi-Agent Recurrent Deep Deterministic Policy Gradient (MARDDPG) algorithm to decide whether to evict or retain the requested content inspired by [40,41]. The MARDDPG is a simple addition of the DDPG mechanism for a multi-agent system, where the actor can only obtain its local information and the critic augmented with additional information about other agents' policies. An individual agent is unaware of the other agents' policies in a multi-agent system, leading to a non-stationary problem. Each agent estimates the other agent policies by leveraging the actions and global state through the training phase to avoid the environment from the non-stationary problem. Hence, each agent attains the global optimal strategy by altering its policy depending on the other agents' estimated policy. Each agent consists of its own actor and critic networks in the proposed mechanism considering agent independently learns a distinct policy because of different locations (see Fig. 3).

Actor network: The actor-network described as a function that learns a caching policy $\pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ that maps the state to a permissible action taken from the action space $\mathcal{A}$. The state comprises the global state g and local state s_i observed by agent i. The agent i chooses an action a_i^t depends on its state s_i^t and policy $\pi_i^{\emptyset_i}$ throughout the testing phase without critic.

$$a_i^t = \pi_i^{\emptyset_i}(s_i^t) \tag{18}$$

Critic network: The critic network adopted for approximating the action-value function $V(S)$ provides the overall reward while taking action a_i^t based on the state s_i^t and the global state g at time t in the training phase. Each agent executes the actions in the environment and sends the present state information s_i^t and response from the environment to the critic network after the actor-network in time t chooses the actions $a^t = \{a_1^t, a_2^t, \dots, a_M^t\}$. The feedback consists of the next time instant state information s^{t+1} and reward R^t. Hence, Q-function defined as $Q_i^{\theta_i}(s_1^t, s_2^t, \dots, s_M^t, a_1^t, a_2^t, \dots, a_M^t)$ for each agent, which solves the problem caused by a not-stationary environment. Let us consider M agents with policy of agent i is π_i, then

$$\begin{aligned} P(s_i^{t+1}|s_i^t; Env) =& P(s_i^{t+1}|s_i^t; s_1^t, s_2^t, \dots, s_M^t, a_1^t, a_2^t, \dots, a_M^t, \\ & \pi_1, \pi_2, \dots, \pi_M) \\ =& P(s_i^{t+1}|s_i^t; s_1^t, s_2^t, \dots, s_M^t, a_1^t, a_2^t, \dots, a_M^t, \\ & \pi_1', \pi_2', \dots, \pi_M') \end{aligned} \tag{19}$$

By minimizing the loss function $\mathcal{L}(\theta_i)$, each critic updates its network and the loss function $\mathcal{L}(\theta_i)$ parameterized by $\theta = \{\theta_1, \theta_2, \dots, \theta_M\}$ defined as

$$\mathcal{L}(\theta_i) = \mathrm{E}_{s^t, a^t}\left[\left(Q_i^{\theta_i}(s^t, a^t) - y^t\right)^2\right] \tag{20}$$

where $s^t = \{s_1^t, s_2^t, \dots, s_M^t\}$, $a^t = \{a_1^t, a_2^t, \dots, a_M^t\}$ and

$$y^t = R^t + \gamma Q_i^{\theta_i}(s^{t+1}, a^t)|_{a_i^t = \pi_i^{\emptyset_i'}(s_i^{t+1})} \tag{21}$$

here $0 \le \gamma < 1$ is the discount factor.

Update: Individual agent maximizes the reward by optimizing the policy directly where the policy parameterized by $\emptyset$. Therefore, the objective is to maximize the cumulative reward function.

$$J(\emptyset_i) = \mathrm{E}_{s^t, a^t}\left[Q_i^{\theta_i}(s^t, a^t)|_{a_{s_i}^t = \pi_i^{\emptyset_i}(i)}\right] \tag{22}$$

The MADDPG can tackle cooperative tasks, but it cannot manage partially observable settings or history-dependent decision making because it does not represent explicit communication among agents. Since the edge nodes cannot communicate all the time, each node needs to track the previous interaction (how the actions of an agent influence the communication cost over the period and when the previous message is transmitted) obtained from all edge nodes. Recurrency serves as an effective tool for accomplishing the previous interactions. As a result, a good model design must take advantage of the temporal sequence of activity. Therefore, this mechanism builds on multi-agent actor–critic architecture with LSTM, allowing to remember the last communication of agents.

In the MARDDPG algorithm, the recurrent neural network LSTM added to the actor-network and critic network. Since the agents cannot

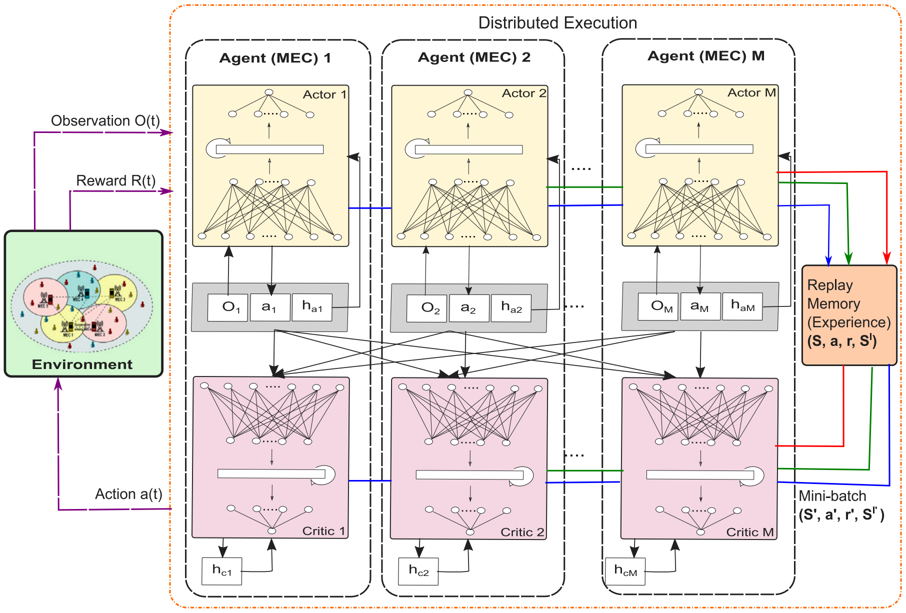


Fig. 3. Multi-agent recurrent DRL framework for cooperative caching. Here O_i, a_i represents the observation and actions of agent i and h_a, h_c represents the history of actor and critic.

