Wireless Networks (2023) 29:569–588
https://doi.org/10.1007/s11276-022-03135-1

ORIGINAL PAPER

Deep reinforcement learning mechanism for deadline-aware cache placement in device-to-device mobile edge networks

Manoj Kumar Somesula[1] · Sai Krishna Mothku[2] · Anusha Kotte[3]

Accepted: 6 September 2022 / Published online: 6 October 2022

Abstract
Caching the most likely to be requested content at the mobile devices 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. In addition to content popularity, it is also essential to consider the users' dynamic behaviour for real-time applications, which can further improve communication between user devices, leading to efficient content service time. Most previous studies consider stationary network topologies, in which all users remain stationary during data transmission, and the user can receive desired content from the corresponding base station. In this work, we study an essential issue: caching content by taking advantage of user mobility and the randomness of user interaction time. We consider a realistic scenario in a cooperative caching problem with user devices moving at various velocities. We formulate the cache placement problems as maximization of saved delay with capacity and deadline constraints by considering the contact duration and inter-contact time among the user devices. A deep reinforcement learning-based caching scheme is presented to solve the high dimensionality of the proposed Integer linear programming problem. The proposed caching schemes improve the long-term reward and higher convergence rate than the Q-learning mechanism. Extensive simulation results demonstrate that the proposed cooperative caching mechanism significantly improves up to 23 % on hit ratio, 24 % on acceleration ratio and 25 % on offloading ratio compared with existing mechanisms.

Keywords Device-to-Device mobile networks · Cooperative cache placement · Deep reinforcement learning · DDPG · User mobility

1 Introduction

The proliferation of 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 (MEN) [3]. 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 (i.e., base stations and mobile devices) as caching nodes to bring the contents near users, thus alleviating the core network

✉ Manoj Kumar Somesula
manojkumar.somesula@gmail.com

Sai Krishna Mothku
saikrishna@nitt.edu

Anusha Kotte
anusha.k2291@gmail.com

[1] Department of Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Hyderabad 500090, India

[2] Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, Tamil Nadu 620015, India

[3] Department of Computer Science and Engineering, JNTUH College of Engineering Hyderabad, Hyderabad 500085, India

burden and enhancing user Quality of Experience (QoE) [3]. So, the edge nodes can serve a massive amount of duplicate content requests and this reduces the service delay and reduces the content delivery distance. Caching content at mobile devices with the use of device-to-device (D2D) communications [5] can alleviate the stress on the backhaul links by offloading network traffic to D2D links as contrary to base station caching [6]. In addition to providing content near to users, caching at mobile devices even reduces network latency [7]. Therefore, this can support latency-critical mobile applications in a mobile edge computing framework.

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 [8]. Since the user demands follow skewed distribution, most of the users' demands are for a few contents [9]. Hence, caching the content proactively during the off-peak time enhances the network performance. Due to the enormous amount of content and limited cache capacity, optimizing proactive caching strategy by utilizing the user requests is crucial in obtaining the advantage of edge caching [10]. Most proactive caching mechanisms cache the content cooperatively at base stations (BS) to reduce fetching content from a central server [11]. Some earlier works [5, 12–15] focuse on cooperatively caching data at mobile users for static D2D mobile networks. In a dense network, the assumption stated by previous publications [5, 12–15] is unreasonable. This study looks at the cache placement problem in a real setting where users of varying speeds connect to other users and BSs at irregular intervals. Users will consistently move among users and will be able to download only portions of the requested content from the various users they encounter throughout their path. If the user cannot receive the entire content from the contacted user, the requested content is retrieved from a base station, which ultimately enhances delay and adversely impacts QoS. Even though [16–18] assume user mobility, the randomization of contact duration is not taken into account. Data transmission is related to contact duration, according to [19]. The user travels at a fast speed if the contact time is short and at a low speed if the contact time is long. As a result, contact time randomization produced by user mobility has an impact on data transmission, which in turn has an impact on content placement. Hence, we want to build caching approach that take into account limited resources, content popularity, deadlines, contact length randomization (user speed), and user mobility.

A few challenges need to be handled when it comes to efficient caching in D2D enabled MEN. First, the user's mobility, short contact time, and inter-contact time make it challenging for mobile devices to cache content proactively. Second, determine eligible mobile devices for content caching, as devices with long duration's of communication coverage may be enabled to cache acceptable content. Third, determine the number of the content segments that need to be cached at the eligible mobile devices. In order to achieve real-time device selection, some learnt patterns produced by resource availability in a dynamic mobile edge network environment must be exploited. As a result, real-time content caching on mobile devices improves user experience while maintaining a tolerable response latency.

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 with the randomness of user mobility by considering the limited storage at each edge node. The mobility aware cooperative content caching using D2D communications is modeled as an Integer linear programming problem. As the number of mobile devices grows, it becomes more challenging to locate possible mobile devices in real-time using exhaustive search algorithms. Hence, we proposed a DRL (deep reinforcement learning) based cooperative caching mechanism to identify possible mobile devices to be cached. We apply deep deterministic policy gradient (DDPG) to speed up the learning process of the proposed approach because of the slower convergence of traditional learning methods induced by vast action space and higher dimensionality. 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 D2D enabled mobile edge networks.
- Formulating the designed cooperative cache placement problem as a Markov decision process problem to maximize the cumulative reward by ensuring the coordination of the mobile users.
- Design a cooperative caching scheme based on deep reinforcement learning mechanism, DDPG to speed up the learning rate and enhance content placement at mobile users to maximize the saved delay.
- Extensive simulations have been performed to show the efficacy of the proposed deep reinforcement learning based cooperative caching algorithm by considering

acceleration ratio, hit ratio, offloading ratio and caching reward.

The rest of this paper is organized as follows. The summary of related work is discussed in Sect. 2, and the network model, user mobility model, content request model and problem formulation are presented in Sect. 3. The deep reinforcement learning algorithm, DDPG is presented in Sect. 4. The simulation environment, and the results are discussed in Sect. 5. Concluding remarks are provided in Sect. 6.

2 Related work

Content caching has been studied widely in the literature. Li *et al.* [9] have presented a survey on content placement and delivery mechanisms in various cellular networks. The advantages of using content caching in mobile networks were highlighted by Zhou *et al.* [20]. Practically users request a small number of frequently accessed top-ranked content. Content pre-fetching that depends on content popularity has been investigated in the literature [12, 21]. Shanmugam *et al.* [12] have proposed a caching mechanism for each helper node to reduce the expected download delay. Authors in [12] studied a content assignment problem in an uncoded scenario and showed the proposed problem is NP-hard, and presented a 2-approximation algorithm. Poularakis *et al.* [21] presented an approximation approach for content cache placement in small cell network (SCN) using content popularity knowledge. To minimize the load on the backhaul, the authors proposed a mobility framework by modeling the random walks on the Markov chain. Collaborative cache placement has been investigated in SCN to handle the limitation of cache capacity at each node to improve the user QoS [8, 22, 23]. The works mentioned above consider the content popularity known in advance. Moreover, popularity prediction based caching strategies were also studied in [10, 24, 25]. In [24], authors proposed a learning theoretic perspective for content caching heterogeneous networks with time-varying and unknown popularity profiles. Further, the authors presented a cache update mechanism showing better performance for periodic updates. Chen *et al.* [10] have presented an echo state network to estimate the content popularity and node mobility to maximize the user QoE in unmanned areal vehicle placement. However, the works mentioned above consider the content popularity prediction and neglected the device caching capabilities, the dynamic user requests and environment complexities. In contrast, our work considers the reinforcement learning to handle the dynamic environment.

The increasing mobile devices' capabilities exploit user devices as the caching nodes to bring the content near users. Utilizing the user devices as caching nodes reduces user-perceived latency and enhances user QoS. There have been some studies focusing on device-to-device communications. Wang *et al.* [26] have introduced a D2D big data framework for intelligent content dissemination and offloading. Li *et al.* [13] have investigated a distributed caching mechanism to select appropriate user nodes and allocate content at selected nodes in cellular networks using D2D communications. The authors aim to maximize social welfare and minimize the delay at the same time. Based on the user's social relationship, the social welfare is modeled as a many-to-one game-further, the authors show that the presented scheme performs better than existing schemes and is stable. Pan *et al.* [5] have studied the data offloading using D2D communication, assuming the physical transmission requirements and cooperation among users to maximize the offloading gain. The users request the desired content from trustworthy users in the D2D proximity. Further, the authors presented an iterative scheme depending on the asymptotic approximation of success probability. Prerana *et al.* [6] have discussed the classification, challenges and solutions of employing device-to-device communications in a 5G environment. Wu *et al.* [27] have presented a content sharing framework with placement and delivery stages and proposed a cooperative caching mechanism in 5G environment. Yang *et al.* [14] have presented a cost-aware energy-efficient data offloading technique to trade-off between cost, energy efficiency and delay. The data offloading problem is formulated as an optimal control problem and presented as an approximation scheme. Liu *et al.* [28] studied an optimal caching mechanism by modelling a multi-objective optimization problem to maximize the hit rate and minimize the ergodic delay and outage probability. Fu *et al.* [29] have studied a caching mechanism to improve the caching quality of device-to-device facilitated mobile networks. Nevertheless, the above-mentioned earlier studies only consider fixed network architectures, do not consider user mobility, and employ machine learning mechanisms to place the content.

