UAV Assisted Cooperative Caching on Network Edge Using Multi-Agent Actor-Critic Reinforcement Learning

Sadman Araf, Adittya Soukarjya Saha, Sadia Hamid Kazi, Nguyen H. Tran, *Senior Member, IEEE*, and Md. Golam Rabiul Alam, *Member, IEEE*

***Abstract*—In recent times, caching at edge nodes is a well-known technique to overcome the limitation of strict latency, which simultaneously improves users' Quality of Experience (QoE). However, choosing an appropriate caching policy and content placement poses another significant issue that has been acknowledged in this research. Conventional caching policies that are experimented with at the edge do not consider the dynamic and stochastic characteristics of edge caching. As a result, we have proposed a cooperative deep reinforcement learning algorithm that deals with the dynamic nature of content demand. It also ensures efficient use of storage through the cooperation between nodes. In addition, previous works on cooperative caching have assumed the users to be static and didn't consider the mobile nature of users. Therefore, we have proposed UAVs as aerial Base Stations (UAV-BS) to assist in peak hours where a ground base station is insufficient to support the surge in user requests. In this novel research, we have demonstrated the cooperation between aerial and Ground Base Stations (GBS) and aimed at maximizing the global cache hit ratio. Simulations have shown that our proposed Cooperative Multi-Agent Actor-Critic algorithm outperforms conventional and reinforcement learning based caching methods and achieves a state-of-the-art global cache hit ratio when there is a surge in user requests. Thus, it opens the door for further research on cooperative caching in joint air and ground architecture.**

***Index Terms*—Cooperative edge caching, multi-acccess edge computing, multi-agent actor-critic, reinforcement learning, unmanned aerial vehicle (UAV).**

I. Introduction

The launching of the Internet of Things (IoT) and the upcoming 6G technologies forecast data to rise rapidly in the next couple of years in comparison to the last ten years. At present, the cloud tackles most of the data, and cloud traffic was around 19.5 ZB by the end of 2021, according to the Cisco global cloud index as mentioned in [1]. As per research conducted by Cisco Visual Networking Index [2], by the year 2023, there is likely to be 5.3B global internet users, 29.3B network devices, connections, and 66% of the total population in the world will have internet access.

Due to such a plethora of interconnected devices, the load of internet traffic will be exceedingly higher to manage and will eventually lead to network latency and delay [3], [4], [5], [6]. On the other hand, the number of connected devices has increased. Thus, almost all new technologies that have been gone through innovation so far require real-time processing. Since most people are more or less connected to infotainment systems on the go, high volumes of data need to be brought to the users in real-time based on the position of the users.

To alleviate this challenge with massive data traffic, a favorable approach has evolved, named Mobile Edge Computing or Multi-access Edge Computing (MEC). It is a distributed framework that brings data computation closer to the source device. The practice of using MEC is not restricted to any particular field as there are several major application areas in the offloading of computation, followed by the distribution of content delivery, caching, as well as in the enhancements of web performance [7].

Caching on edge servers is a widely used application researched consistently nowadays. For efficient edge caching, different approaches have been considered. In [6], deep learning models are used to predict the future popular contents that will cache in the base station. To bring about an efficient caching decision, deep learning models are trained and used in the cloud data center. Finally, a caching decision was sent to each base station which will then cache accordingly. There have been many proactive caching algorithms that cache popular content in base stations and user equipment during off-peak hours, but such techniques do not adapt to changing popularity and are not feasible when the contents are on a large scale. As a result, Deep Reinforcement Learning (DRL) is an appropriate alternative to tackle this issue. A DRL agent can be deployed at local servers and dynamically adapt to changing conditions. According to [8], the policy control task can also be carried out by DRL, i.e., finding out which content to store in the storage.

Manuscript received 24 February 2022; revised 16 July 2022; accepted 25 August 2022. Date of publication 23 September 2022; date of current version 13 February 2023. This work was supported by the School of Computer Science, The University of Sydney, Sydney, Australia. The work of Nguyen H. Tran's was supported by SOAR Prize. The review of this article was coordinated by Prof. Daniele Tarchi. *(Corresponding author: Md. Golam Rabiul Alam.)*

Sadman Araf, Adittya Soukarjya Saha, Sadia Hamid Kazi, and Md. Golam Rabiul Alam are with the Department of Computer Science and Engineering, Brac University, Dhaka 1212, Bangladesh (e-mail: sadman.araf@g.bracu.ac.bd; adittya.soukarjya.saha@g.bracu.ac.bd; skazi@bracu.ac.bd; rabiul.alam@bracu.ac.bd).

Nguyen H. Tran is with the School of Computer Science, Faculty of Engineering, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: nguyen.tran@sydney.edu.au).

Digital Object Identifier 10.1109/TVT.2022.3209079

But caching contents without cooperation with nearby servers will lead to inefficient caching, as both the servers may store the same content. As a result, cooperative caching is also quite significant while implementing DRL in edge networks. Ensuring cooperation should avoid duplicate data caching and fully utilize the computational capacity of each server.

On the other hand, if we consider the end-users who are usually mobile in nature, edge servers cannot always serve them as they are fixed and cannot be available in all locations. Therefore, we require an alternative base station that can provide more data to users by making them available at user locations. As a result, authors in [9] have considered using Unmanned Aerial Vehicles (UAVs) to cache data that meet user demands at specific locations.

A. Research Contribution

In this novel research, we have proposed a context-aware caching policy through a cooperative Deep Reinforcement Learning (DRL)-based algorithm. The proposed algorithm intelligently analyzes context information and complex user request patterns. The base station nodes will be more aware of their environment and make the right caching policy to maximize caching performance. Next, we incorporated Unmanned Aerial Vehicles (UAVs), which will act as flying base stations. We have created a hierarchy of base stations where MEC servers will be acting as a Ground Base Station (GBS), and UAVs as aerial Base Station (UAV-BS). Thirdly, we optimized the UAV's positioning by implementing a clustering-based algorithm which ensured that the UAVs will dynamically reallocate themselves and analyze the mobility patterns of wireless users to serve them efficiently. It is noteworthy to mention we have been inspired by the following works in [10] and [11].