communicate, the model takes a single frame in each time slot. Adding the LSTM enables a way to remember the last communication (the effect of the actions on reward) received from other agents. The actor-network and critic network historical information denoted by h_a^t and h_c^t. The individual agent chooses the action based on previous state h_i^t, i.e., $a_i^t = \pi_i^{\emptyset_i}(h_i^t)$, then the Q-function turn into $Q_i^{\theta_i}(h_c^t, a^t)$ where $h_c^t = \{h_{c,1}^t, h_{c,2}^t, \dots, h_{c,M}^t\}$. Likewise, loss function $\mathcal{L}(\theta_i)$ in the critic network is

$$\mathcal{L}(\theta_i) = \mathrm{E}_{h_c^t, a^t}\left[\left(Q_i^{\theta_i}(h_c^t, a^t) - y_i^t\right)^2\right] \tag{23}$$

where

$$y_i^t = r_i^t + \gamma Q_i^{\theta_i'}\left(h_c^{t+1}, \pi_1^{\emptyset_1'}(h_{a,1}^{t+1}), \dots, \pi_M^{\emptyset_M'}(h_{a,M}^{t+1})\right) \tag{24}$$

The objective function is denoted as

$$J(\emptyset_i) = \mathrm{E}_{h_c^t, a^t}\left[Q_i^{\theta_i}\left(h_c^{t+1}, \pi_1^{\emptyset_1}(h_{a,1}^{t+1}), \dots, \pi_M^{\emptyset_M}(h_{a,M}^{t+1})\right) \Big|_{a_i = \pi_i^{\emptyset_i}(h_{a,i}^t)} \right] \tag{25}$$

The replay buffer stores the experience information in the training phase. The critic and actor networks updated by randomly sampled episodes from the replay buffer in each training step. The target critic and actor network parameters are denoted by θ' and $\emptyset'$ respectively. The target network is updated using soft updates.

5.2. Multi-agent recurrent DRL based cooperative caching algorithm

The MEC node receives the user requests and obtains their features. It supplies the current request and caching state to the actor-network to get the caching actions. Following the action executing based on the policy, each agent receives the reward and the next state. Store the information received from the environment as history using LSTM.

$$\begin{aligned} h_a^{t+1} &= \mathrm{LSTM}(h_a^t, s^{t+1}) \\ h_c^{t+1} &= \mathrm{LSTM}(h_c^t, s^{t+1}, a^t) \end{aligned} \tag{26}$$

Then actor and critic network stores the experience in replay memory. To train the critic and actor-network, randomly select a mini-batch of the transitions from replay memory. For an individual sample, set the target critic network

$$y_{i,j}^t = r_{i,j}^t + \gamma Q_i^{\theta_i'}\left(h_{c,1}^{t+1,j}, \dots, h_{c,M}^{t+1,j}, \pi_1^{\emptyset_1'}\left(h_{a,1}^{t+1,j}\right), \dots, \pi_N^{\emptyset_N'}\left(h_{a,N}^{t+1,j}\right)\right) \tag{27}$$

and updates its parameter θ by reducing the loss function over mini batch

$$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_{j\in S}\left(Q_i^{\theta_i}\left(h_{c,1}^{t,j}, \dots, h_{c,M}^{t,j}, a_{1,j}^t, \dots, a_{M,j}^t\right) - y_{i,j}^t\right)^2 \tag{28}$$

Furthermore, the actor computes the policy gradient leveraging the loss function and the parameter $\emptyset$ updated using the gradient over mini batch.

$$\nabla_{\emptyset_i} J(\emptyset_i) \approx \frac{1}{S}\sum_{j\in S} \nabla_{\emptyset_{i,\pi_i^{\emptyset_i}}}\left(h_{a,i}^{t,j}\right), \nabla_{a_i} Q_i^{\theta_i}\left(h_{c,1}^{t,j}, \dots, h_{c,M}^{t,j}, \pi_1^{\emptyset_1}\left(h_{a,1}^{t,j}\right), \dots, \pi_N^{\emptyset_N}\left(h_{a,N}^{t,j}\right)\right) \tag{29}$$

Update the critic and actor parameters of the target network

$$\begin{aligned} \theta_i' &\leftarrow \tau\theta_i + (1-\tau)\theta' \\ \emptyset_i' &\leftarrow \tau\emptyset_i + (1-\tau)\emptyset' \end{aligned} \tag{30}$$

Algorithm 1 summarizes the cooperative content caching mechanism based on MARDDPG. First, randomly initialize the critic network parameter θ and actor-network parameter $\emptyset$. Initialize the replay memory G and the target network with weights θ' and $\emptyset'$. In each episode, initialize the empty history $h_{a,i}^t, h_{c,i}^t$ and a random process for exploration. Lines 5 to 13 shows that MEC receives the requests and observe the state. The individual agent selects an action a^t depend on the policy $\pi_i^{\emptyset_i}(h_{a,i}^t)$. After performing an action a^t, the agent gets the following state s^{t+1} information, and the reward R^t then stores the information collected from the environment in history using the LSTM network. Save the individual agent's experiences $(s_i^t, a_i^t, r_i^t | t = \{1, \dots, T\})$ in the replay memory G to train actor and critic networks. Line 14 shows that each agent randomly samples a mini batch of S transitions $\{s_{i,j}^1, a_{i,j}^1, r_{i,j}^1, s_{i,j}^2, a_{i,j}^2, r_{i,j}^2, \dots\}$ from the replay memory G to train the critic and actor-network. Lines 15 to 19 show that for each agent, the critic network estimates the Q-approximation for each sample $j \in S$, then compute the temporal difference-error and update its weights by minimizing the loss function $\mathcal{L}(\theta_i)$ over the target network. Further, the actor network computes the policy gradient $\nabla_{\emptyset_i} J(\emptyset_i)$ leveraging the loss function and update its weights by the average policy gradient over the target network. Then update the target network weights, content properties and cache state in lines 21 and 22.

Algorithm 1 MARDDPG based Content Caching Algorithm

```
Initialize the actor network $\pi_i^{\emptyset_i}(h_{a,i}^t)$ with random weights $\emptyset$ and the critic network $Q_i^{\theta_i}(h_c^t, a^t)|h_c^t = \{h_{c,1}^t, h_{c,2}^t, \dots, h_{c,M}^t\}, a^t = \{a_1^t, a_2^t, \dots, a_M^t\}$ with random weights $\theta$
Initialize the target network $\pi_i^{\emptyset_i'}$ $Q_i^{\theta_i'}$ with weights $\emptyset'$ and $\theta'$
Initialize the replay memory $G$ of each agent to capacity $F$
Output: $X$: Content Placement.
 1: for all episode do
 2:     Initialize t = 1;
 3:     Initialize a random process $\mathcal{M}$ for action exploration;
 4:     Initialize empty history $h_{a,i}^0, h_{c,i}^0$;
 5:     for $t \in T$ and $o^t \neq$ terminal do
 6:         The MEC receives user requests $W^t$;
 7:         Observe the cache state $s_i^t$ of each agent $i$;
 8:         For each agent $i$ select an action $a_i^t = \pi_i^{\emptyset_i}(h_{a,i}^t)$ with respect to current policy and exploration noise;
 9:         Execute action $a^t$, store the received reward $r^t$ and new state $s^{t+1}$ information in history using Eq. (26);
10:         $t = t + 1$;
11:     end for
12:     Store episode $(s_i^t, a_i^t, r_i^t | t = \{1, \dots, T\})$ for all agents;
13:     for all $i \in \mathcal{M}$ do
14:         Randomly sample a mini batch of $S$ episodes from replay memory $\{s_{i,j}^1, a_{i,j}^1, r_{i,j}^1, s_{i,j}^2, a_{i,j}^2, r_{i,j}^2, \dots\}$ episodes from replay memory $G$;
15:         for $t = T to 1$ do
16:             Set target network Eq. (27);
17:             Minimizing the loss using Eq. (28) and update the critic network;
18:             Update the actor network policy using the sampled policy gradient Eq. (29);
19:         end for
20:     end for
21:     Update the target networks using Eq. (30);
22:     Update the content properties $\psi_i$ and cache state $s_i$ ;
23: end for
```