Some works in the literature consider user mobility. Wang *et al.* [16] have presented an efficient D2D offloading mechanism using the predicted contact period metric expected available duration to assess the opportunity that an object may be retrieved via a D2D link. Qiao *et al.* [18] studied a cache mechanism founded on the Markov decision process to maximize the QoS in vehicular networks with highway network scenarios. Due to user mobility, there may be frequent interconnections with the users who may suffer from connection delays and long hand-offs. A decomposition-based scheme is presented to

dynamically solve the dynamic programming problem of allocating the available storage space to the users. Zhao *et al.* [30] have presented user mobility and interest prediction based cache placement and cache-aware replacement mechanisms to maximize the utility in IoT networks. Zhang *et al.* [31] have investigated the user interest request prediction based mobility aware D2D caching mechanism to maximize the opportunistic cache utility. In [32], the authors presented a network slicing mechanism to efficiently handle heterogeneous resources in a vehicular network with assured QoS. Ibrahim *et al.* [33] have presented a coded caching mechanism to reduce device-to-device communication-based data dissemination load. Sun *et al.* [34] have investigated a caching mechanism in a D2D environment by considering the user mobility to minimize the network latency. In [23, 35] , the authors have designed the content placement in a small cell environment using mobility and contact duration to reduce the traffic load. Zhou *et al.* [36] have investigated the reverse auction-based incentive-driven deep reinforcement learning scheme to reduce the burden on backhaul links. However, the studies mentioned above consider mobility and the authors have not considered the learning perspective in the D2D networks. In contrast, we utilise a machine learning scheme to solve the high dimensionality problem in a cache-enabled D2D network.

Moreover, some recent works have exploited the ML-based techniques in wireless communications [13, 25, 30, 36–40]. In [38], Qiu *et al.* have presented a model-free DRL based online offloading scheme for blockchain empowered MEC networks. Bajpai *et al.* [37] have presented content caching in the device-to-device network using deep learning-based recurrent neural networks and transformers for better performance. He *et al.* [40] have presented a Deep RL (DRL) mechanism to make resource allocation decisions optimally. Li *et al.* [13] have presented two deep reinforcement learning-based caching schemes to design efficient content placement and a delivery scheme. Furthermore, some familiar RL approaches, including SARSA [30] and deep Q-network (DQN), are explored and employed in real-world communication networks. Zeng *et al.* [41] have studied a DRL based caching mechanism in a D2D scenario to maximize the hit ratio. Jiang *et al.* [42] 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. Li *et al.* [43] have presented two DRL mechanisms for content placement and a content delivery mechanism in IoT networks. Chakraborty *et al.* [44] have presented a cache mechanism to enhance cache hit rate by utilizing the LSTM and CNN. In [28], the authors proposed a DRL based caching mechanism to minimize the outage probability. Authors in [36] present a Deep Q-Network based incentive mechanism. However, the works mentioned above consider the learning perspective in the D2D networks. The authors of earlier works have not taken environment complexities, user mobility, user velocity and randomness of contact duration. This inspires us to utilise machine learning methodologies to solve the exciting problem of designing an efficient content placement mechanism in a cache-enabled D2D network while considering the contact duration and inter-contact time.

3 System model and problem formulation

In this section, the network model, user mobility model, content delivery 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 macro base stations (MBS) equipped with a MEC server, a set $\mathcal{U}$ of U mobile users with limited cache capacity and a content server as shown in Fig. 1. Each mobile device $u \in \mathcal{U}$ has a limited cache S_u called local storage. The storage of each mobile device is used for content caching. The content server acts as an origin server that stores all contents. A user may be in the communication range of more than one user but user can communicate with only one user at a particular time. The user may get the desired content from more than one mobile users who are in the communication range of the user. The mobile users are communicated to each other using the device-to-device communication. 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 in 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 user receives content requests from multiple users in the communication range without knowing its popularities. The user can serve the requests in three ways:

– *Local storage* The mobile user checks the local storage of the user. If the requested content is available on the mobile device, then receive the content within the deadline. Otherwise, the content is obtained from either

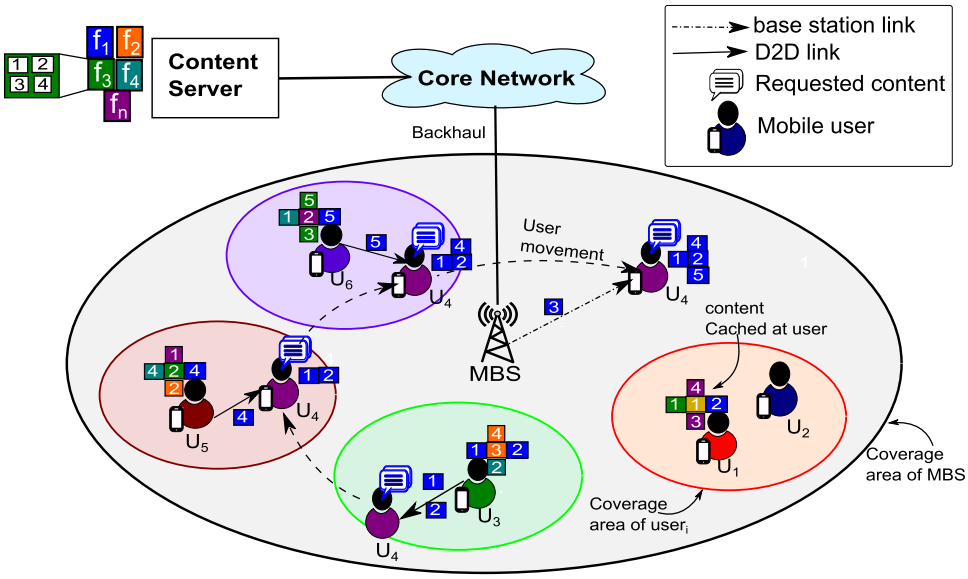


Fig. 1 Illustration of the proposed system model

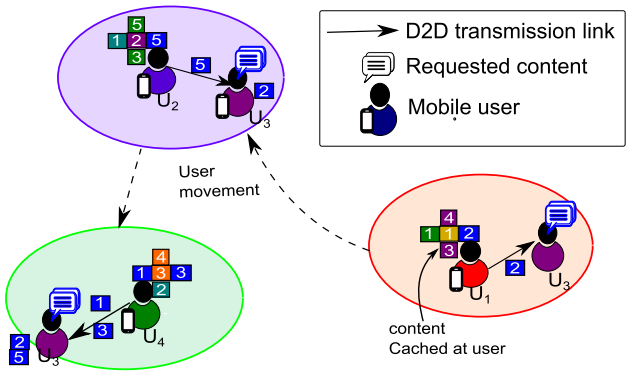
neighbour user or base station based on the availability of the content.

- *Neighbour user* If the user requested content is not available at the local storage, then the content can be obtained from the nearby users in the communication range based on the availability of the content at neighbouring users.
- *Base station* If the requested content does not exist with any users in the communication range, the content would be served from the associated base station by fetching the content from the central server through a backhaul link.

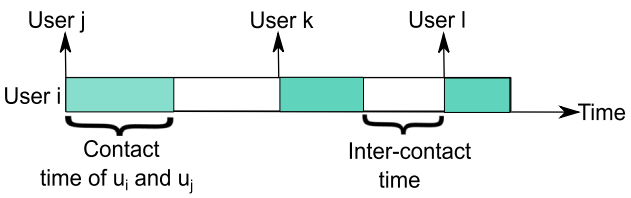
It is depicted in Fig. 1 that each content is shown in various colours, divided into multiple segments. Each segment of the content is denoted with a number. The user with a color demonstrates that the user is requesting the content of the same color, and the coverage area of a user is depicted with a circle of the same color. The user needs to acquire all the numbered segments of the same color to get the desired content. A user U_4 is moving across the MBS starting from U_3 and requesting content f_1. Since U_4 is in the coverage of U_3, U_4 gets the content segments 1 and 2 of content f_1 and moves to U_5. U_4 gets segment four from U_5 and segment five from U_6. The leftover segment (3) is obtained from the MBS. U_4 gets the desired content by collecting the content segments from the different users and MBS.

3.2 User mobility model

The user mobility pattern and contact information has been modelled similar to [32]. The mobile user moves across the base stations while contacting multiple users in the moving path. The users in the communication range of the moving user are called the neighbours, and these neighbours met with a particular average time as illustrated in Fig. 2b. The mobile users in contact may exchange the content based on the contact time and inter contact time shown in Fig. 2a. The contact and inter contact pattern between two users in the given network is considered a set of independent Poisson processes and these random variables follow an exponential distribution with parameters $\lambda_{u,v}$. The time between two sequential contact times of two users is denoted as inter contact time. The average contact rate of



(a) Sample Mobility scenario



(b) Inter-contact and contact duration

Fig. 2 Illustration of the mobility model based on inter-contact time and contact duration

users u and v is denoted as $\lambda_{u,v}$, determined from historical information.

3.3 Content request model

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 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. In this work, we consider the Maximum Distance Separable (MDS) codes to encode the given content into multiple segments to reduce the redundant content caching at mobile devices and improve efficiency. Each mobile device caches the encoded content segments rather than original content. The content requested user need to collect at least B bits to retrieve the original content.

The user may request the content based on the content popularity. In this work, we consider that each user has different content preferences. When a user u requests a content f, the requested content is served immediately if available in the local storage. If the content is not available at local storage, the requested content can be served from the users who come across the user moving path. Otherwise, the user retrieves the content from the base station, which is associated with the requesting user. The contact duration of users u and v is indicated as $C_{u,v}$, the content transmission rate is indicated as $r_{u,v}$, and the number of segments transmitted from user v to u in one contact is $\beta_{u,v}$. If the requesting user obtains l segments from the users in contact, then the remaining content segments will be retrieved from the associated base station. The list of symbols used in this work are listed in Table 1.

Table 1 List of Notations

Term	Definition
$\mathcal{U}$	Set of base stations
$\mathcal{F}$	Set of contents
S_u	The cache capacity of u-th user
S_f	The size of f-th content
dl_f	The deadline of content f
$t \in T$	Time slot
$\mathcal{C}_{u,v}$	Contact duration of user u and v
$r_{u,v}$	Content transmission rate between user u and v
$\beta_{u,v}$	Number of segments transmitted in one contact between user u and v
x_f^v	Number of coded segments of content f cached at user u
$y_f^{v,l}$	Useful content f downloaded from user u in l th contact
α_u^v	Number of appearances of user in v
d_u^v, d_u^m, d_u^{cs}	The delay for transmitting unit of coded content to mobile user u from user, base station, content server
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
$c_i, c_{i,j}, c_{i,h}$	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
$y_{i,j}^t$	Target network
$\emptyset, \theta$	Actor and critic network weight parameters

3.4 Definitions

Definition 1 (Contact time) The two mobile devices in the communication range of each other and can share the content with other devices in communication range is known as the contact time.