The main contributions of this work are as follows:

- We have designed a three-layer architecture where we have deployed UAVs as aerial Base Stations (BS) to cache contents in peak hours alongside Ground Base Stations. Both the GBS and UAV caches contents dynamically. To support the caching for mobile users the UAVs will reallocate themselves according to the user mobility. If the cached file is not found from either GBSs or UAVs, then the file is retrieved from the cloud data center.
- A novel cooperative multi-agent actor-critic-based reinforcement learning approach has been designed for the caching model that considers the signal-to-noise ratio to ensure QoE and achieves a significantly greater global cache hit ratio by mitigating duplication of contents. To reduce the complexity caused by the vast action space, we have considered a discrete action space that would simply output a caching policy instead.
- We have formulated the optimization function as the maximization of the global cache hit ratio and signal-to-noise ratio.
- To address mobile users, we have placed UAVs in public congregation areas. Here we have implemented a clustering algorithm where the UAVs have been considered as centroids which will reallocate itself according to the user mobility in each period to cater to the surge in user requests.
- To demonstrate the performance of our proposed algorithm, we have shown a global cache hit ratio comparison amongst conventional and state-of-the-art caching models in peak hours where the user request is high. Global cache hit ratio for varying storage capacities has also been studied.

The remainder of the paper is structured as follows. We proposed the Related Works in Section II followed by System Model in Section III and Cache Node Selection in Section IV respectively. In Section V, we elaborate on the multi-agent actor-critic reinforcement learning algorithm. In Section VI, we evaluate the performance of our proposed caching algorithm by simulation and finally conclude the paper in Section VII.

II. Related Works

To conduct a research on MEC, several constraints need to be considered before designing the applications. In [12], the authors provide a detailed investigation of technical challenges and limitations of Mobile Edge Computing by identifying, discussing, and classifying the different types of applications for the deployment at the mobile edge. In [13], the authors carried out a theoretical analysis that considered the computational and communication models in MEC. Moreover, they also tried to come up with the identification of challenges and opportunities that MECs had from the viewpoint of communication. In [3], [4], the authors addressed the challenges of joint caching, computation, and communication (3C) and joint caching, computation, communication, and control for Big Data processing (4C) in MEC. In [3], the authors proposed the problem as a non-linear, mix-integer, and non-convex optimization to minimize the latency and provided a solution using ADMM. In [4] they opted for Big Data as a solution, and the caching functions took place at edge devices instead of the cloud. They used collaborative methods in between MEC servers located in the same cluster to reduce backhaul traffic. Alongside in [14], the authors came up with the idea of an integrated framework that allows dynamic orchestration of networking, caching, and computing resources to upgrade the performance of next-generation vehicular networks. There have been some noteworthy researches on edge caching in the recent past. In [15], the authors made use of proactive caching in two phases, mainly in the offline phase they have evaluated the heuristics and applied the heuristics to make cache management decisions in the online stage. In [5], the authors have proposed a unique caching mechanism of infotainment for close-range driver-less cars. According to their proposed methodology, a self-driving car should be capable of downloading relevant infotainment contents by passengers from edge servers and are cached accordingly. Furthermore, in [16] the authors proposed a mobility-aware proactive caching scheme using Federated Learning (FL). However, synchronized FL falls behind in terms of computational speeds as vehicles switch back to offline frequently.

Furthermore, edge caching can decrease the immense traffic load and the end-to-end latency in Radio Access Networks (RANs). The research regarding the issue of resource allocation policies coded caching and computational offloading strategies in RAN using a multi-user MEC system has been carried out in [17]. A C-RAN which is a dynamic resource allocation

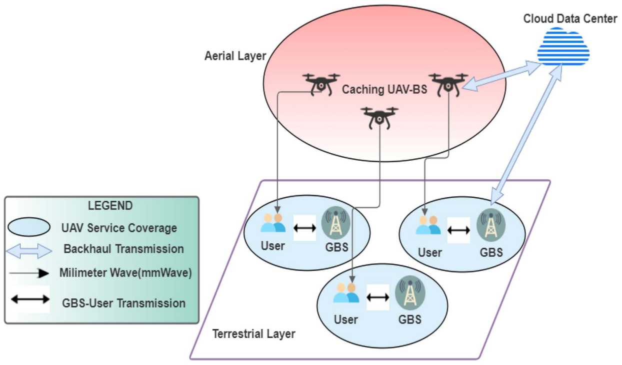

Fig. 1. System model of the joint air-ground architecture for cooperative caching.

framework based on Deep RL (DRL), has been proposed in [18] which reduced time and energy consumption and kept track while making sure that user demands were satisfied.

Alongside, the deployment of UAVs to support edge caching have been proposed by noteworthy researches. In [19], the authors derived an upper bound expression to the coverage probability. They were able to relate that the amount of performance that aerial users were being served to was dependent on several other factors that include the content availability, collaboration distance, and the target bit rate. In [20], they have optimized UAV trajectory and time scheduling to ensure secured transmission for caching users. UAV-BS are more desirable than ground base stations because they can reach high altitude and is successful in producing a high ratio of line-of-sight (LoS) connections toward ground users as stated in [21]. Moreover, since the users' movement is random and unpredictable there needs to be a model to predict the current position of the user requesting popular content. Therefore, the random way-point model has been used as a technique to model the users' position [22].