There are primarily two methodologies for making caching decisions: decentralized and centralized. In the centralized caching method, the central node must coordinate edge nodes by consolidating local information and executing actions in a distributed fashion at each edge node individually. Here, the communication overhead is described as the total dimension of variables exchanged between the central and edge nodes. The central node needs the state $(N_i^t, K_i^t, B_i^t, \psi_i^t)$ information in each epoch. The dimension of the cache state N_i^t can be NULL since the cache allocation information is available at the central node. The user requests can be $\mathcal{O}(MU)$. The content deadline and priority can be $\mathcal{O}(2F)$. Thereafter, the central node performs the action on the environment, the dimension of variables involved is given as $\mathcal{O}(MF)$. Therefore, the total communication overhead in the training state is $\mathcal{O}(MU + 2F + MF)$. Similarly, the communication overhead for the evaluation stage is $\mathcal{O}(MU+2F+MF)$. Whereas in the proposed decentralized caching methodology, the exchange of information between the central node and edge nodes are not necessary. That is, each edge node acts as its own decision-maker, training caching strategies depending on local observation and global information (i.e., other agents actions and observations) without the need for collaboration with the central node. Each agent stores the experience in replay memory in the training stage and utilizes the stored experience in the evaluation stage. The training stage communication overhead can be $\mathcal{O}(ME)$, where E is the dimension of experience. Moreover, the L number of mini-batch of experiences are used to train the actor and critic. Therefore, the total communication overhead in the evaluation stage is $\mathcal{O}(LE)$.

6. Performance evaluation

In this section, we validate the performance of the proposed MARDDPG based cooperative caching mechanism using simulations. Particularly, first, we elaborate the simulation environment, performance metrics and reference algorithms. Furthermore, the performance of the MARDDPG mechanism is compared to the reference methods in terms of system parameters, and the simulation results are analysed in detail.

We use the real-world Dataset MovieLens 1M Dataset [42] in our simulations to investigate for requesting content. The MovieLens dataset consists of 3952 movies, 1 000 209 user ratings that take integer values [1 (worst), 5 (best)] and 6040 users. Each row of the dataset consists of userid, movieid, rating and timestamp. The rating information is considered the content request since the user rates a movie followed by watching it [43]. We consider the rating information as the frequency of movie requested by a user. We also assume that the number of requests for a movie within 10, 100 and 1000 requests as the features. Therefore, we select the top 600 popular content requested by users and the 100 most active users to analyse the user request statistics. More than 90% of the ratings are from the first year in the dataset, so we consider only the first year ratings for simulation. We obtain the skewness parameter $\alpha = 0.8$ by fitting the actual data from the dataset with the Zipf distribution as shown in Fig. 4(a).

To evaluate the performance of the proposed caching algorithm, we present the system setting in this section. Consider a square region with an area of 500 m × 500 m. In the given simulation area, we consider the Poisson point process (PPP) for base stations. The users are distributed within each base station coverage based on PPP shown in Fig. 4(b). We take 600 contents with size determined uniformly at random from the range of [10 to 100 MB], 15 BSs and 90 users. Each content request is associated with a deadline. That is, each request must be served within the deadline. We have considered the deadline in terms of seconds similar to [44] in this work. Therefore, the content deadline was assigned randomly from a uniform distribution range [10–40 ms] [44,45]. Content accessibility varies geographically because users are arbitrarily scattered across the BSs' coverage area. So, the delay in fetching the content from different BSs varies. The end-to-end latency to fetch the content from the local base station to the user and neighbouring BS to the user are randomly assigned from the uniform distribution ranges [10–20 ms] and [25–40 ms], respectively [46]. Similarly, the latency to fetch content from the content server to the user is 100 ms [46,47].

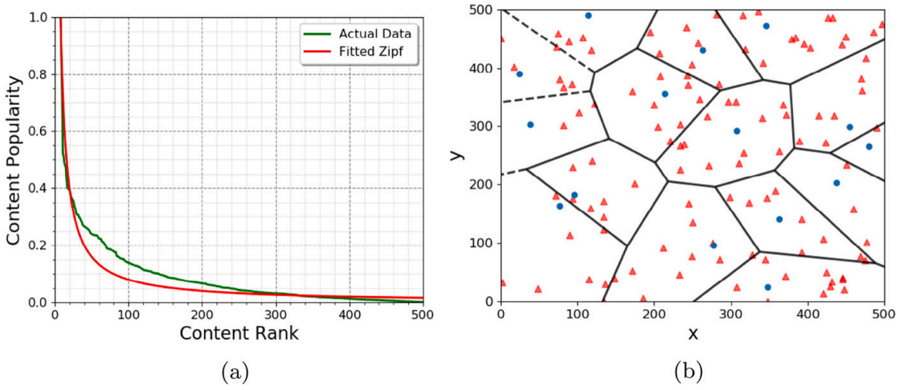


Fig. 4. (a) Comparison of content popularity of Movielens dataset vs. content rank (b) Voronoi cell diagram with size 500 m × 500 m where blue circle indicates the BSs and red triangles are mobile users.

Table 1
Simulation parameters.

Parameters	Values
Simulation area	500/m × 500/m
Number of users	90
Number of contents	600
Number of base stations	15
Content size	(10, 100] MB
The delay between BS and user	[10,20] ms
The delay between BSs	[25,40] ms
The delay between content server and BS	100 ms
The deadline of the content	[10,40] ms
Actor and critic learning rate	0.001, 0.0005
Network update rate	0.01
Discount	0.9
Mini batch size	256
Replay memory capacity	10^5
Number of episodes	1500
Number of steps in each episode	100

We consider Python with the TensorFlow platform for implementing the proposed MARDDPG caching mechanism and implemented it on the open-source R-MADDPG package. The neural network model composes the evaluated actor and critic network and the target actor and critic network for each agent. The evaluated critic and actor networks are similar to that of the target networks. The networks have three hidden layers with 64 neurons in each layer. The middle layer is an LSTM layer, and the other two layers are fully connected, where the first layer has a ReLU activation function. The target network is updated using the Adam optimizer, where the critic and actor networks learning rates are 0.001 and 0.0005, respectively, and the discount reward is 0.9. We consider the capacity of the replay memory as 10^5, and the mini-batch size is 256 (see Table 1).