Definition 2 (Inter-contact time) The time between each consecutive contact of two mobile devices is known as inter-contact time.

Definition 3 (Saved delay) The difference between the download delay from the content server and the mobile device is defined as a saved delay.

Definition 4 (Deadline) The requested content needs to be served within the given time limit, which describes the maximum allowable time for the service denoted as deadline.

3.5 Problem formulation

The caching mechanism of encoded segments in a mobile user is indicated as $X_{v\times f}$, where $x_f^v \in X$ represents the number of coded segments f cached at user v. Because of the user mobility, the user may contact with same user multiple times and communicate with multiple users in the moving path. The valuable content retrieved by the user in first contact with other users is denoted as

$$y_f^{v,1} = \min\left\{x_f^v, C_{u,v}\frac{r_{u,v}}{S_f}\right\} \quad (1)$$

Retrieving the valuable content f by user in the

l^{th}contact($l \in \{2, 3, \ldots, \alpha_u^v\}$) with v^{th} mobile user is $x_f^{v,l} = x_f^v - \sum_{t=1}^{l-1} y_f^{v,t}$ represented as

$$y_f^{v,l} = \min\left\{x_f^{v,l}, C_{u,v}\frac{r_{u,v}}{S_f}\right\} \tag{2}$$

Retrieving a content f successfully determines that the number of encoded segments obtained by mobile users needs to satisfy at least the requested contents' size. The content retrieved from the users who come across the path is denoted as $\psi_f = \sum_{v\in U, v\neq u}\sum_{l\in\alpha_u^v} y_f^{v,l}$. Since the user can get the desired content from the neighbouring users and MECs within the communication range, first, we formulate the delay to get the content from the users and then formulate the delay to get the content from the MEC. The average number of encoded segments retrieved from the users in the path by a user is denoted as

$$\psi_f^u = \sum_{f\in\mathcal{F}} p_f \cdot \psi_f \cdot (d_u^{cs} - d_u^v) \tag{3}$$

Here, d_u^{cs} and d_u^u denote the delay in transmitting the content from the content server and user. The term $(d_u^{cs} - d_u^u)$ gives the saved delay. The delay in transmitting the content from MEC to the user is computed using Eq. (2) (i.e., the reasonable amount of content can be retrieved from the associated MEC). Hence, the average number of encoded segments retrieved from the MEC in the path by a user is denoted as

$$\psi_f^m = \sum_{m\in\mathcal{M}}\sum_{f\in\mathcal{F}} p_f \cdot \min\left\{x_f^v, C_{u,m}\frac{r_{u,m}}{S_f}\right\} \cdot (d_u^{cs} - d_u^m) \tag{4}$$

Therefore, the total number of encoded segments retrieved from the other users and MEC is denoted as

$$\psi = \frac{1}{U}\sum_{v\in U} \psi_f^u + \psi_f^m \tag{5}$$

ψ gives the total delay to get the content from the neighbouring users and MECs.

We aim to maximize the saved delay by placing the encoded segments of requested content at each user device subjective on deadline and capacity constraints. Hence, the problem is formulated as:

$$\max\ \psi^t \tag{6}$$

s.t.

$$\sum_{f\in\mathcal{F}} S_f \cdot x_f^v \leq S_u, \forall v\in\mathcal{U} \tag{7}$$

$$\psi^t \leq dl_f, \forall v\in\mathcal{U}, \forall f\in\mathcal{F} \tag{8}$$

$$x_f^v \leq S_f, \forall v\in\mathcal{U}, \forall f\in\mathcal{F} \tag{9}$$

$$x_f^v \in \{0, 1, \ldots, B\}, \forall v\in\mathcal{U}, \forall f\in\mathcal{F} \tag{10}$$

The objective (6) is the total saved delay caused by users of the overall network. The constraints (7) ensures that the the number of segments cached at user should not exceed the capacity of each user device. Constraints (8) assures the maximum permissible delay for the reply to a demand. Constraints (9) specifies that each user does not require to cache more than B segments to avoid redundant storage. The constraints (10) is the non-negativity constraint of the decision variables.

The content placement problem presented in Eq. (6) is a mixed integer non-linear programming (MINLP) problem and proved as NP-hard [17, 35]. The problem presented in Eq. (6) can be addressed by finding the optimal decision variables $\{X^t\}$ in the present time slot. The distribution of the summation of U independent random variables is required to evaluate Eq. (6). Nevertheless, the decision variable present in Eq. (6) is an integer variable and changing dynamically which requires to gather a huge quantity of network state information. Besides, the exponentially increasing number of mobile users experience high computational complexity for the straightforward computation of the proposed objective function. Because of the user mobility, randomness of contact and to take an intelligent caching decision, we cannot adopt the conventional optimization methods [45]. The continued advancements and strong characteristic representation capabilities of Deep Learning (DL) [46] have encouraged learning in wireless networks. Hence, we designed a deep reinforcement learning scheme to cooperatively cache content at user devices.

4 Deep reinforcement learning mechanism

We designed a reinforcement learning mechanism to solve the proposed problem where the agent learns an optimal caching scheme to cache the content at the user devices in the mobile edge network. RL facilitates the agent to learn from the environment through trial and error with the help of its own experiences and feedback by interacting with the environment by performing actions. The RL agent aims to maximize the overall reward by greedily exploiting all probable actions till the most suitable set of actions. In this work, we consider the states and actions as the time and mobile devices, respectively. Besides, the reward is designed as a function of saved delay. In the dynamic scenario, the agent does not know mobile devices directly as a fixed set of actions. Hence, we present deep reinforcement learning mechanism to select the more preferable mobile devices based on the highest reward. The base station collects and maintains the complete information about each mobile device in its transmission range, like the

number of devices contacted, average contact time, data transmission rate, and node storage capacity. *Devices encountered:* Number of devices encountered in the mobile user path is denoted as *E*. *Contact time:* The average contact time of the mobile device is denoted as *T*. *Transmission rate:* The amount of content transmitted in a contact time is denoted as *R*. The efficient content caching mechanism using the DRL mechanism, DDPG, is presented in subsection 4.1.

4.1 Content caching using deep reinforcement learning

It is harder to ensure maximum saved delay in Eq. (6) since different users have different needs for limited resources. The proposed problem needs to gather a massive amount of network state information. Due to the dynamic nature of the problem and to take an optimal caching decision, we cannot adopt the conventional optimization methods. With the huge success of reinforcement learning (RL) and deep learning (DL) in the recent years motivated us to use the deep reinforcement learning (DRL) in wireless networks which is considered as an effective technique to handle complex problems. Inspired by the benefits of DRL in resource management in wireless networks, we use a deep deterministic policy gradient (DDPG) mechanism to attain an efficient solution for Eq. (6). The agent in DRL gathers the necessary information about the mobile device resources as well as the various requests of users. After that, the agent handles the caching decisions by taking action.

4.1.1 Problem formulation based on DRL

Implementing DRL in this mechanism aims to enhance the system's adaptability in a challenging (dynamic) environment. The caching decisions are determined based on the present state which is not depend on the previous state information. Hence, we can model the the proposed problem as Markov decision process (MDP). A MDP is defined as a tuple $\{S, A, R, P, \gamma\}$. S defines the sate space, A defines the action space, R defines the system reward, P defines the immediate reward and γ denotes the discount factor.

- Let S is the set of system state space where $S = \{s_i | s_i = (N_i^t, K_i^t, B_i^t, \psi_i^t, \xi_i^t)\}$. 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, number of devices encountered ψ_i^t, average contact duration ξ_i^t and available cache size C_i^t at user i. Where $K_i^t = \{k_{i,1}^t, k_{i,2}^t, \ldots, k_{i,U}^t\}$, $k_{i,u}^t$ is the contents requested by user u at user i in time t, $B_i^t = \{b_{i,1}^t, b_{i,2}^t, \ldots, b_{i,F}^t\}$, $b_{i,f}^t$ is the content delivery deadlines of user i for accessing the requested content f in time t.
- A is the set of actions where $A = \{a_1, a_2, \ldots, a_n\}$. The action represents a set of mobile devices over which the content is cached to maximize the saved delay. The number of mobile devices in each action is equal to the number of wireless channels. The actions entirely depend on the state information (i.e., each state may have different actions). $A = \bigcup_{t \in T} a_i^t, \forall s_i \in S$ represents the action space analogous to the state space S. The agent selects the actions depending on the current policy.
- R is the immediate reward r_i obtained by performing an action a_i on the environment with state s_i. The reward also contains penalties in case of sufficient resources not available. This work maximises the saved delay by obtaining the desired content at a low transmission delay within the fetching deadline. Each mobile device replaces the cached content if the cache is full otherwise caches the content at the local storage. In the cooperative environment, based on the availability of the content, either neighbouring mobile users or the BS serves the users requests. The user's local storage is denoted as the local user, the nearby users in the communication range of mobile users are called neighbouring users, and the BS is associated with the user called central server.