The research in [23] provided a novel Air-Ground Integrated Mobile Edge Network (AGMEN) architecture that operates through scheduling and deployment of UAVs to assist in caching, communication as well as computing in edge networks. The AGMEN has a two-layer network architecture, i.e., the multi-UAV aerial network and the ground network is a RAN. For transmission of information amongst UAVs, a flying ad hoc network (FANET) is formed, which uses aerial-to-aerial (A2A) communications using heterogeneous radio interfaces and WiFi. Furthermore, authors in [24] discussed multiple Unmanned Aerial Vehicles (UAVs), where they create an aerial network to help the ground vehicular sub-network through A2A and air-to-ground (A2G) communications. They have introduced three kinds of communication links in a multi-UAV-aided vehicular ad-hoc network(VANET) which contains A2A links, V2V links, and A2G links.

To address the dynamic complexities of edge caching several reinforcement learning based caching have been proposed in recent papers. In [25], the authors have demonstrated actor-critic DRL framework along with wolpertinger architecture initially for a single base station. Afterward, implementation for multiple base stations in a multi-agent actor-critic environment for decentralized cooperative caching was proposed. Their analysis of the performance has further demonstrated the capability of the proposed multi-agent framework in recognizing popular content quite efficiently. Authors in [26] emphasize on Device-to-Device (D2D) networks to reduce the load on base stations. The D2D caching problem has been modeled as a multi-agent multi-armed bandit problem, and the Q-learning algorithm is applied to organize the caching decisions. Caching in DRL is usually done by replacing contents that are less frequently requested. Furthermore, wireless caching usually includes high temporal variability of user requests. To be able to meet such time-varying user requests, base stations with limited storage often have to replace their local caches according to different cache replacement policies like- least recently used (LRU) and first-in-first-out (FIFO) as discussed in [27], [28].

Reinforcement learning combined with the deep neural network can solve complicated control problems [29]. To accomplish context-awareness in mobile edge caching, researchers in [30] have used DRL to maximize the long-term system reward. They put forward a thorough analysis of recent papers related to DRL in edge computing. Furthermore, the following papers [31], [32] discuss multi-agent multi-armed bandit caching as a solution when popularity distribution is not known beforehand.

To ensure cooperation among each server, they must know the caching state and request features of its neighboring servers. But communicating information among servers will lead to additional overheads. As a result, an efficient and cooperative caching strategy should be there. In related papers [8], [11] the central server sends the update information after each request which causes the back-haul traffic to be at a significant level. To overcome this limitation, in our proposed methodology, the cloud server sends out the global parameters to the UAV and the ground base station only once at each time slot. As a result, this reduces the overhead transmission and allows the central server to make updated policies after observing all the requests in that time period. In addition, the research in [33] also presented a cooperative content placement algorithm through caching the popular contents in small base stations (SBSs) that had substantially better channel qualities, thereby leading to lower average transmission delays. They focused on learning the popularity profile through the use of an algorithm that used Bayes-based learning. The learning rate has also increased as they took into consideration the exploitation versus exploration trade-off. Developing a caching technique for each server separately may result in insufficient cache utilization. This occurs when cached data from neighboring servers overlap or when the caching loads of edge servers are unbalanced [34]. Secondly, for effective collaboration management, a dynamic and adaptive caching system is necessary [35].

In summary, researches such as [6], [14], [18], [25], cached contents dynamically using DRL but haven't considered a cooperative caching mechanism. On the other hand, recent research like [8], [11], [33], have taken cooperative caching into consideration but have not taken user mobility into account. Moreover, in [9], [10], they have considered mobility but did not consider cooperative caching as such. Our proposed research caters to every aspect of mobility, cooperative-dynamic-caching, along with the use of multi-agent actor-critic reinforcement learning to overcome all the shortcomings of the above-mentioned papers. Furthermore, we have proposed a 3-tier architecture to retrieve contents on user demand. In our proposed caching mechanism, the UAV layer supports the ground base stations and reduces dependency on the cloud data center to make an efficient caching system. Therefore, along with our proposed architecture, we have developed algorithms that address the problem of where to cache and what to cache. It acts as a combined system that orchestrates to optimize all the constraints to provide a better cache hit ratio.

III. System Model

A. Proposed Architecture

The proposed joint-air-ground architecture for cooperative content caching is illustrated in Fig. 1. It consists of the ground base stations, UAVs, and cloud servers, forming a hierarchy while retrieving context-aware cached data. The proposed architecture is a customized version of the more commonly recognized two-layer network architecture [20], [23], [24]. We also incorporate UAVs based on user mobility which brings significant improvements in communication, computation, and storage capabilities, and the cloud server also acts as the third tier in our proposed architecture. In the ground, we have base stations, which will cache contents dynamically. In the air, the UAVs will reallocate themselves according to the user mobility to support the caching for the mobile users. If the cached file is not found from either GBSs or UAVs, then the file is retrieved from the cloud data center. Thus, our methodology reduces dependency on the cloud and decreases the transmission delay. In addition, we are also proposing a cooperative caching policy using a multi-agent actor-critic network to dynamically cache contents.

B. Problem Formulation

The integration of the air and the ground networks leads to some constraints that must be satisfied to optimize the problem of efficient caching. Throughout the paper, $G = \{g_1, g_2, \dots, g_N\}$, has been used to refer to the number of ground base stations, where g_N represents the capacity of the server n to cache contents. The N in this case corresponds to the total number of servers of the GBSs. On the other hand, $U = \{u_1, u_2, \dots, u_M\}$, has been used to denote the number of unmanned aerial vehicles where u_M represents the capacity of the servers of the UAVs to cache contents. The M in this case corresponds to the total number of UAV servers. The movie data that we are considering are from the MovieLens dataset. So, we have chosen $D = \{d_1, d_2, \dots, d_I\}$, to represent the total movie items and d_I corresponds to the i^{th} movie item's size. Time has been assumed to be taken into slots where each slot represents a certain time period, represented by $T = \{1, 2, \dots, T\}$. In addition to this, each user is being associated with the closest UAV-GBS pair. So, we have defined $r^t_{n,j} = \{r^t_{n,1}, r^t_{n,2}, \dots, r^t_{n,J}\}$ where $r^t_{n,j} = 1$ represents the information that the data item which is requested by the user is being served by the closest GBS at time t. Similarly we have also defined $r^t_{m,j} = \{r^t_{m,1}, r^t_{m,2}, \dots, r^t_{m,J}\}$ where $r^t_{m,j} = 1$ represents the information that the data item which is requested by the user is being served by the closest UAV at time t. Besides this, the caching state of GBS, n, is defined as $c^t_{i,n} = \{c^t_{1,n}, c^t_{2,n}, \dots, c^t_{I,n}\}$ where $c^t_{i,n}$ is the movie item stored in i^{th} position of the n^{th} GBS. Similarly the caching state of UAV, m is defined as $c^t_{i,m} = \{c^t_{1,m}, c^t_{2,m}, \dots, c^t_{I,m}\}$ where $c^t_{i,m}$ is the movie item stored in i^{th} position of the m^{th} UAV at time t. Finally, to record the request frequency of movie item i at time t, $F^t = \{f^t_1, f^t_2, \dots, f^t_I\}$ is introduced.