6.1. Performance metrics

To compare the performance of cache replacement schemes, we consider the following metrics:

1. *Cache Hit Ratio*: The fraction of requests served over the total requests.
2. *Acceleration ratio*: The fraction of saved delay and overall delay (from the controller).
3. *Caching Reward*: The reward measures the cumulative long-term reward collected from caching (i.e., Sum of the intermediate reward of all MECs) using Eq. (15).
4. *Local Hit*: The fraction of requests served within the MEC.
5. *Neighbouring Hit*: The fraction of requests served within the network and not within the MEC.

6.2. Reference algorithms

In this section, we compare the proposed algorithms with the following caching mechanisms: Least Recently Used (LRU) [48], First In First Out (FIFO) [49], Least Frequently Used (LFU) [50], Multi-agent actor–critic (MAC) [14] and Deep Reinforcement Learning (DRL) [9].

1. LRU: It is a recency based mechanism where the least recently requested file is updated with the fetched file when the cache is already full.
2. LFU: It is a frequency-based mechanism where the least number of times requested file is updated with the fetched file when the cache is already full.
3. FIFO: It is an arrival based mechanism where the earliest cached file is updated with the fetched file when the cache is already full.
4. DRL: It is a cache replacement decision mechanism where individual MEC performs the caching decisions individually with the help of local observations without considering the impact of other MECs.
5. MAC: It is a cache replacement decision mechanism where the actor takes caching decisions and critic evaluates the policy. In this mechanism, communication between the agents is not considered; hence, there is no global state to process for actor networks.

The first three cache replacement (benchmark mechanisms) strategies update the content individually based on arrival, frequency and recency, whereas the other strategies consider deep reinforcement learning to place the contents. These benchmark caching mechanisms have not considered the learning mechanism, which shows how the proposed mechanism benefits from the learning. The fourth cache replacement strategy (DRL) is different from the fifth strategy (MAC) because, in DRL, the individual node is not aware of the other nodes' information, and the former is the value-based RL, and the latter is policy-based RL. DRL is a single agent learning-based caching scheme, which shows how the learning-based caching schemes improve caching performance. MAC is a partially decentralized cooperative caching scheme with a centralized critic is employed to train independent actors. This mechanism has been used to compare the proposed mechanism to show the difference between the fully distributed and partially distributed cooperative caching mechanisms. Moreover, we consider the variant (i.e., MADDPG [40]) of the proposed strategy without the recurrence model (LSTM) and uses a similar network setup. MADDPG is considered to demonstrate the efficacy of the inclusion of recurrency.

6.3. Impact of cache size

The impact of cache size on acceleration ratio and cache hit ratio is shown in Fig. 5. In this simulations, the number of MECs is 15, skewness

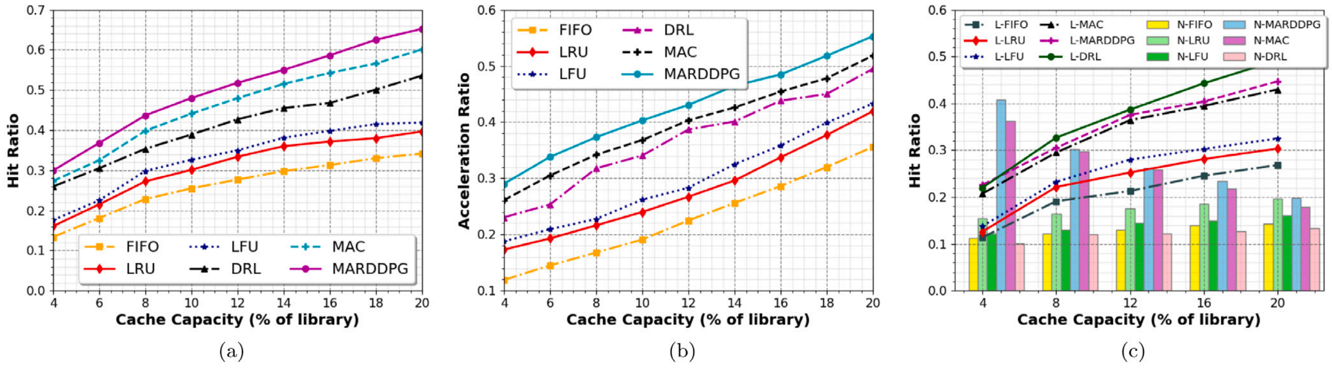


Fig. 5. Comparison of caching schemes using cache capacity vs. (a) cache hit ratio (b) acceleration ratio (c) local and neighbour cache hit ratio.

parameter is 0.8 and the cache capacity varies from 4% to 20% total library size with step size 2.

In Fig. 5(a), the impact of cache size on the cache hit ratio is presented. The curves indicate an upward trend with the rise in cache size since the large cache size allows MECs to cache more content, allowing them to satisfy more user requests from local or neighbouring MECs. The learning-based mechanisms show superiority over conventional rule-based replacement mechanisms since the learning-based mechanisms capture the user request features from the historical data. MARDDPG has better performance than MAC and DRL. The DRL mechanism does not consider the cooperation among the nodes where each node tries to maximize its reward without concern about other nodes. MAC considers no cooperation between agents even though it considers the multi-agent framework. The proposed MARDDPG mechanism provide improvement of up to 24, 19, 17, 9 and 4% on hit ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

In Fig. 5(b), the impact of cache size on the acceleration ratio is presented. The trend of the curves indicates that the growing capacity allows more content at the edge nodes. This shows that the growing cache capacity improves user satisfaction by caching content with less delay. We can notice that the proposed mechanism is steady compared to the DRL because of the cooperation among the nodes. We can also notice that the learning-based schemes have a clear upper hand over benchmark mechanisms. The reason is that learning allows replacing the more appropriate content by finding the user request format. The proposed MARDDPG mechanism provide improvement of up to 20, 15, 13, 6 and 3% on acceleration ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

In Fig. 5(c), the impact of the cache size on the local and neighbouring cache hit is presented. The local hit rate is denoted with ‘L’ and the neighbouring hit rate denoted with ‘N’. The local hit ratio of the learning-based algorithms has superiority over rule-based mechanisms. The reason is that the learning-based mechanisms perform the cache replacement decision depending on the history of the data that enables the nodes to cache more popular content locally to the MEC and moderately popular content cached at neighbouring nodes. The upward trend indicates that the rise in cache size improves the local hit rate. We can notice that the DRL has higher local hit ratio than MAC and MARDDPG and lower neighbour hit ratio because each agent in DRL cache based on local cache information leading to redundant content at each agent. We can also notice that MAC and the proposed mechanisms have more neighbouring hit rate with less capacity and decreases as the capacity increases. The reason is that both the MAC and proposed mechanisms considers the cooperation among the agents leading to higher neighbour hit ratio. Overall, both MAC and MARDDPG have a better cache hit ratio than DRL. Further, we can observe that the proposed mechanism may sacrifice the local hit rate, but it satisfies more user requests with a small delay than all other baseline algorithms.