1. Suppose the content requested by user is available at the local storage, the content can be delivered immediately with low latency. The cost of delivering content from local user is denoted as c_i. Let us consider that the user i fetches l contents from its local storage in time t is indicated as $r_{i,l}^t$. Therefore, the cost of the local user service is represented as $c_i r_{i,l}^t$.
2. Suppose some of the contents requested by the user are not served by the local storage i. Let us consider that the content requested by user is available at neighbour user j, and the content is served by j to the user i. The cost of fetching content from j to i is denoted as $c_{i,j}$. Let l contents are fetched from j to i in time t is denoted as $r_{i,j}^t$. Therefore, the cost of neighbouring user 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 users. The corresponding BS obtains the content from the content server. Let us consider the cost to get the content from the content server to the user i via BS h denoted as $c_{i,h}$. Let l 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 miss at local storage and neighbouring users. Hence, the user replace the newly fetched content with less popular content. Therefore, the cost should contain the replacement cost along with the delivery cost. Let the cost of replacing content at user i is denoted by α_i. The number of content segments replaced by user 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 user i in time t, $x_{f,i}^{t-1}$ indicates the content f cached in user i at time $t-1$ and f_i indicates the content requests at user 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 user i and $\rho_{i,f}^t$ is the content frequency.

The cost of utilizing the neighbour users 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 user i is denoted as

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

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 the proposed system, each BS is considered as an agent. Based on the systems sates, each agent determines its cache placement. We indicate $\pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ set of all caching strategies, $\pi : S \rightarrow A$ 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 RL. 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 $0 \leq \gamma < 1$ 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. Assume that there is a list of all acceptable policies Π. The optimal policy is then determined as

$$\pi^* = \underset{\pi\in\Pi}{\operatorname{argmax}}\, V^\pi(S) \tag{18}$$

- P is the transition probability. Agent performs an action based on the transition probability from one state to the next state.
- γ is the discount factor which is in the range of 0 to 1. The $\gamma = 1$ means the agent evaluates its actions based on sum total of all its future rewards and $\gamma = 0$ means the agent learns an action based on the immediate reward.

4.2 DDPG framework for cache placement

There are two learning mechanisms in RL, model-based RL and model-free RL. A transition table is used by model-based RL algorithms (such as value and policy iterations). A transition table can be thought of as a life hack book that comprises all the information an agent requires to succeed in the world it exists. Model-free RL algorithms such as Q-Learning and SARSA do not employ the transition probability distribution associated with Markov Decision Processes (MDP). However, they learn the optimal policy through observation and experimentation. The maximum reward (i.e., reinforcement signal) obtained from all potential actions is used to update a Q-learning agent's policy, independent of the agent's policy. On the other hand, SARSA learning updates the agent's policy immediately from the actions performed, which are dependent on the current policy. However, when the number of nodes in the network is large, the RL-based approach has several drawbacks, such as inefficiency and instability. Because of

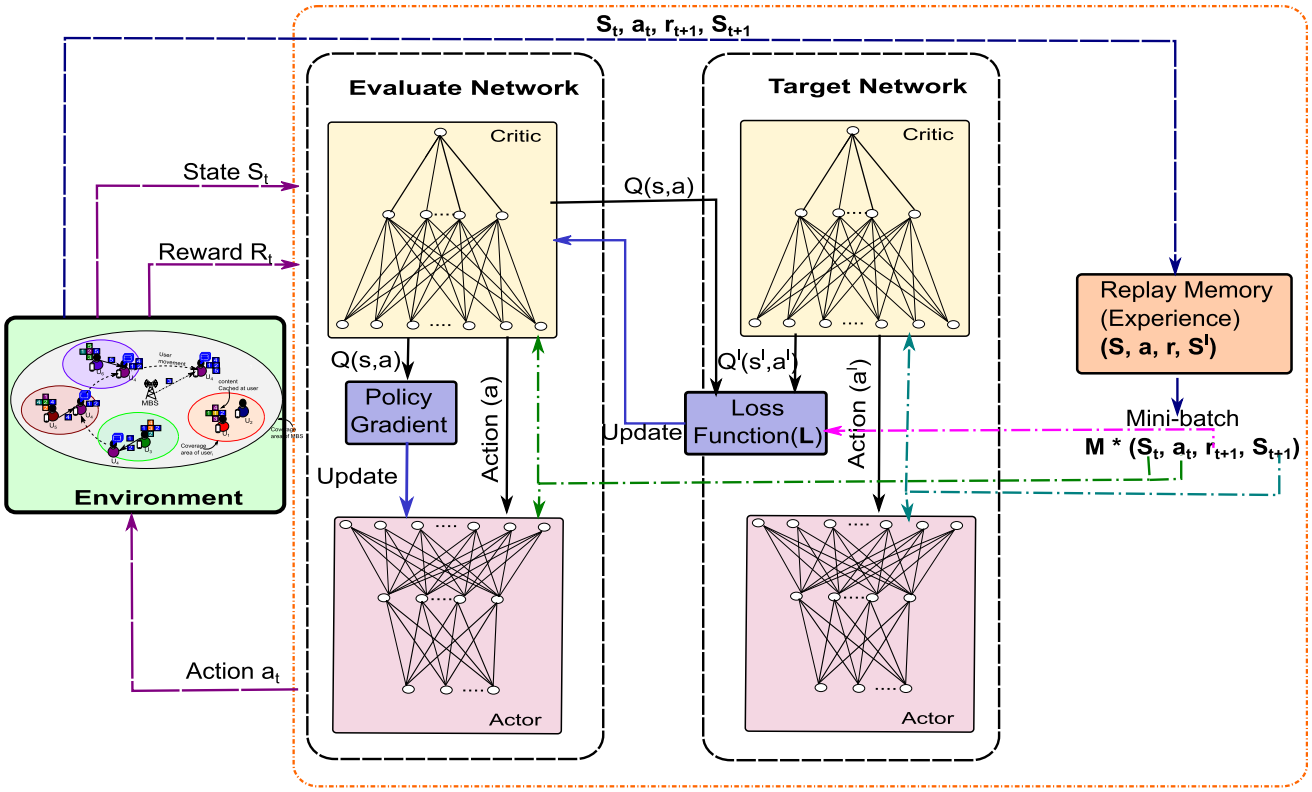


Fig. 3 Schematic diagram of DDPG Algorithm

its great features, excellent performance, and sufficient processing time, deep reinforcement learning, a combination of RL and deep neural networks, has been widely employed in wireless communication.

We apply DRL to address the presented optimization problem by characterizing the state, action, and reward. Q-learning is indeed a classic instance of reinforcement learning. Q-learning employs a Q-table to keep track of Q values for various state and action orders. Nevertheless, this might not be appropriate for cases having enormous state and action spaces. Deep Q-Network (DQN) calculates Q values using a deep neural network, accommodating greater dimensional states and action spaces but having a slow convergence speed. We use the deep deterministic policy gradient (DDPG), a model-free and actor-critic approach to solving the caching problem.

Deep neural networks (DNNs) approximate the policy and value functions in the DDPG algorithm, which is an expanded version of the actor-critic method. Compared to traditional RL algorithms, DDPG can solve optimization problems with huge state and action spaces with high dimensional concerns. Further, with continuous action spaces, DDPG may ensure efficient decisions. The structure of DDPG learning algorithm is shown in Fig. 3. The DDPG network consists of three essential components replay memory, primary and target networks. The actor-network and the critic network each have two neural networks, making four neural networks for the primary network and the target network. The actor-network is used to investigate policies, whereas the critic network is used to evaluate policies. In order to enhance the policy gradient, the critic network also provides critic values. The state-action pairings, the associated reward, and the future state are stored in the replay memory. To reduce data correlation effects, the agent will randomly choose these samples during the training process. The proposed DDPG technique consists of two primary DNNs:

- An actor, parameterized by ω, produces action in response to a state, such as $a_t = \pi(s_t; \omega)$.
- Given a state, $Q(s_t, a_t; \emptyset)$, a critic parameterized by $\emptyset$ gives the Q-value of the performed action.

The policy parameter ω is updated using the primary actor-network by interacting with the environment using the current state s and action a to produce reward r and the next state s'. The policy parameter $\emptyset$ and the current Q values ($Q(s_t, a_t, \emptyset)$) are updated using the primary critic network. DDPG further employs a target actor, $\pi(s_t; \omega')$, and a target critic, $Q(s_t, a_t; \emptyset')$, to enhance the network training stability, where ω' and $\emptyset'$ are the target actor and target critic parameters, respectively. The values of

parameters ω' and $\emptyset'$ are not copied directly from ω and $\emptyset$, but they've been updated in the following way:

$$\begin{aligned} \omega_i' &\leftarrow \xi\omega_i + (1-\xi)\omega' \\ \emptyset_i' &\leftarrow \xi\emptyset_i + (1-\xi)\emptyset' \end{aligned} \tag{19}$$

Where ξ signifies the update parameters, which are often modest values like 0.1 or 0.01. The actor network updates its parameter ω by reducing the loss function

$$\nabla_\omega J(\omega) \approx \mathrm{E}[\nabla_a Q(s_t, a; \emptyset) \mid_{a=\pi(s_i;\omega)}, \nabla_\omega \pi(s_t; \omega)] \tag{20}$$

In the meantime, the critic is optimized iteratively to minimize the loss function, which is described as

$$\mathcal{L}(\emptyset) = \mathrm{E}_{s_t, a_t}\left[\left(Y_t - Q(s_t, a_t; \emptyset)\right)^2\right] \tag{21}$$

where $s^t = \{s_1^t, s_2^t, \ldots, s_M^t\}$, $a^t = \{a_1^t, a_2^t, \ldots, a_M^t\}$ and $Y_t = R_t + \gamma Q(s_{t+1}, \pi(s_{t+1}; \omega'); \emptyset')$ here $0 \leq \gamma < 1$ is the discount factor.

After analyzing system state s_t, the agent applies action a_t to the environment at time slot t. At the end of the time slot, the agent receives an instant reward R_t, and the system moves to state s_{t+1}. The state, action, reward and next state information (s_t, a_t, R_t, s_{t+1}) is stored in the limited replay memory as an agent's experience. When new data arrives, the oldest sample will be removed if the buffer is full. The actor and critic networks are trained by randomly chosen mini-batches of experiences from the replay memory. Algorithm 1 summarized the content caching mechanism based on DDPG mechanism.