1) Caching model At GBS: We define $y^{j\to n}_j \epsilon \{0, 1\}$ as a decision variable, which indicates whether or not GBS server, n, should cache the movie request, $r^t_{n,j}$. The decision variable $y^{j\to n}_j$ is given by (1):

$$y^{j\to n}_j = \begin{cases} 1, \text{ if request by user, r}^t_{n,j} \\ \text{ should be cached at GBS, n} \\ 0, \text{ otherwise.} \end{cases} \quad (1)$$

In our proposed caching model, a movie request of size, d^t_j, is cached in GBS server, n, when the summation of d^t_j and the size of the movies cached at that instance is less than the total capacity GS_n, where $n \,\epsilon\, N$. The constraint is given by the (2):

$$\sum_{j=1}^{J} \left(d^t_j y^{j\to n}_j + \sum_{i=1}^{I} c^t_{i,m} \right) \le GS_n \quad (2)$$

At UAV: In the case of UAV we define $y^{j\to m}_j \epsilon \{0, 1\}$ as a decision variable, which indicates whether or not UAV server, m, should cache the movie request, $r^t_{m,j}$. The decision variable $y^{j\to m}_j$ is given by (3):

$$y^{j\to n}_j = \begin{cases} 1, \text{ if request by user, r}^t_{m,j} \\ \text{ should be cached at UAV, m} \\ 0, \text{ otherwise.} \end{cases} \quad (3)$$

In our proposed caching model, a movie request of size, d^t_j, is cached in UAV server, m, when the summation of d^t_j and the size of the movies cached at that instance is less than the total capacity US_m, where $m \,\epsilon\, M$. The constraint is given by the (4):

$$\sum_{j=1}^{J} \left(d^t_j y^{j\to m}_j + \sum_{i=1}^{I} c^t_{i,n} \right) \le US_m \quad (4)$$

2) Mobility Model: To model the mobility of the users, we have implemented the Random Waypoint model [22]. The mobile nodes (MNs), which in our case, can be considered as the users, begin at a starting point where it stays for a certain amount of time. After the pause, the MNs start moving towards the randomly chosen destination points. The velocity of the Mobile Nodes can be selected randomly, but to keep the model on a fixed track, we are keeping the minimum and maximum speed constant. Furthermore, a definite waiting time for each node is selected to look for the next node if the waiting time for the chosen node exceeds the one that is made constant. Alongside these factors, we considered the probability of moving randomly, which is crucial in applying the random-waypoint model effectively.

$$Pr\left(v_x' \mid v_x\right) = \begin{cases} p_\mu, & v_x^{t'} = v_x^t, x \in \mathcal{X} \text{ or } t \in \mathcal{T} \\ p_\xi, & \text{otherwise} \end{cases} \tag{5}$$

where ξ denotes an angle distributed between [0, 2π]. The new location of the user $v_x^{t'}$ depends only on the past location v_x^t. The user may stay in the same position for the next time slot with probability p_μ. Accordingly, user x can randomly move to a different position with probability p_ξ.

The Random Waypoint Model took in the number of nodes, i.e., (users), the simulation area, which is the dimension of "x" multiplied by "y" and the minimum and maximum velocity and the corresponding waiting time needed for the nodes. The above factors were considered, and the movement of the nodes was assigned by simply calculating the product of the direction and the velocity of the node. After the updated node position is calculated, the distance between the new node position and the waypoint is also calculated. If we have arrived at the correct node, then the information for the arrived nodes is being updated. This decision of arriving at the desired node was taken by comparing whether the previously calculated distance is less than or equal to the velocity at which the node was traveling and after checking if the waiting time for the node was less than or equal to 0. If the nodes still surpass the waypoint, then a step back is performed to ensure that the position of the arrived node is now assigned to that of the waypoint. In this way, the Random-Waypoint model is being used to predict the mobility of the users.

However, the constraint of delivering content to mobile users remains as ground base stations at different timestamps cannot always serve the users with the popular content in peak hours. For that reason, we have introduced UAVs to tackle the problem of delivering content to mobile users. On the other hand, the UAVs also has some limitations while serving mobile users as UAVs have a limit to the flight distance that these can cover as well as the height at which these need to operate, which we have considered as follows:

$$\begin{aligned} &\sum_{t=1}^{T-1}\left(u_m^{t+1} - u_m^t\right) \le \varpi^{\max} \quad \forall m \in \boldsymbol{M} \\ &h_{\min} \le h_m \le h_{\max} \quad \forall m \in \boldsymbol{M}, \end{aligned} \tag{6}$$

The maximum and the minimum altitude of the UAV is represented by $h_{\min}, h_{\max}$ respectively. $\varpi^{\max}$ denotes the total flight distance that the UAV can cover in the given time steps, which cannot be exceeded by the UAVs when they change their position from u_m^t to u_m^{t+1} at each time step.

The positioning of the UAV's will be determined by the mobility of users, which will be optimized in each time step to provide the best signal or QoE to users.