6.4. *Impact of number of MECs*

We show the impact of number of MECs on cache hit ratio and acceleration ratio in Fig. 6. In this simulations, the cache capacity is 10%, skewness parameter is 0.8 and the number of MECs varies from 5 to 15 with step size 2.

We can see the effect of the cache hit ratio with a varying number of MECs in Fig. 6(a). We can notice that the proposed mechanism shows clear superiority over other baseline mechanisms since it uses the cooperative mechanism and learning the user request pattern from the history data. The conventional rule-based mechanism is performing less than the learning-based mechanisms. The proposed MARDDPG mechanism provide improvement of up to 22.5, 18, 15, 6 and 4.5% on hit ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

From Fig. 6(b), the acceleration ratio grows slowly with fewer MECs, and rapidly increases as MECs increases. The proposed mechanism outperforms other mechanisms because the cooperation makes the space for more content. We notice that the LFU shows relatively better performance than the other baseline schemes by caching the most accessed content at edge nodes. We can also notice that with the smaller number of MECs, the proposed scheme and MAC have identical performance, whereas the increase in the number of MECs makes the proposed mechanism grow slowly. The proposed MARDDPG mechanism provide improvement of up to 19, 13, 10, 6 and 1% on acceleration ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

6.5. *Impact of number of contents*

The impact of contents on cache hit ratio and acceleration ratio is shown in Fig. 7. In this simulations, the number of MECs is 15, skewness parameter is 0.8, cache capacity is 10% total library size and number of contents varies from 0.2 to 1.0 with step size 0.2.

In Fig. 7(a), the impact of number of contents on the cache hit ratio is presented. The bars indicate an upward trend as contents rise. More popular content need to be cached at MECs, leading to frequent cache replacement due to limited cache capacity. We can see that MARDDPG has better performance than other reference algorithms. The DRL mechanism does not consider the cooperation among the nodes where each node tries to maximize its reward without concern about other nodes leading to less cache hit ratio than MAC and MARDDPG. MAC considers no cooperation between agents even though it considers the multi-agent framework. The proposed MARDDPG mechanism provide improvement of up to 20, 17, 17, 11 and 2.2 per cent on hit ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

We can notice from Fig. 7(b) that the conventional replacement mechanism grows slowly compared to the learning-based mechanisms. Because the learning-based mechanism captures the content popularity, enabling the MEC to replace the less popular content, leading to more saved delay. The proposed mechanism outperforms other algorithms.

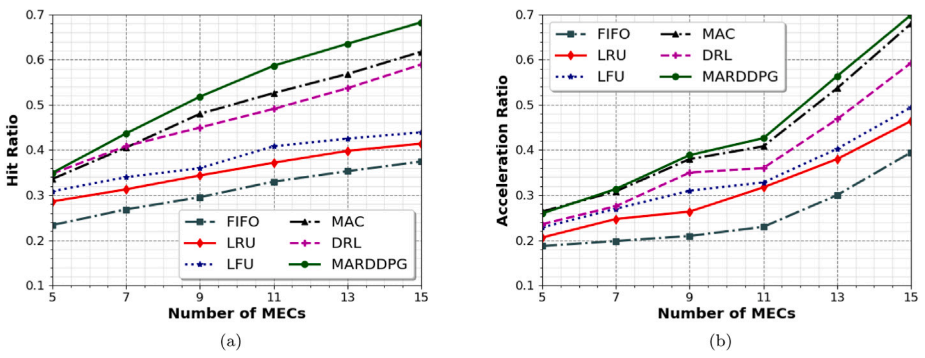


Fig. 6. Comparison of caching schemes using number of MECs vs. (a) cache hit ratio (b) acceleration ratio.

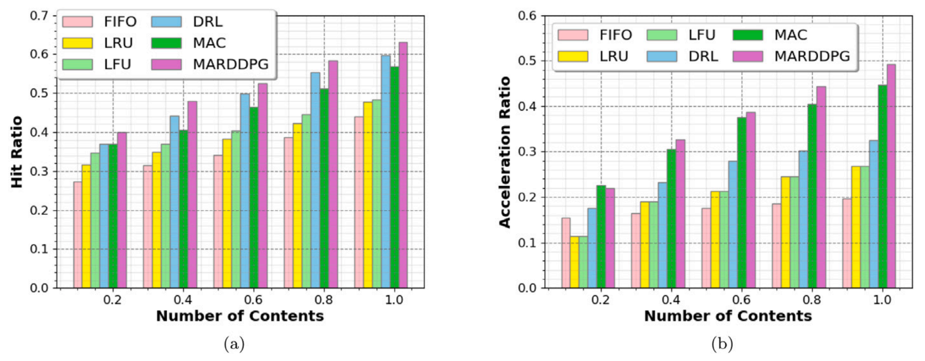


Fig. 7. Comparison of caching schemes using number of contents vs. (a) cache hit ratio (b) acceleration ratio.

We notice that the acceleration ratio is low with fewer contents and grows as the number of contents increases. The reason is that when the number of contents is increasing allows caching an additional number of frequently accessed content. The proposed MARDDPG mechanism provide improvement of up to 17, 13, 11, 6 and 3% on acceleration ratio compared with FIFO, LRU, LFU, DRL and MAC, respectively.

6.6. *Impact of Zipf parameter*

The impact of Zipf parameter on cache hit ratio and accelerated ratio is presented in Figs. 8(a) and 8(b) respectively. In this section, to generalize the findings, we use an analytical content-request model in which Zipf distribution is considered as the file popularity distribution. The skewness parameter is varied for different measures from 0.2 to 1.0 with step size 0.2. To produce the synthesized content requests, we utilize a collection of 10,000 contents. We consider the number of MECs to be 15, and cache capacity is 10% total library size.

From Fig. 8(a), we can notice that the upward trend of all the algorithms indicates that the more requests are for few contents as the Zipf skewness parameter rises; this leads to an increase in the hit ratio. The reason is that the increasing skewness parameter makes the frequently accessed content increase. From Fig. 8(b), we can see that the increase in the skewness parameter allows access to the requested content within a smaller delay improves the acceleration ratio.