Algorithm 1 DDPG based Caching Algorithm

Input: System model parameters, number of episodes, number of time steps in each episode, replay buffer size, mini-batch size s, learning rates for critic network and actor network, and update rates for the target critic network and target actor network, respectively.

Output: Optimal content placement policy.

```
Initialize the actor network π(s_t; ω) with random weights ω and the critic network Q(s_t, a_t, ∅) with random weights ∅
Initialize the target network π(s_t; ω'), Q(s_t, a_t, ∅') with weights ω' and ∅'
Initialize the replay memory G of each agent as an empty array
for all episode do
    Initialize t = 1;
    Initialize a random process M for action exploration;
    for t ∈ T and o^t ≠ terminal do
        Observe the cache state s_{t,i} of each agent i;
        For each agent i select an action a_{t,i} = π(s_{t,i}; ω) with respect to current policy and exploration noise;
        Execute action a_t, store the received reward r_t and new state s_{t+1} information;
        t = t + 1;
        Store episode (s_{t,i}, a_{t,i}, r_{t,i} | t = {1, ..., T}) for all agents in replay memory. Discard the old samples if it is full;
    end for
    for all i ∈ M do
        Randomly sample a mini batch of S episodes from replay memory {s_{1,i,j}, a_{1,i,j}, r_{1,i,j}, s_{2,i,j}, a_{2,i,j}, r_{2,i,j}, ...} episodes from replay memory G;
        for t = Tto1 do
            Set target network (Y_t = R_t + γQ(s_{t+1}, π(s_{t+1}; ω'); ∅'));
            Minimizing the loss using Eq. (21) and update the critic network;
            Update the actor network policy using the sampled policy gradient Eq. (20);
        end for
    end for
    Update the target networks using Eq. (19);
    Update the cache state s_{t,i} ;
end for
```

5 Performance evaluation

This section employs simulations to evaluate the proposed deep reinforcement learning based cache placement scheme (DDPGCP). Firstly, the details of the simulation environment, performance metrics, and reference algorithms have been presented. In addition, the proposed DDPGCP mechanism's results have been compared to that of reference methods in terms of system parameters, and the simulation studies are thoroughly analyzed.

5.1 Simulation environment

We do our experiment with the following settings in order to assess the effectiveness of the proposed caching technique. We consider a cellular network with a single cell scenario where the cell consists of a macro base station (MBS) and 20 mobile users are distributed randomly across the coverage of the MBS. The mobile users (user devices) move across the network, randomly contact the other mobile users, and communicate using D2D communication in the considered scenario. The Gamma distribution is followed for contact rate among the various devices [17]. The central library contains 600 contents with a size of 40 MB [25, 35], and individual content is encoded into five segments [34]. In each contact, a mobile user can successfully transmit two encoded segments. The cache capacity of an individually mobile user is 0.2GB, accommodating at most five contents. The content popularity follows Zipf distribution [12, 23]. Needed content for the mobile users are served by other mobile users through D2D communications before the deadline 800s, and MBS will serve the failed segments after the deadline. All results of the simulation shown here are the average of 50 runs. The values of the simulation parameters are presented in Table 2.

5.2 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. *Offloading Ratio* The offloading ratio measures the capacity of the base station to offload data by assessing the volume of information offloaded by D2D content delivery to the total proportion of demand data in the cell.
4. *Cache Reward* The reward measures the cumulative long-term reward collected from caching (i.e., Sum of the intermediate reward of all mobile devices) using Eq. (15).

Table 2 Simulation Parameters

Parameters	Values
Simulation area	$500/m \times 500/m$
Number of users	90
Number of contents	600
Number of base stations	15
Content size	(10, 100] MB
The delay between BS and user	(5,25]s
The delay between BSs	20s
The delay between content server and BS	80s
The deadline of the content	(10,30] s
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

5.3 Reference algorithms

In this section, we compare the proposed scheme with the following caching approaches: Most Popular Caching (MPC) [23, 47, 48], Random Caching (RC) [10, 49], Deep Q-Network (DQN) [36] and Greedy Caching Policy (GCP) [17].

1. MPC: Each device caches the most popular content fragments based on user request statistics until the cache is full.
2. RC: A random caching scheme ensures that every device stores data randomly until the cache is full, regardless of user popularity.
3. GCP: [17] proposes greedy mobility aware caching approach to maximise the D2D offloading ratio while considering user mobility. The average number of successfully delivered file segments via D2D communications to the total number of file segments is known as the D2D offloading ratio.
4. DQN: In this scheme the caching policy is determined based on the algorithm used in [36].

The first three cache replacement strategies update the content individually based on popularity, randomly and greedily, whereas the other strategies consider deep reinforcement learning to place the contents. The fourth cache replacement strategy (DQN) is different from the proposed strategy (DDPGCP) because, the former is the value-based RL, and the latter is policy-based RL.

5.4 Demand model

We use the real-world Dataset MovieLens 1M Dataset [50] in our simulations to investigate for requesting content. The MovieLens dataset consists of 3952 movies, 1000209 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 [51]. We consider the rating

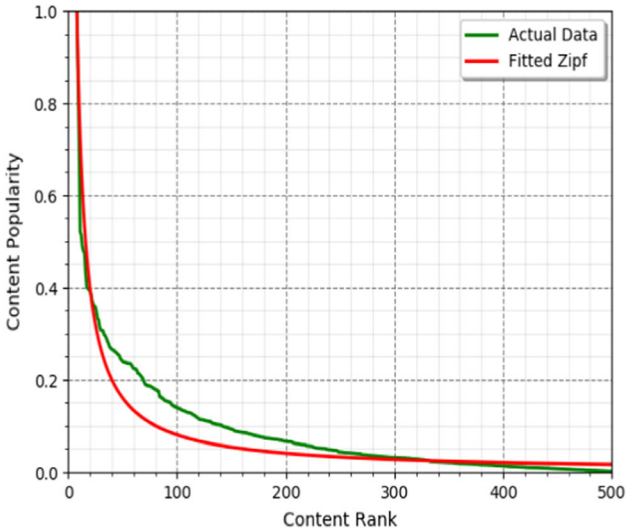


Fig. 4 a Comparison of content popularity vs content rank of Movielens dataset

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 analyze 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.

This simulation runs the proposed algorithms on a desktop with a dual-core Intel i5-5200U 3.20 GHz, and 8 GB installed RAM. The simulation environment was created using the Python 3.9.9 programming language. We conducted comprehensive experiments to compare the proposed caching mechanism to existing caching techniques across various parameters.

Four scenarios are considered to demonstrate the performance of the DDPGCP. In scenario 1, the number of contents is 600, the deadline considered is 800s, the Zipf parameter is 0.6, and the cache capacity ranges from 0.2GB and the number of user devices varies from 10 to 50 with step size 10. In scenario 2, the number of users is 20, the number of contents is 600, the deadline considered is 800s, the Zipf parameter is 0.6, the cache capacity ranges from 0.1 to 0.3GB with step size 0.05GB. In scenario 3, the number of users is 20, the deadline considered is 800s, the Zipf parameter is 0.6, the cache capacity is 0.2GB, and the number of contents varies from 200 to 1000 with step size 200. In scenario 4, The number of users is 20, the number of contents is 600, the Zipf parameter is 0.6, the cache capacity is 0.2GB, and the content deadline varies from 500 to 1000 with step size 100.

5.5 Impact of number of user devices

We show the impact of number of users on cache hit ratio, offloading ratio and acceleration ratio in Fig. 5. The number of contents is 600, the deadline considered is 800s, the Zipf parameter is 0.6, the cache capacity is 0.2GB and the number of user devices varies from 10 to 50 with step size 10.

We can see the effect of the cache hit ratio with a varying number of MECs in Fig. 5a. The cache hit ratio increases as the number of mobile users grows. The reason behind this is that as the number of mobile devices grows, the network's overall cache storage grows, providing mobile users additional opportunities to obtain necessary information via D2D interactions. Due to the utilization of user mobility information, both the mobility aware caching approaches outperform conventional caching techniques. Additionally, the performance disparity among the proposed caching algorithms and other caching strategies widens when mobile devices grow. As mobile devices evolve, user mobility information becomes critical for caching strategy design. The proposed DDPGCP mechanism provide improvement of up to 23, 15, 11 and 6.4 % on hit ratio compared with RC, MPC, GCP and DQN, respectively.

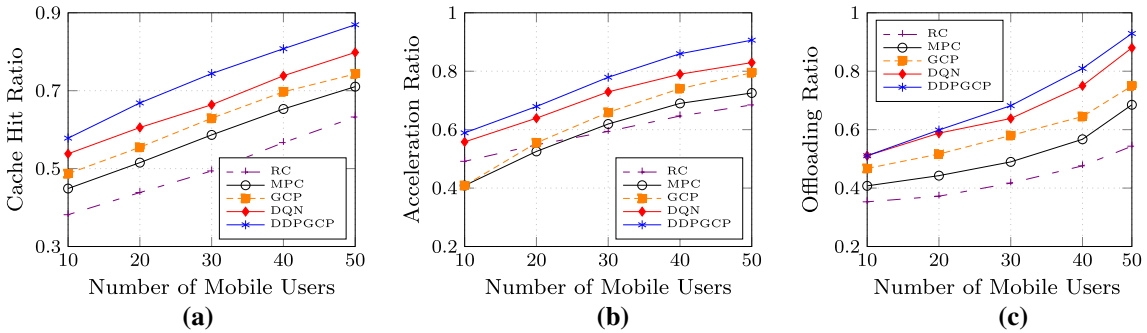


Fig. 5 Comparison of the proposed and existing schemes using number of mobile users vs **a** Cache hit ratio **b** Acceleration ratio **c** D2D offloading ratio