3) Communication Model: To clearly explain various modes of communication among users, GBS, UAVs, and the cloud, a channel model has to be defined. The channel model takes into account particular aspects of communication. So, we are proposing a concrete channel model that is described below.The $l_{FS}\left(d_0\right)$ is calculated by applying the following formula that takes into account a number of attributes:

TABLE I
VARIABLES AND ATTRIBUTES

Notation	Description
u_m^t	Position of m^{th} UAV in the t^{th}time index
g_n^t	Position of the n^{th} GBS in the t^{th}time index
v_x^t	The x^{th} user's position in the t^{th}time index
$d_{m,x}^t$	The distance from the m^{th} UAV to the x^{th} user in the t^{th} time index
$d_{n,x}^t$	The distance from the n^{th} GBS to the x^{th} user in the t^{th} time index
XσLoS, XσNLoS	Shadowing random variable
$l_{FS}\left(d_0\right)$	Free space path loss from a reference distance
$\mathrm{PL}_{\mathrm{LoS}_u}$	$\mathrm{l}_u^{\mathrm{LoS}}\left(u_m^t, v_x^t\right)$
$\mathrm{PL}_{\mathrm{NLoS}_u}$	$\mathrm{l}_u^{\mathrm{NLoS}}\left(u_m^t, v_x^t\right)$
$\mathrm{Pr(LoS)}_u$	The probability of the LoS link for UAV
$\mathrm{PL}_{\mathrm{LoS}_g}$	$\mathrm{l}_g^{\mathrm{LoS}}\left(g_n^t, v_x^t\right)$
$\mathrm{PL}_{\mathrm{NLoS}_g}$	$\mathrm{l}_g^{\mathrm{NLoS}}\left(g_n^t, v_x^t\right)$
$\mathrm{Pr(LoS)}_g$	The probability of the LoS link for GBS
d_0	Reference distance
f_c	Carrier frequency
c	Speed of light

$$l_{FS}\left(d_0\right) = 20\log\left(d_0 \cdot f_c \cdot 4\pi/c\right) \tag{7}$$

The time horizon of UAV-GBS is split into T equal time slices, indexed by $t \,\epsilon \{1, 2, \ldots, T\}$ and the UAV-GBS transmission occurs through the mmWave frequency band for the A2G links. For the channel model, the downlink transmission is only considered because the sheer volume of download data content is relatively far greater than the number of service requests that are fetched. The mobility of UAVs plays a role in managing the cross-link channel strength, which is an intriguing component of UAV-enabled communications [36]. As orthogonal transmission schemes like time-division multiple-access (TDMA) have been used to mitigate the cross-link interference [37]. We assume that each UAV will use the entire primary band alone for a certain amount of time based on dynamic Time Division Multiple Access (TDMA) [38]. The bandwidth utilization of the UAVs will be handled by a central coordinator in the cloud. The coordinator will assign the primary band to a UAV for a set length of time, avoiding the possibility of the channel interference caused by other UAV base stations nearby.

The path loss of LoS and NLoS links is formulated according to the log-normal shadowing model by picking specific channel parameters, given by the following set of (8) and (9):

1) For UAV:

$$\begin{aligned} l_u^{\mathrm{LoS}}\left(u_{m,}^t, v_x^t\right) &= l_{FS}\left(d_0\right) + 10\mu_{\mathrm{LoS}}\left(d_{m,x}^t\right) + \chi_{\sigma\,\mathrm{LoS}} \\ l_u^{\mathrm{NLoS}}\left(u_m^t, v_x^t\right) &= l_{FS}\left(d_0\right) + 10\mu_{\mathrm{NLoS}}\left(d_{m,x}^t\right) + \chi_{\sigma\mathrm{NLoS}} \end{aligned} \tag{8}$$

2) For GBS:

$$\begin{aligned} l_{q\,\mathrm{LoS}}^{\mathrm{LoS}}\left(g_n^t, v_x^t\right) &= l_{FS}\left(d_0\right) + 10\mu_{\mathrm{LoS}}\left(d_{n,x}^t\right) + \chi_{\sigma\,\mathrm{LoS}} \\ l_g^{\mathrm{NLoS}}\left(g_n^t, v_x^t\right) &= l_{FS}\left(d_0\right) + 10\mu_{\mathrm{NLoS}}\left(d_{n,x}^t\right) + \chi_{\sigma\mathrm{NLoS}} \end{aligned} \tag{9}$$

where the symbols that have been used in the 8 and 9 is described in Table I above.

The in (7) is used for both the cases of the UAV and the GBS as well, to calculate the free space path loss from a reference distance. From the formula that we previously defined for calculating the path loss of line-of-sight (LoS) and non-line-of-sight (NLoS) links, we again took one parameter each for modelling the links using the shadowing model as stated in Table I.

The elevation angle ϕ_m^t between the UAV and the user plays a crucial factor in determining the probability of the LoS links given by the (10),

$$\phi_m^t = h_m^t / \left(u_m^t - v_x^t\right) \tag{10}$$

where the attributes in (10) represented by ϕ_m^t which is the elevation angle between UAV and user. h_m^t is the height of the m^{th} UAV in the t^{th} time index.