6.7. *Performance evaluation with training episode*

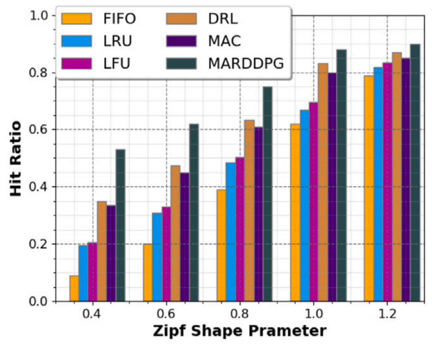
In Fig. 9, the performance comparison of cache hit ratio, local and neighbour cache hit ratio is presented. In this simulations, the number of MECs are 15, skewness parameter is 0.8, cache capacity is 10%.

From Fig. 9(a), we can see that the rule-based cache replacement mechanism shows a relatively stable hit ratio since they have not considered real-time continuous learning from the environment. The learning-based algorithms curves indicate an upward trend and stabilize after that. The DRL has a higher hit ratio than MAC and MARDDPG initially, but as the episodes increase, it slowly diminishes. That is because the DRL is a non-cooperative cache replacement mechanism where each agent performs the cache replacement based on local information, not considering the other agents' information in caching decision. Therefore, each agent may cache content redundantly leads to obtain more content from the content server. In MAC, the agents cache the content based on the central controller, which cooperates with communication overhead. The proposed MARDDPG outperforms the other mechanisms since it uses the LSTM to learn the better policy to cache more popular content.

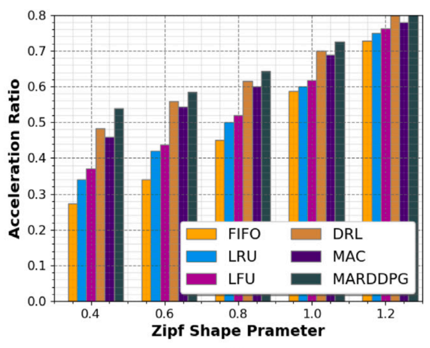
Figs. 9(b) and 9(c) show that the rule-based mechanisms have more local hit ratios and fewer neighbour-hit ratios. The reason is that the rule-based mechanism not consider the learning from the environment; therefore, each MEC caches the redundant content leads to fetch and replace more content compared to other mechanisms. We can also notice that the DRL mechanism has a lower neighbour hit ratio and higher local cache hit ratio since each agent in DRL caches based on the local cache state. The MAC and MARDDPG have lower local hit rate than DRL, but both mechanisms have better neighbour hit ratio. Overall, both the MAC and MARDDPG have a better cache hit ratio than DRL, even though they sacrifice a little local cache hit ratio. We can observe that the proposed MARDDPG has an improved neighbour hit ratio than all baseline algorithms since it learns the better policy by using LSTM leads to cache more popular content near users instead of fetching from the distant content server.

6.8. *The convergence performance*

From Fig. 10(a), we can notice that the rule-based replacement mechanisms like FIFO, LRU and LFU fluctuate around 33, 42 and 46, respectively, and do not increase training episodes. The reason is that the rule-based mechanism cannot learn from the environment. We can see that the DRL has an upward trend in the first 600 episodes and fluctuates around reward 62. In DRL, each agent learns from its local cache state, leading to more difficulty learning optimal strategy. The MAC raises slowly till 400 episodes and constantly fluctuates around reward 80 since the centralized controller simultaneously controls the

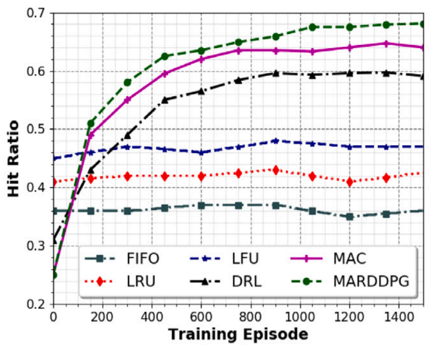


(a)

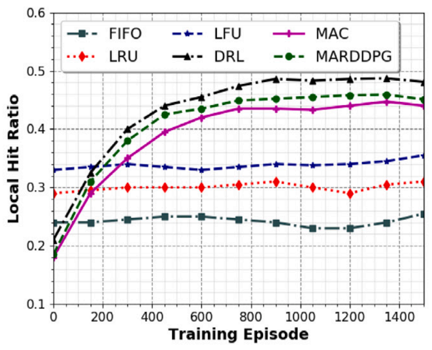


(b)

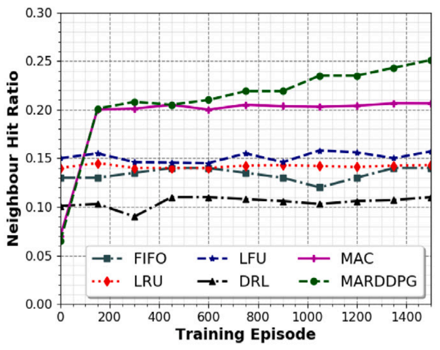
Fig. 8. Comparison of caching schemes using zipf shape parameter vs. (a) cache hit ratio (b) acceleration ratio.



(a)

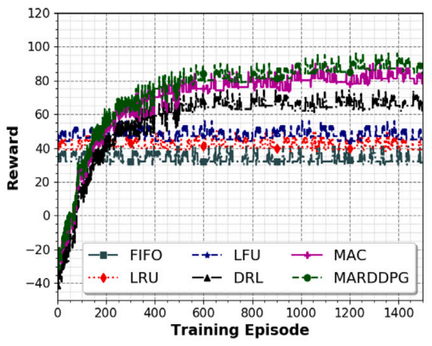


(b)

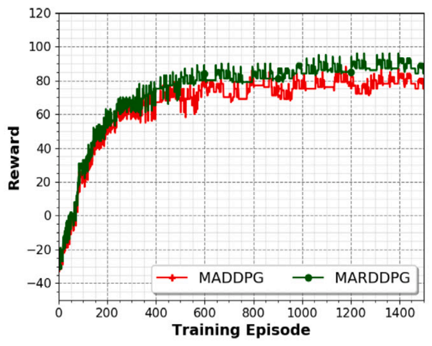


(c)

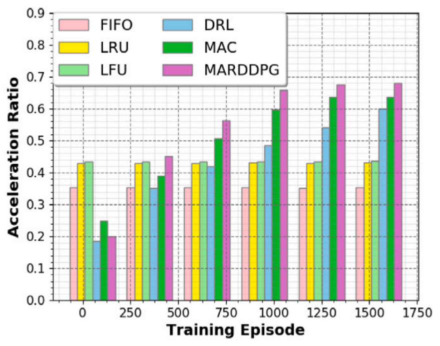
Fig. 9. Comparison of caching schemes using training episode vs. (a) cache hit ratio (b) local cache hit ratio (c) neighbour cache hit ratio.