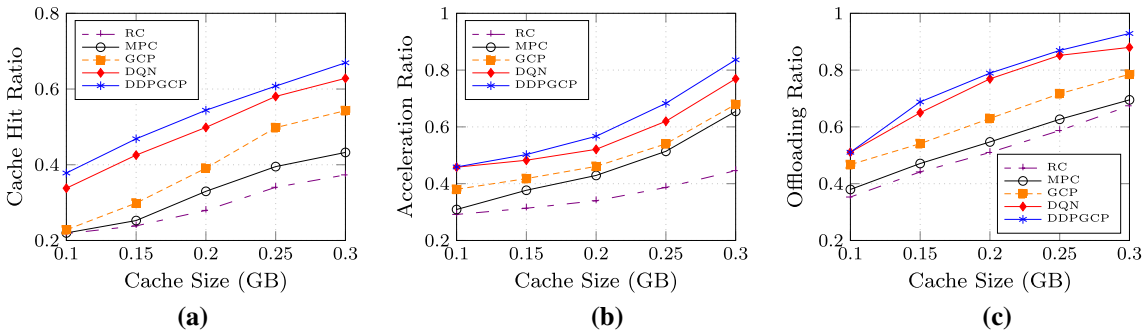


Fig. 6 Comparison of the proposed and existing schemes using cache size vs **a** Cache hit ratio **b** Acceleration ratio **c** D2D offloading ratio

In Fig. 5b, the impact of the number of user devices on the acceleration ratio is presented. The acceleration ratio displays an upward trend with the growing number of users. This is because the increasing number of users cumulatively provide more cache storage and enough space to cache more appropriate content at the user devices leads to serving the content near to users. We can notice that the performance gap between the DRL based schemes, namely DDPGCP and DQN, increases with the increasing user devices. The growth of the curves became low with the higher number of users since the popular content is already cached at the nearby user devices. Most importantly, the proposed mechanism achieves a higher acceleration ratio than other caching schemes due to the utilization of mobility information. The proposed DDPGCP mechanism provide improvement of up to 17, 17, 13 and 5.3 % on acceleration ratio compared with RC, MPC, GCP and DQN, respectively.

In Fig. 5c, the impact of the number of mobile devices on offloading ratio is presented. The proposed caching scheme outperforms other caching methods because it exploits the user mobility information. With the increasing number of users, the performance gap between the proposed mechanism and other caching mechanisms is wider than DQN. This shows that more users make more room to cache appropriate content by utilizing the mobility information. The proposed DDPGCP mechanism provide improvement of up to 27, 18.6, 11.4 and 3.2 % on D2D offloading ratio compared with RC, MPC, GCP and DQN, respectively.

5.6 Impact of cache size of user devices

The impact of cache size on hit ratio, D2D offload ratio, and acceleration ratio is presented in this subsection (Fig. 6). The number of users in this scenario is 20, the number of contents is 600, the deadline considered is 800s, the Zipf parameter is 0.6, and the cache capacity ranges from 0.1 to 0.3GB with step size 0.05GB.

In Fig. 6a, the impact of cache size on the cache hit rate is shown. The cache hit rate displays an upward trend with the expansion of the cache size. The rationale for this is that mobile users will be able to cache more contents with higher storage sizes, which will be successfully shared via D2D connections, leading to faster content delivery. We can notice that the proposed mechanism outperforms other caching schemes by achieving a higher cache hit ratio. The DDPGCP and DQN mechanism shows superiority over the other caching schemes due to the mobility information. The proposed DDPGCP mechanism provide improvement of up to 24.3, 20.7, 14 and 3.9 % on hit ratio compared with RC, MPC, GCP and DQN, respectively.

In Fig. 6b, the impact of cache size on the acceleration ratio is presented. The acceleration ratio evolves higher with the larger cache sizes. With the increase in the cache capacity, all the mechanisms increase quickly compared to lower cache sizes. This is because of getting more storage to accommodate more popular content at each user device. The proposed DDPGCP outperforms other existing schemes with significant improvement in the acceleration ratio. By utilizing the mobility information, DQN and DDPGCP gain the acceleration ratio than other schemes. The proposed DDPGCP mechanism provide improvement of up to 24.8, 14.6, 11 and 4 % on acceleration ratio compared with RC, MPC, GCP and DQN, respectively.

In Fig. 6c, the impact of the cache size on the offloading ratio is presented. As expected, the D2D offloading rate of all caching mechanisms increases as cache size increases. The DRL based caching schemes, namely DDPGCP and DQN, produce a greater offloading rate than existing cache approaches. The reason is that exploiting device contact information of the mobility-based cache approaches provides an advantage over the non-mobility based caching schemes. The proposed mechanism has a significant advantage with the limited cache size since it exploits the user contact information and transmission rate. The proposed DDPGCP mechanism provide improvement of up to 24.4, 21, 13 and 2.5 % on D2D offloading ratio compared with RC, MPC, GCP and DQN, respectively.

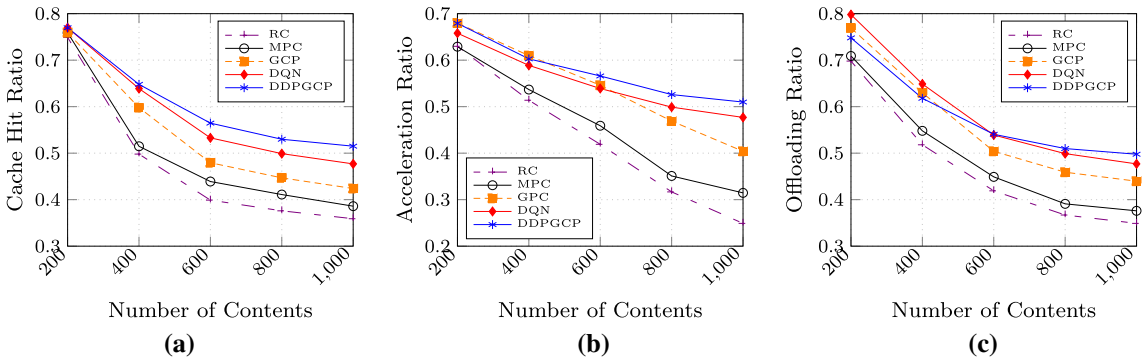


Fig. 7 Comparison of the proposed and existing schemes using number of contents vs **a** Cache hit ratio **b** Acceleration ratio **c** D2D offloading ratio

5.7 Impact of number of contents

We show the impact of number of contents on cache hit ratio, offloading ratio and acceleration ratio in Fig. 7. The number of users is 20, the deadline considered is 800s, the Zipf parameter is 0.6, the cache capacity is 0.2GB and the number of contents varies from 200 to 1000 with step size 200.

In Fig. 7a, the impact of number of contents on the cache hit ratio is presented. The bars indicate an downward trend as contents rise. The user attention may be more distributed when there is more content. Unfortunately, the cache size of caching users stays unchanged and caching users cannot store such a vast quantity of content, making the requesters' demands extremely challenging to satisfy. Random caching and most popular schemes perform poorly compared to other methods. The quantity of available content grows, the allocation of popular content becomes more scattered, and consumer preferences for content become more divided. However, the proposed caching approach outperforms other caching schemes. The proposed DDPGCP mechanism provide improvement of up to 13, 10.3, 6 and 2 % on hit ratio compared with RC, MPC, GCP and DQN, respectively.

In Fig. 7b, the impact of the number of contents on the acceleration ratio is presented. With the volume of the increasing content, the acceleration ratio decreases. The RC and MPC are declining more than other caching approaches. The GCP has a better performance with less volume of content. As the volume of content increases, the gain of the GCP also starts decreasing, the performance gap between the DRL based schemes is widening. Exploiting the mobility information makes the DQN and DDPGCP mechanism serve the user demanded content at the appropriate node. The proposed DDPGCP mechanism provide improvement of up to 15, 11.8, 3 and 2.4 % on acceleration ratio compared with RC, MPC, GCP and DQN, respectively.

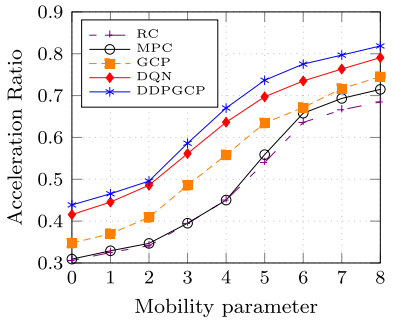


Fig. 8 Comparison of the proposed and existing schemes using user mobility vs Acceleration ratio

In Fig. 7c, the impact of the number of contents on the offloading ratio is presented. The downward trend of the curves indicates that with the growing volume of content, the contents' diversity rises. We can notice that DDPGCP shows similar performance with GCP with fewer contents. The ever increasing contents allow DDPGCP to cache the popular content based on user preferences at suitable user nodes, improving the performance gap between GCP and DDPGCP. The proposed DDPGCP mechanism provide improvement of up to 11.7, 9, 2.7 and 0.4 % on D2D offloading ratio compared with RC, MPC, GCP and DQN, respectively.

5.8 Impact of mobility

We shown the impact of the user mobility on acceleration ratio in Fig. 8. Every mobile user pair's contact rate is generated based on the gamma distribution $\Gamma(4.43, 1/1088)$ [17]. The different average mobile device speed is generated using $\Gamma(4.43\theta^2, 1/1088\theta)$. We notice that the acceleration ratio first decreases and raises slowly with the increasing contact rate. The lower θ value represents the lower moving speed, which results in a higher contact duration but a lower number of contacts. This leads to getting a higher number of segments with fewer

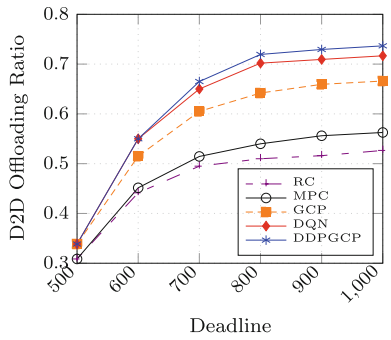


Fig. 9 Comparison of the proposed and existing schemes using deadline vs D2D offloading ratio

contacted user devices, hence caching the most popular content at the user devices. Also, the higher θ values mean user devices are moving at a fast pace, resulting in more contact with less contact time, leading to cache most popular contents to address most of the user demands. The medium θ values show that the users are moving with moderate speed, leading to cache more diverse content at each user device to minimize duplicate contents at the entire network. As a result, we may deduce that user movement knowledge is critical in developing caching strategies, particularly in high mobility scenarios. The proposed mechanism achieves better acceleration than other caching schemes. The proposed DDPGCP mechanism provide improvement of up to 16, 14.75, 9.4 and 2.8 % on acceleration ratio compared with RC, MPC, GCP and DQN, respectively.