The probability of the LoS link for the UAV is calculated through the following (11):

$$\Pr(\text{LoS})_u = \frac{1}{1 + Ze^{-W(\phi_m^t - Z)}} \tag{11}$$

where the attributes in (11) represented by Z,W which are constants depending on environment. The elevation angle ϕ_n^t between the GBS and the user is a key factor in determining the probability of the LoS links. So, $\phi_n^t = Z$ is considered for this case. The probability of the LoS link for the GBS is calculated through the following (12):

$$\Pr(\text{LoS})_g = \frac{1}{1 + Ze^{-W(\phi_n^t - Z)}} \rightarrow \Pr(\text{LoS})_g = \frac{1}{1+Z} \tag{12}$$

The height and density of the buildings largely affect the probability of LoS connection. For instance, rural areas have a very low density of buildings, i.e. there is less number of buildings in close proximity that results in higher LoS probability whereas an urban area there exists a lower LoS probability. So, the total path loss is expressed as follows:

1) For UAV:

$$PL_{\text{total}_u} = \Pr(\text{LoS}_u) \times PL_{\text{LoS}_u} + \Pr(\text{NLoS}_u) \times PL_{\text{NLoS}_u} \tag{13}$$

2) For GBS:

$$PL_{\text{total}_g} = \Pr(\text{LoS}_g) \times PL_{\text{LoS}_g} + \Pr(\text{NLoS}_g) \times PL_{\text{NLoS}_g} \tag{14}$$

where each component of the (13) and (14) have been defined in Table I. Another significant factor that has to be looked at closely is the signal-to-noise ratio(SNR). Let γ_x denote the SNR of the x^{th} user, which can be expressed as follows:

1) For UAV:

$$\gamma_{x,u} = \frac{P_a \left|h_m\right|^2}{10^{PL_{\text{total}_u}/10} \sigma^2} \tag{15}$$

2) For GBS:

$$\gamma_{x,g} = \frac{P_a \left|h_n\right|^2}{10^{PL_{\text{total}_g}/10} \sigma^2} \tag{16}$$

where the transmit power of the cache node is being denoted by P_a, $|h_m|^2$ is the Rayleigh fading channel gain. This value of SNR has a certain threshold for each GBS, UAV.

4) *Cache Node Selection Model:* In this section we will be defining the cache node selection model i.e maximizing the QoE for users. We define $q_j^{j\rightarrow n} \epsilon \{0,1\}$ as a decision variable, which indicates that the communication link can be formed when the SNR of the user with the GBS, n, is greater than the minimum threshold, η^{GBS}, given in the (17):

$$q_j^{j\rightarrow n} = \begin{cases} 1, & \text{if } \gamma_{\text{i}} > \eta^{\text{GBS}} \\ & \text{communication link can be formed with GBS,} \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Similarly, we define $q_j^{j\rightarrow m} \epsilon \{0,1\}$ as a decision variable, which indicates that the communication link can be formed when the SNR of the user with the UAV, m, is greater than the minimum threshold, η^{UAV}, given by the (18):

$$q_j^{j\rightarrow m} = \begin{cases} 1, & \text{if } \gamma_{\text{i}} > \eta^{\text{UAV}} \\ & \text{communication link can be formed with UAV,} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Finally, for the cache node selection, the communication link has been defined as $a \in \{0,1,2\}$, where $a = 0$ is formed with the nearest GBS, n, when the request can be cached and the communication link can be formed, given that it does not exceed the storage capacity of n. If the request cannot be served by GBS n, then, $a = 1$ is formed with the closest UAV, m, when conditions identical to GBS are met. When communication link cannot be formed with neither GBS nor UAV, the request is retrieved backhaul from the cloud with the communication link, $a = 2$. The cache node selection model is given by (19):

$$\text{a} = \begin{cases} 0, & \text{if } \sum_{j=1}^{J} \left(d_j^t y_j^{j\rightarrow n} q_j^{j\rightarrow n} + \sum_{i=1}^{I} c_{i,n}^t\right) \leq GS_n \\ 1, & \text{if } \sum_{j=1}^{J} \left(d_j^t y_j^{j\rightarrow m} q_j^{j\rightarrow m} + \sum_{i=1}^{I} c_{i,m}^t\right) \leq US_m \\ 2, & \text{otherwise.} \end{cases} \tag{19}$$

5) *Optimization Formulation:* To formulate our objective function we have considered both the cache hit performance as well as the SNR.

Firstly, it is crucial to chose the caching policy that will ensure efficient content retrieval. However, a hierarchy of cache models, including caching in GBSs and UAVs is quite challenging. As a result, for the caching problem, we have proposed to maximize the global cache hit ratio, Ψ_{GCHR}^t of the UAV and GBS to improve the back-haul traffic reduction which can be formulated by the (20):

$$\Psi_{\text{GCHR}}^{\text{t}} = \left(\chi \cdot \ l^{\text{t}} + (1-\chi) \cdot \text{o}^{\text{t}}\right) \tag{20}$$

where

$$l^{\text{t}} = \frac{1}{J} \sum_{j=1}^{J} \frac{\mho_{l,j}^t}{r_{l,j}^t}$$

$$\mho_{l,j}^t = \begin{cases} 1, & \text{if } \text{r}_{l,j}^t \in \text{c}_l^t \\ 0, & \text{otherwise.} \end{cases}$$

and

$$\mathrm{o}^{\mathrm{t}} = \frac{1}{J}\sum_{j=1}^{J}\sum_{\mathrm{o}=1}^{\mathrm{O}}\frac{\mho^{t}_{\mathrm{o,j}}}{r^{t}_{\mathrm{o,j}}}$$

$$\mho^{t}_{o,j} = \begin{cases} 1, & \text{if } \mathrm{r}^{t}_{\mathrm{o,j}} \in \mathrm{c}^{t}_{\mathrm{o}} \\ 0, & \text{otherwise.} \end{cases}$$

Here in (20), l^t represents the hit ratio of the local server and o^t represents the hit ratio of the neighboring server at time t. The weight χ is used to balance the rewards gained by the local and other servers. When χ is 1, it means there is no cooperation with the neighboring server. So, the weights can be tuned to set the cooperation among the servers. Furthermore, $\mho^{t}_{o,j}$ and $\mho^{t}_{l,j}$ are the cache hit decision variable, alongside $r^{t}_{l,j}$ and $r^{t}_{\mathrm{o,j}}$, which represents the request to the l^{th} and o^{th} server at time t.