(a)



(b)



(c)

Fig. 10. (a) Reward of all schemes vs. training episode (b) reward of proposed and MADDPG schemes during training episodes (c) training episode vs. acceleration ratio.

multiple agents in the environment leading to slow convergence. We can notice that the proposed MARDDPG curve increases quickly and fluctuates around reward 87, which outperforms all other references algorithms. Thus, the MARDDPG finds the best strategy quickly compared to other reference mechanisms in maximizing saved delay. Compared to DRL and MAC, MARDDPG has a 12 and 15 per cent increase in reward.

From Fig. 10(b), we can observe that both the curves have a similar trend. Both the curves raise quickly till episode 400 and then stabilizes slowly. We can notice that even though both curves have a similar trend, the proposed MARDDPG mechanism has a higher reward and more stability than the MADDPG. The reason is that the inclusion of LSTM enables the agents in the MARDDPG algorithm to learn a better policy compared to MADDPG, where the LSTM has not considered. Overall we can notice that the proposed MARDDPG outperforms all other reference algorithms by considering the LSTM for learning a better strategy to maximize the reward cooperatively among the agents.

In Fig. 10(c), the impact of training episodes on the acceleration ratio is presented. We can see that the rule-base mechanism does not increase as the training episode increase. The proposed mechanism raises quickly and stabilizes after episode 1000. That indicates that the MARDDPG learns a better policy quickly. That means the MARDDPG caches the more popular content cooperatively with fewer replacements leads to more saved delay. The DRL and MAC have less acceleration ratio since in DRL agent updates policy based on local cache state without cooperation. As in MAC, a centralized coordinator updates the agents' policy simultaneously.

7. Conclusion

In this work, we have considered caching in multi-cell scenario. Specifically, we have designed MARDDPG algorithm to maximize the saved delay in the cooperative mobile edge network. We have integrated the LSTM model in MADDPG to design the cooperative caching algorithm for multi-cell scenarios and discussed the network update in detail. Extensive simulations are performed to determine the performance of the proposed algorithm over existing algorithms. The proposed cooperative cache update algorithm outperforms the existing algorithms by considering performance metrics such as the cache hit ratio, acceleration ratio and reward. It has been shown that the proposed mechanism provides an improvement over other learning-based and non-learning (rule-based) based algorithms. It has been observed that the proposed mechanism improves up to 24, 17, 19, 11, and 4.5 per cent on cache hit ratio compared with FIFO, LFU, LRU, DRL and MAC, respectively.