5.9 Impact of deadline

We show the impact of content deadline on offloading ratio in Fig. 9. The number of users is 20, the number of contents is 600, the Zipf parameter is 0.6, the cache capacity is 0.2GB and the content deadline varies from 500 to 1000 with step size 100. The proposed DDPGCP mechanism outperforms other caching schemes. As the deadline evolves larger, it allows user devices to disseminate the content through D2D connections and gives extra time to move across the region. We can notice that the proposed caching mechanism performs similar to the DQN caching schemes with a smaller deadline, and the performance gap is widening with the larger deadlines. Since the shorter deadline produces insufficient user interactions among the user devices to send the entire content for the maximum number of contents leads to delivering only small portions of the content from various contents. The larger deadlines allow the user devices to deliver the entire content successfully. The proposed DDPGCP mechanism provide improvement of up to 14.6, 12.6, 4.8 and 1 % on D2D offloading ratio compared with RC, MPC, GCP and DQN, respectively.

5.10 The convergence performance

We show the convergence performance of the DDPGCP and DQN schemes in Fig. 10. We can observe that the proposed DDPGCP scheme has a higher reward than the DQN scheme. The reason is that DQN learns the Q values utilized to determine the policy, which is often an epsilon-greedy policy, whereas DDPG tries to learn the policy directly. Hence, the proposed mechanism has higher convergence performance than other schemes. We can see that the initial episodes reward is higher for DQN, and as the number of episodes increases, DDPGCP converges much faster.

We show the convergence performance of the DDPGCP schemes with different learning rates in Fig. 11. After 400 training episodes, the average rewards for four learning rate values grow and start to converge. Specifically, with a low learning rate, the proposed method converges slowly, and as the learning rate grows, the proposed method converges quickly. However, when the learning rate is equal to 0.1, the presented method may converge to locally optimal values; therefore, raising the learning rate may not yield better results. We can notice that the learning rates 0.1 and 0.01 are converged around 400 episodes. In our simulations, the proposed technique can give the highest rewards at a learning rate of 0.01. As a result, in the current simulations, we set the learning rate to 0.01.

In Fig. 12, the performance comparison of cache hit ratio and acceleration ratio is presented. In this simulations, the number of users are 20, skewness parameter is 0.6, the deadline is 800s, number of contents is 600 and the cache capacity is 0.2GB.

From Fig. 12a, 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 deep reinforcement

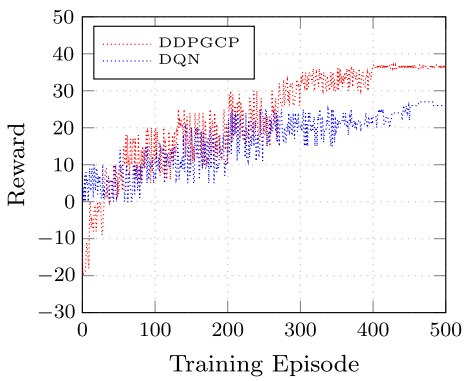


Fig. 10 Convergence of proposed scheme (DDPGCP) and DQN

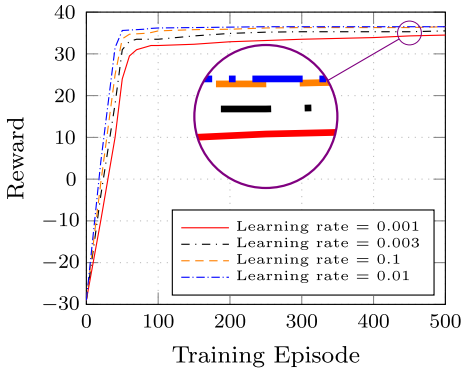


Fig. 11 Convergence of proposed scheme (DDPGCP) with different learning rates

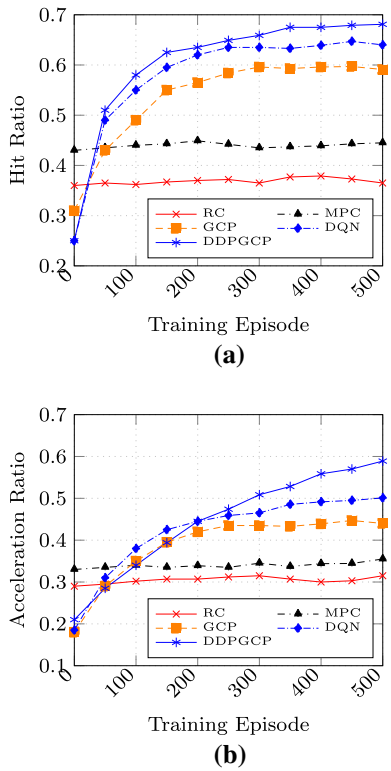


Fig. 12 Comparison of the proposed and existing schemes using training episode vs **a** Cache hit ratio **b** Acceleration ratio

learning-based algorithms curves indicate an upward trend and stabilize after that. The GCP has a higher hit ratio than DQN and DDPGCP initially, but as the episodes increase, it slowly diminishes. That is because the GCP is a greedy cooperative cache replacement mechanism where each user greedily caches the content based on local information, not considering the other agents' knowledge in caching decisions. Therefore, each agent may cache content redundantly leads to obtain more content from the content server. In DQN, the agents cache the content based on the central controller, which cooperates with communication overhead. The proposed DDPGCP outperforms the other mechanisms since it directly learns instead of learning from the determined policy to cache more popular content.

In Fig. 12b, 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. We can notice that the proposed mechanism has a lower acceleration ratio than GPC and DQN, and the performance gap is widening with increasing training. The proposed mechanism raises slowly but achieves a better acceleration ratio than other caching schemes. That means the DDPGCP caches more popular content with increasing training, leading to more saved delays. The DQN and GCP have a lower acceleration ratio since, in DRL, agents learn and update policy determined by the centralized critic without cooperation. GCP considers the mobility of the users into the caching decisions; hence it achieves a better acceleration ratio, but the lack of learning mechanism it caches the redundant copies of the content at the users leads to a lower acceleration ratio than DQN and DDPGCP.

6 Conclusion

In this paper, the cooperative cache placement problem has been analyzed in device-to-device mobile edge networks by placing the content to maximize the saved delay with deadline and capacity constraints. The saving latency has been calculated analytically using the inter-contact movement pattern. We formulate the problem as an Integer linear programming problem for cooperative cache placement. Since the proposed problem is NP-hard, we designed a caching scheme for large-sized D2D enabled mobile edge networks by integrating a deep reinforcement learning based deep deterministic policy gradient mechanism to enhance the long term reward and speed up the learning process. It has been demonstrated that exploiting user cooperation, content deadlines, and randomness of user interaction information results in a considerable performance gain in D2D enabled mobile edge networks. The simulation results show that the proposed cooperative cache placement improves up to 23, 24 and 25 per cent on cache hit rate, acceleration ratio and offload ratio compared with RC, MPC, GCP and DQN, respectively. For future work, we investigate the optimized user association and D2D link quality to improve the user quality of experience.

Data availibility The *MovieLens 1M* dataset was used to support this study and its application details are available in https://grouplens.org/datasets/movielens/1m/. The data set is cited at relevant places within the text as references.

Code availability The program of this paper is supported by custom code. It can be applied from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they do not have any known competing interest.

Ethical statement The work submitted by the authors is his own work and it is neither published nor considered for publication elsewhere.