Secondly, in order to improve the QoE of users, we need to make sure that the maximum SNR is achieved, Ψ^{t}_{SNR} as formulated by the (21):

$$\Psi^{t}_{\text{SNR}} = \frac{\gamma^{t}_{g}}{\eta_g} + \frac{\gamma^{t}_{u}}{\eta_u} - P_{\mathrm{g}} - P_u \tag{21}$$

where:

$$P_g = \begin{cases} \zeta_{snr} * \frac{n_g}{\gamma^{t}_{g}}, & \text{if } \gamma^{t}_{g} < \eta_g \\ 0, & \text{otherwise.} \end{cases}$$

$$P_u = \begin{cases} \zeta_{snr} * \frac{n_u}{\gamma^{t}_{u}}, & \text{if } \gamma^{t}_{u} < \eta_u \\ 0, & \text{otherwise.} \end{cases}$$

In (21), we are rewarding if γ is greater than minimum threshold η and penalizing if otherwise. Here P_g and P_u are the penalties considered and ζ_{snr} acts as the weight which is used to control the penalties.

Based on the discussion above, we have formulated a maximization of global cache hit ratio as an optimization problem as well as making sure the maximum SNR is achieved. The formulation is illustrated below:

$$\begin{aligned} &\max_{a,\mathbf{y}^{j\to n}_{j},\mathbf{y}^{j\to m}_{j}} \frac{1}{T}\sum_{t=1}^{T}\Psi^{t}_{\text{GCHR}} + \Psi^{t}_{\text{SNR}} \\ &\text{s.t. } C1: \sum_{j=1}^{J}\left(d^{t}_{j}y^{j\to n}_{j}q^{j\to n}_{j} + \sum_{i=1}^{I}c^{t}_{i,n}\right) \le GS_n, \\ &\quad C2: \sum_{j=1}^{J}\left(d^{t}_{j}y^{j\to m}_{j}q^{j\to m}_{j} + \sum_{i=1}^{I}c^{t}_{i,m}\right) \le US_m, \\ &\quad C3: \quad c^{t}_{n} \ne c^{t}_{m}, c^{t}_{n} \in \boldsymbol{N}, c^{t}_{m} \in \boldsymbol{M}, \\ &\quad C4: \quad h_{\min} \le h_m \le h_{\max} \quad \forall m \in \boldsymbol{M}, \\ &\quad C5: \quad \sum_{t=1}^{T-1}\left(u^{t+1}_{m} - u^{t}_{m}\right) \le \varpi^{\max} \quad \forall m \in \boldsymbol{M}, \\ &\quad C6: \quad a \in \{0,1,2\} \end{aligned} \tag{22}$$

Here constraint $C1$ and $C2$, ensures that the maximum storage is not exceeded when caching. $C3$ restricts that cache state of UAV-GBS pair are not identical at time t. $C4$ deals with the operating height for the UAVs. The constraint in $C5$ represents the total flight distance that the UAVs can travel. Finally, a, represents the communication links, where for $a = 0$ user request is served by GBSs; $a = 1$ means UAVs serve user request; $a = 2$ means the Cloud serves the user request.

To reach the exact solution where a cache state will output the maximum global cache-hit ratio will be computationally very expensive as the above problem is NP-hard.

In the later sections, to address the challenges of cooperative caching, a cooperative DRL approach is being proposed in which the GBS server would adjust caching decisions with the UAVs based on the actions and states, ensuring a global reward.

IV. Cache Node Selection

A. *Cache Node Selection Using K-Means Clustering*

From the Communication model that we have proposed earlier, we ended up with the channel model and the SNR, which caused our joint air-ground architecture to segregate the users. Since the path loss calculated from the channel model is dominated by the distance between the user and the UAV or GBS. So, the initial position of the UAVs and GBSs are taken into account and can be performed based on the joint air-ground architecture. The K-means algorithm is applied to find the minimum distance from a centroid to each user. The K-means algorithm mainly consists of two parts, namely initialization, and iteration. For initialization, the m cluster centers have to be determined. A position is picked at random to be the cluster center. Also, in the iteration part, each user is assigned to its closest cluster center, i.e., the set of the m^{th} cluster u_m, is defined. After that, the new cluster center is re-computed until the iteration error is less than a threshold that had been previously set. In this way, each particular UAV gets assigned to a cluster and the K-Means Clustering algorithm holds, leading to proper cache node selections.

The User's mobility matrix, M_n, is taken as the input from the Random Waypoint Model that has been generated. In addition to this, the communication links, $a \in \{0, 1, 2\}$ signifies that for $a = 0$ user request is served by GBSs; $a = 1$ means user request is served by UAVs; $a = 2$ shows user request is served by the Cloud. K-means clustering algorithm has been used as the estimator where the total number of clusters is taken to be equal to the number of UAVs that we have, as each UAV is capable of serving one specific cluster of users. As a result, centroid of clusters $[\, u_1, u_2, \ldots, u_M]$ has been assumed to be the UAVs. To serve the mobile users we have incorporated K-means clustering algorithm which segments them into separate clusters. At the start of each time period, based on the mobility of the users, the UAVs are reallocated to their optimal positions in each cluster by means of the clustering algorithm.

For V number of users, first of all, each GBS is checked to see if the signal-to-noise Ratio (SNR) value, γ_x, is greater than the SNR threshold. To calculate the SNR value, the PL_{Total} value is used from the Channel Model equations. Only if these