CRediT authorship contribution statement

Manoj Kumar Somesula: Conceptualization, Methodology, Software, Formal analysis, Investigation, Resources, Writing – original draft, Visualization, Validation. **Rashmi Ranjan Rout:** Conceptualization, Investigation, Writing – review & editing, Supervision. **D.V.L.N. Somayajulu:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, M. Chen, In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning, IEEE Netw. 33 (5) (2019) 156–165.
[2] C. S. Inc., Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017—2022, White Paper, Cisco, 2019.
[3] J. Yao, T. Han, N. Ansari, On mobile edge caching, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2525–2553.
[4] L. Qiu, G. Cao, Popularity-aware caching increases the capacity of wireless networks, IEEE Trans. Mob. Comput. 19 (1) (2019) 173–187.
[5] E. Baştuğ, M. Kountouris, M. Bennis, M. Debbah, On the delay of geographical caching methods in two-tiered heterogeneous networks, in: 2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), IEEE, 2016, pp. 1–5.
[6] R. Wang, J. Zhang, S. Song, K.B. Letaief, Mobility-aware caching in D2D networks, IEEE Trans. Wirel. Commun. 16 (8) (2017) 5001–5015.
[7] M.S. ElBamby, M. Bennis, W. Saad, M. Latva-Aho, Content-aware user clustering and caching in wireless small cell networks, in: 2014 11th International Symposium on Wireless Communications Systems (ISWCS), IEEE, 2014, pp. 945–949.
[8] M. Chen, W. Saad, C. Yin, M. Debbah, Echo state networks for proactive caching in cloud-based radio access networks with mobile users, IEEE Trans. Wirel. Commun. 16 (6) (2017) 3520–3535.
[9] H. Zhu, Y. Cao, W. Wang, T. Jiang, S. Jin, Deep reinforcement learning for mobile edge caching: Review, new features, and open issues, IEEE Netw. 32 (6) (2018) 50–57.
[10] T.X. Tran, D.V. Le, G. Yue, D. Pompili, Cooperative hierarchical caching and request scheduling in a cloud radio access network, IEEE Trans. Mob. Comput. 17 (12) (2018) 2729–2743.
[11] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, W. Wang, A survey on mobile edge networks: Convergence of computing, caching and communications, IEEE Access 5 (2017) 6757–6779.
[12] W. Jiang, G. Feng, S. Qin, Optimal cooperative content caching and delivery policy for heterogeneous cellular networks, IEEE Trans. Mob. Comput. 16 (5) (2017) 1382–1393.
[13] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT press, 2018.
[14] C. Zhong, M.C. Gursoy, S. Velipasalar, Deep reinforcement learning-based edge caching in wireless networks, IEEE Trans. Cogn. Commun. Netw. 6 (1) (2020) 48–61.
[15] M.T. Spaan, Partially observable Markov decision processes, in: Reinforcement Learning, Springer, 2012, pp. 387–414.
[16] L. Li, G. Zhao, R.S. Blum, A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies, IEEE Commun. Surv. Tutor. 20 (3) (2018) 1710–1732.
[17] K. Shanmugam, N. Golrezaei, A.G. Dimakis, A.F. Molisch, G. Caire, Femtocaching: Wireless content delivery through distributed caching helpers, IEEE Trans. Inf. Theory 59 (12) (2013) 8402–8413.
[18] K. Poularakis, G. Iosifidis, L. Tassiulas, Approximation algorithms for mobile data caching in small cell networks, IEEE Trans. Commun. 62 (10) (2014) 3665–3677.
[19] Y. Sun, Z. Chen, H. Liu, Delay analysis and optimization in cache-enabled multi-cell cooperative networks, in: 2016 IEEE Global Communications Conference (GLOBECOM), IEEE, 2016, pp. 1–7.
[20] M.K. Somesula, R.R. Rout, D. Somayajulu, Contact duration-aware cooperative cache placement using genetic algorithm for mobile edge networks, Comput. Netw. 193 (2021) 108062.
[21] B. Bharath, K.G. Nagananda, H.V. Poor, A learning-based approach to caching in heterogenous small cell networks, IEEE Trans. Commun. 64 (4) (2016) 1674–1686.
[22] M.K. Somesula, R.R. Rout, D. Somayajulu, Deadline-aware caching using echo state network integrated fuzzy logic for mobile edge networks, Wirel. Netw. (2021) 1–21.
[23] B. Bharath, K.G. Nagananda, D. Gündüz, H.V. Poor, Caching with time-varying popularity profiles: A learning-theoretic perspective, IEEE Trans. Commun. 66 (9) (2018) 3837–3847.
[24] S. Podlipnig, L. Böszörmenyi, A survey of web cache replacement strategies, ACM Comput. Surv. (CSUR) 35 (4) (2003) 374–398.
[25] A. Feriani, E. Hossain, Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial, IEEE Commun. Surv. Tutor. (2021).
[26] A. Sadeghi, F. Sheikholeslami, G.B. Giannakis, Optimal and scalable caching for 5g using reinforcement learning of space-time popularities, IEEE J. Sel. Top. Signal Process. 12 (1) (2017) 180–190.
[27] S.O. Somuyiwa, A. György, D. Gündüz, A reinforcement-learning approach to proactive caching in wireless networks, IEEE J. Sel. Areas Commun. 36 (6) (2018) 1331–1344.
[28] Y. He, Z. Zhang, F.R. Yu, N. Zhao, H. Yin, V.C. Leung, Y. Zhang, Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks, IEEE Trans. Veh. Technol. 66 (11) (2017) 10433–10445.
[29] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, S. Jin, Caching transient data for internet of things: A deep reinforcement learning approach, IEEE Internet Things J. 6 (2) (2018) 2074–2083.
[30] W. Jiang, G. Feng, S. Qin, T.S.P. Yum, G. Cao, Multi-agent reinforcement learning for efficient content caching in mobile D2D networks, IEEE Trans. Wirel. Commun. 18 (3) (2019) 1610–1622.
[31] Y. Zhang, B. Feng, W. Quan, A. Tian, K. Sood, Y. Lin, H. Zhang, Cooperative edge caching: A multi-agent deep learning based approach, IEEE Access 8 (2020) 133212–133224.
[32] S. Chen, Z. Yao, X. Jiang, J. Yang, L. Hanzo, Multi-agent deep reinforcement learning-based cooperative edge caching for ultra-dense next-generation networks, IEEE Trans. Commun. 69 (4) (2020) 2441–2456.
[33] F. Wang, F. Wang, J. Liu, R. Shea, L. Sun, Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 2499–2508.
[34] K. Jiang, H. Zhou, D. Zeng, J. Wu, Multi-agent reinforcement learning for cooperative edge caching in internet of vehicles, in: 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), IEEE, 2020, pp. 455–463.
[35] T. Wu, P. Zhou, K. Liu, Y. Yuan, X. Wang, H. Huang, D.O. Wu, Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks, IEEE Trans. Veh. Technol. 69 (8) (2020) 8243–8256.
[36] B. Huang, X. Liu, S. Wang, L. Pan, V. Chang, Multi-agent reinforcement learning for cost-aware collaborative task execution in energy-harvesting D2D networks, Comput. Netw. (2021) 108176.
[37] J. Song, M. Sheng, T.Q. Quek, C. Xu, X. Wang, Learning-based content caching and sharing for wireless networks, IEEE Trans. Commun. 65 (10) (2017) 4309–4324.
[38] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, et al., MEC in 5G Networks, ETSI White Paper 28, 2018, pp. 1–28.
[39] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38.
[40] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, 2017, arXiv preprint arXiv:1706.02275.
[41] R.E. Wang, M. Everett, J.P. How, R-MADDPG for partially observable environments and limited communication, 2020, arXiv preprint arXiv:2002.06684.
[42] F.M. Harper, J.A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 19:1–19:19, http://dx.doi.org/10.1145/2827872, URL http://doi.acm.org/10.1145/2827872.
[43] N. Garg, M. Sellathurai, V. Bhatia, B. Bharath, T. Ratnarajah, Online content popularity prediction and learning in wireless edge caching, IEEE Trans. Commun. 68 (2) (2019) 1087–1100.
[44] K. Poularakis, L. Tassiulas, Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks, IEEE Trans. Mob. Comput. 16 (3) (2016) 675–687.
[45] G. Ahani, D. Yuan, Optimal scheduling of content caching subject to deadline, IEEE Open J. Commun. Soc. 1 (2020) 293–307.
[46] X. Li, X. Wang, S. Xiao, V.C. Leung, Delay performance analysis of cooperative cell caching in future mobile networks, in: 2015 IEEE International Conference on Communications (ICC), IEEE, 2015, pp. 5652–5657.
[47] T.X. Tran, D. Pompili, Adaptive bitrate video caching and processing in mobile-edge computing networks, IEEE Trans. Mob. Comput. 18 (9) (2018) 1965–1978.

[48] M. Ahmed, S. Traverso, P. Giaccone, E. Leonardi, S. Niccolini, Analyzing the performance of LRU caches under non-stationary traffic patterns, 2013, arXiv preprint arXiv:1301.4909.
[49] D. Rossi, G. Rossini, Caching Performance of Content Centric Networks Under Multi-Path Routing (and More), Relatório técnico, Telecom ParisTech, 2011, pp. 1–6.
[50] A. Jaleel, K.B. Theobald, S.C. Steely Jr, J. Emer, High performance cache replacement using re-reference interval prediction (RRIP), ACM SIGARCH Comput. Archit. News 38 (3) (2010) 60–71.

Manoj Kumar Somesula received B.Tech. degree in computer science and engineering from Mahaveer Institute of Science and Technology, JNTU, Hyderabad, India and M.Tech. degree in computer science from School of Information Technology, JNTU, Hyderabad, India. He is currently pursuing Ph.D. at the National Institute of Technology (NIT) Warangal, India. His primary research area includes edge computing, wireless networks, content centric networks and Internet of Things.

Rashmi Ranjan Rout received Ph.D. from Indian Institute of Technology (IIT) Kharagpur, WB, India. He is currently working as an Associate Professor in the Department of Computer Science and Engineering at National Institute of Technology (NIT), Warangal, India. His research interest includes online social networks, P2P networks, delay tolerant networks, wireless sensor networks, vehicular networks, Internet of Things, cloud computing and network security.

D.V.L.N. Somayajulu is currently on deputation as Director for Indian Institute of Information Technology Design and Manufacturing (IIITDM), Kurnool, Andhra Pradesh. Prior to joining this post, he served as professor of Computer Science and Engineering, National Institute of Technology (NIT), Warangal and Chair, Electronics & ICT Academy, NIT Warangal. He obtained his Ph.D. from Indian Institute of Technology (IIT) Delhi, India. His research interest includes Databases, Data Analytics, Information Extraction, Query Processing, Big Data and Privacy.