References

1. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., & Chen, M. (2019). In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network, 33*(5), 156–165.
2. Inc, C. S. (2019). Cisco visual networking index: Global mobile data traffic forecast update. *Update, 2017,* 2022.
3. Yao, J., Han, T., & Ansari, N. (2019). On mobile edge caching. *IEEE Communications Surveys & Tutorials, 21*(3), 2525–2553.
4. Qiu, L., & Cao, G. (2019). Popularity-aware caching increases the capacity of wireless networks. *IEEE Transactions on Mobile Computing, 19*(1), 173–187.
5. Pan, Y., Pan, C., Yang, Z., Chen, M., & Wang, J. (2019). A caching strategy towards maximal d2d assisted offloading gain. *IEEE Transactions on Mobile Computing, 19*(11), 2489–2504.
6. Prerna, D., Tekchandani, R., & Kumar, N. (2020). Device-to-device content caching techniques in 5g: A taxonomy, solutions, and challenges. *Computer Communications, 153,* 48–84.
7. Yu, S., Dab, B., Movahedi, Z., Langar, R., & Wang, L. (2019). A socially-aware hybrid computation offloading framework for multi-access edge computing. *IEEE Transactions on Mobile Computing, 19*(6), 1247–1259.
8. Tran, T. X., Le, D. V., Yue, G., & Pompili, D. (2018). Cooperative hierarchical caching and request scheduling in a cloud radio access network. *IEEE Transactions on Mobile Computing, 17*(12), 2729–2743.
9. Li, L., Zhao, G., & Blum, R. S. (2018). A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies. *IEEE Communications Surveys & Tutorials, 20*(3), 1710–1732.
10. Chen, M., Saad, W., Yin, C., & Debbah, M. (2017). Echo state networks for proactive caching in cloud-based radio access networks with mobile users. *IEEE Transactions on Wireless Communications, 16*(6), 3520–3535.
11. Zhu, H., Cao, Y., Wang, W., Jiang, T., & Jin, S. (2018). Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Network, 32*(6), 50–57.
12. Shanmugam, K., Golrezaei, N., Dimakis, A. G., Molisch, A. F., & Caire, G. (2013). Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory, 59*(12), 8402–8413.
13. Li, J., Liu, M., Lu, J., Shu, F., Zhang, Y., Bayat, S., & Jayakody, D. N. K. (2019). On social-aware content caching for d2d-enabled cellular networks with matching theory. *IEEE Internet of Things Journal, 6*(1), 297–310.
14. Yang, C., & Stoleru, R. (2020). Ceo: cost-aware energy efficient mobile data offloading via opportunistic communication. In 2020 International Conference on Computing (pp. 548–554). IEEE: Networking and Communications (ICNC).
15. Dai, X., Xiao, Z., Jiang, H., Alazab, M., Lui, J., Dustar, S., & Liu, J. (2022). Task co-offloading for d2d-assisted mobile edge computing in industrial internet of things. IEEE Transactions on Industrial Informatics
16. Wang, Z., Shah-Mansouri, H., & Wong, V. W. (2016). How to download more data from neighbors? a metric for d2d data offloading opportunity. *IEEE Transactions on Mobile Computing, 16*(6), 1658–1675.
17. Wang, R., Zhang, J., Song, S., & Letaief, K. B. (2017). Mobility-aware caching in d2d networks. *IEEE Transactions on Wireless Communications, 16*(8), 5001–5015.
18. Qiao, J., He, Y., & Shen, X. S. (2016). Proactive caching for mobile video streaming in millimeter wave 5g networks. *IEEE Transactions on Wireless Communications, 15*(10), 7187–7198.
19. Lu, Z., Sun, X., & La Porta, T. (2016). Cooperative data offloading in opportunistic mobile networks. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE, pp 1–9
20. Zhou, H., Wang, H., Li, X., & Leung, V. C. (2018). A survey on mobile data offloading technologies. *IEEE Access, 6,* 5101–5111.
21. Poularakis, K., Iosifidis, G., & Tassiulas, L. (2014). Approximation algorithms for mobile data caching in small cell networks. *IEEE Transactions on Communications, 62*(10), 3665–3677.
22. Baştuğ, E., Kountouris, M., Bennis, M., Debbah, M. (2016). 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, pp 1–5
23. Somesula, M. K., Rout, R. R., & Somayajulu, D. (2021). Contact duration-aware cooperative cache placement using genetic algorithm for mobile edge networks. *Computer Networks, 193,* 108062.
24. Bharath, B., Nagananda, K. G., Gündüz, D., & Poor, H. V. (2018). Caching with time-varying popularity profiles: A learning-theoretic perspective. *IEEE Transactions on Communications, 66*(9), 3837–3847.
25. Somesula, M. K., Rout, R. R., & Somayajulu, D. (2021). Deadline-aware caching using echo state network integrated fuzzy logic for mobile edge networks. *Wireless Networks, 27*(4), 2409–2429.
26. Wang, X., Zhang, Y., Leung, V. C., Guizani, N., & Jiang, T. (2018). D2d big data: Content deliveries over wireless device-to-device sharing in large-scale mobile networks. *IEEE Wireless Communications, 25*(1), 32–38.
27. Wu, D., Zhou, L., Cai, Y., & Qian, Y. (2018). Collaborative caching and matching for d2d content sharing. *IEEE wireless communications, 25*(3), 43–49.
28. Liu, Z., Song, H., & Pan, D. (2020). Distributed video content caching policy with deep learning approaches for d2d communication. *IEEE Transactions on Vehicular Technology, 69*(12), 15644–15655.
29. Fu, Y., Salaün, L., Yang, X., Wen, W., & Quek, T. Q. (2021). Caching efficiency maximization for device-to-device communication networks: A recommend to cache approach. *IEEE Transactions on Wireless Communications, 20*(10), 6580–6594.
30. Zhao, D., Wang, H., Shao, K., & Zhu, Y. (2016) Deep reinforcement learning with experience replay based on sarsa. In 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp 1–6
31. Zhang, W., Wu, D., Yang, W., & Cai, Y. (2019). Caching on the move: A user interest-driven caching strategy for d2d content sharing. *IEEE Transactions on Vehicular Technology, 68*(3), 2958–2971.
32. Zhang, S., Quan, W., Li, J., Shi, W., Yang, P., & Shen, X. (2018). Air-ground integrated vehicular network slicing with content

pushing and caching. *IEEE Journal on Selected Areas in Communications, 36*(9), 2114–2127.
33. Ibrahim, A. M., Zewail, A. A., & Yener, A. (2020). Device-to-device coded-caching with distinct cache sizes. *IEEE Transactions on Communications, 68*(5), 2748–2762.
34. Sun, R., Wang, Y., Lyu, L., Cheng, N., Zhang, S., Yang, T., & Shen, X. (2020). Delay-oriented caching strategies in d2d mobile networks. *IEEE Transactions on Vehicular Technology, 69*(8), 8529–8541.
35. Poularakis, K., & Tassiulas, L. (2016). Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks. *IEEE Transactions on Mobile Computing, 16*(3), 675–687.
36. Zhou, H., Wu, T., Zhang, H., & Wu, J. (2021). Incentive-driven deep reinforcement learning for content caching and d2d offloading. *IEEE Journal on Selected Areas in Communications, 39*(8), 2445–2460.
37. Bajpai, R., Chakraborty, S., Gupta, N. (2022). Adapting deep learning for content caching frameworks in device-to-device environments. IEEE Open Journal of the Communications Society
38. Qiu, X., Liu, L., Chen, W., Hong, Z., & Zheng, Z. (2019). Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing. *IEEE Transactions on Vehicular Technology, 68*(8), 8050–8062.
39. Somesula, M. K., Rout, R. R., & Somayajulu, D. V. (2022). Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks. *Computer Networks, 209,* 108876.
40. He, Y., Liang, C., Yu, F.R., Leung, V.C. (2018). Integrated computing, caching, and communication for trust-based social networks: A big data drl approach. In 2018 IEEE Global Communications Conference (GLOBECOM), IEEE, pp 1–6
41. Zeng, S., Ren, Y., Wang, Y., Zhao, T., Qian, Z. (2019). Caching strategy based on deep q-learning in device-to-device scenario. In 2019 12th International Symposium on Computational Intelligence and Design (ISCID), IEEE, vol 1, pp 175–179
42. Jiang, W., Feng, G., Qin, S., Yum, T. S. P., & Cao, G. (2019). Multi-agent reinforcement learning for efficient content caching in mobile d2d networks. *IEEE Transactions on Wireless Communications, 18*(3), 1610–1622.
43. Li, L., Xu, Y., Yin, J., Liang, W., Li, X., Chen, W., & Han, Z. (2019). Deep reinforcement learning approaches for content caching in cache-enabled d2d networks. *IEEE Internet of Things Journal, 7*(1), 544–557.
44. Chakraborty, S., Bajpai, R., & Gupta, N. (2021). R2-d2d: A novel deep learning based content-caching framework for d2d networks. In 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), IEEE, pp 1–5
45. Jiang, W., Feng, G., & Qin, S. (2017). Optimal cooperative content caching and delivery policy for heterogeneous cellular networks. *IEEE Transactions on Mobile Computing, 16*(5), 1382–1393.
46. Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine, 34*(6), 26–38.
47. Ahlehagh, H., & Dey, S. (2014). Video-aware scheduling and caching in the radio access network. *IEEE/ACM Transactions on Networking (TON), 22*(5), 1444–1462.
48. Peng, X., Shen, J.C., Zhang, J., & Letaief, K.B. (2015). Backhaul-aware caching placement for wireless networks. arXiv preprint arXiv:1509.00558
49. Blaszczyszyn, B., & Giovanidis, A. (2015). Optimal geographic caching in cellular networks. In 2015 IEEE International Conference on Communications (ICC), IEEE, pp 3358–3363
50. Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems, 5*(4), 1–19. https://doi.org/10.1145/2827872.
51. Garg, N., Sellathurai, M., Bhatia, V., Bharath, B., & Ratnarajah, T. (2019). Online content popularity prediction and learning in wireless edge caching. *IEEE Transactions on Communications, 68*(2), 1087–1100.

Manoj Kumar Somesula Manoj Kumar Somesula received his PhD from National Institute of Technology Warangal, India. He completed M. Tech. degree in Computer Science from School of Information Technology, JNTU, Hyderabad, India and B. Tech. degree in Computer Science and Engineering from Mahaveer Institute of Science and Technology, JNTU, Hyderabad, India. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering at BVRIT Hyderabad College of Engineering for Women, India. He has published several research papers on mobile edge networks and D2D networks in reputed international journals. His primary research area includes Edge computing, Wireless networks, Internet of Things and Unmanned aerial vehicles. He is a member of IEEE, IEEE ComSoc, IEEE Computer Society and ACM.

Sai Krishna Mothku Sai Krishna Mothku received the B. Tech degree in Computer Science and Engineering in 2009 from Kakatiya Institute of Technology and Science (KITS), Warangal, Telangana, India and M. Tech degree in Computer Science and Engineering - Information Security in 2012 from the National Institute of Technology, Surathkal, Karnataka, India. He received PhD degree from the National Institute of Technology, Warangal, Telangana, India in 2019. Currently, he is working as an Assistant Professor in the Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, India. His primary research includes Wireless adhoc networks, Wireless sensor networks, Internet of things, Cloud, Fog and Edge computing.

Anusha Kotte Anusha Kotte received M. Tech degree in Computer Science from School of Information Technology, JNTU Hyderabad, India and B. Tech. degree in Computer Science and Engineering from Mahaveer Institute of Science and Technology, JNTU, Hyderabad, India. She is currently pursuing her PhD in the Department of Computer Science and Engineering at JNTU Hyderabad, India. Her research interest includes Wireless networks, Machine learning and Deep learning.