Algorithm 1: Cache Node Selection Using K-Means Clustering.

```
Input: User's mobility matrix $\mathrm{M_n}$; Number of clusters K;
Output: Communication link a $\in \{0, 1, 2\}$.
1:  while $t \leq T$ do
2:    Use K-means to derive the centroid of clusters
      $[\mathrm{P_1, P_2, \ldots P_K}]$, where $\mathrm{P_i}$ is the position of the $i^{th}$
      UAV-BS at the current time index e.
3:    while $v \leq X$, where $x_v \in X$ do
4:      for all $n$ such that $n \leq G$, where $\mathrm{g_n} \in G$ do
5:        Use the distance between the user and the GBS
          $\|\mathrm{x_v^t} - \mathrm{g_n^t}\|$ to derive the path loss $\mathrm{PL^{GBS}}$ do
6:        Calculate the user SNR, $\gamma_\mathrm{i}$
7:        if $\gamma_\mathrm{i} > \eta^{\mathrm{GBS}}$: then
8:          set $c_i = 0$
9:          a.append(0)
10:         break
11:       end if
12:     end for
13:   end while
14:   if $c_i \neq 0$ then
15:     while $m \leq \mathrm{U}$, where $\mathrm{u_m} \in \mathrm{U}$ do
16:       Use the distance between the user and the
          UAV-GBS $\|\mathrm{x_v^t} - \mathrm{p_m^t}\|$ to derive the path loss
          $\mathrm{PL^{UAV}}$
17:       Calculate the user SNR, $\gamma_\mathrm{i}$
18:       if $\gamma_\mathrm{i} > \eta^{UAV}$: then
19:         set $c_i = 1$
20:         a.append(1)
21:         break
22:       end if
23:     end while
24:   end if
25:   if $\mathrm{c}_i \neq 1$ then
26:     a.append(2)
27:   end if
28: end while
29: return a
```

constraints are satisfied, then 0 value is appended to the connection matrix and, the SNR values are stored for each possible connection, which will be used in Algorithm 2 afterwards. Otherwise, the same set of conditions work for the UAVs. If the criteria are satisfied, then the value of 1 gets appended to a. If the above conditions are not true, then the value of 2 is set to a. In this way, the selection of caching nodes takes place by picking out caching nodes from clusters defined from the usage of the K-means clustering algorithm.

V. Caching Using Multi-Agent Actor-Critic Framework

After selecting where to cache, using the cache node selection algorithm, we need to develop a dynamic caching policy that would decide what to cache in the UAV and GBS server. As a result, we have developed a multi-agent actor-critic algorithm to serve our purpose. For simplicity, we are assuming that a user request will be handled by the closest GBS and UAV, as they would have better SNR rather than a server that is located far away. After receiving a user request, the server selects the caching action and sends the features and actions to the cloud server, which acts as a centralized server. The server then evaluates the action performed by the UAV and GBS, and based on how they performed, sends the updated policies back to them to improve their respective caching actions. The evaluation is done using temporal difference error and is used to update the local server models. Since our goal is to maximize the global cache hit ratio, the content can be served by any one of the servers. As a result, the servers need to cooperate to reach the global goal. For example, if GBS cannot serve a content request, then the request is passed to the UAV server, depending on if the UAV has the requested item.

Using centralized DRL Fig. 2, we will be training our model. The cache buffer in the GBS will determine a local action from the data requests that it received. The local cache actions, buffer states, and requested features from all GBS, and UAV, will be sent to the central cloud server at each time step. The cloud server evaluates the actions sent by the local servers and sends the updated policy depending on the parameters that it has received. The update that the central server sends will consist of only the parameters and cooperation reward. Therefore, this will indicate the local servers as to which caching policy to be used, in the next time step. As a result, the transmission overheads will be insignificant. For instance, when the UAV or GBS receives the user requests, they make the caching decisions based on their current local model. Then they send the caching states, actions, and request features to the central server. The central server evaluates the received states and trains its parameters to evaluate the caching actions in a better way. The Critic network does the evaluation that sends the TD errors to the local servers which are then used to update their action policy.

A. *Markov Decision Process*

We approximate the optimization of global cache hit ratio as a Markov Decision Process (MDP). Accordingly three critical elements have been identified in MDP, which are state space, action and reward function.

1) State Space: It is extremely important to recognize the features that would lead to better cache hit ratio. Conventionally, a binary matrix $C_n^r = \{c_{i,n}^r \mid d_i \in D, g_n \in G\}$ and $C_m^r = \{c_{i,m}^r \mid d_i \in D, u_m \in U\}$ can be pre-defined for GBS and UAV respectively, however it would be inefficient to train using a sparse matrix and will be computationally more expensive. Also it would not be realistic to know about the total data items beforehand in practical scenarios as stated in [11].

As a solution, we are proposing a fixed length caching state of size N for GBS and M for UAV which can be described as $C_n = \{c_1, c_2, \ldots, c_N\}$ and $C_m = \{c_1, c_2, \ldots, c_M\}$ respectively. Here $c_n = \{d_n, \Omega_n\}$, where d_n represents the item in n^{th} location and Ω represents the feature.

For our work, we have considered 6 features $\{\alpha, \Omega, \Omega_s, \Omega_m, \Omega_l, \mu\}$ as described in Table II. When the

TABLE II
FEATURE SPACE SELECTION

Notation	Description
C_n^r	The cache state of GBS$_n$ at r
C_m^r	The cache state of UAV$_m$ at r
c_n^r	The nth data item in GBS at r
c_m^r	The mth data item in UAV at r
d^r	The data item in Cache at request r
Ω^r	The feature of a data item in Cache at r
α^r	The request arrival time of data item at r
ω^r	The total request frequency of data item at r
ω_s^t	Short-term request frequency of data item at t
ω_m^t	Medium-term request frequency of data item at t
ω_l^t	Long-term request frequency of data item at t
μ	Average user rating of item

buffer is full, these will be used to determine the content that is to be replaced when a new request comes in. The first two features are used for conventional caching techniques such as LRU and LFU where α is the arrival time of each request so that when a new request arrives, the buffer can replace the least recent one with the new one.The second one Ω^r can be defined as $\Omega = \sum_r f_{x,i}^r$ where $f_{x,i}^r$ represents the frequency of data item i at r^{th} request by user x. Next we have also considered the short ω_s^t, medium ω_m^t and long term ω_l^t request frequency of items as introduced by [8]. They are calculated as follows:

$$\Omega_s^t = \sum_{t-\tau_s}^{t} \sum_r f_{x,i}^r \tag{23}$$

$$\Omega_m^t = \sum_{t-\tau_m}^{t} \sum_r f_{x,i}^r \tag{24}$$

$$\Omega_l^t = \sum_{t-\tau_l}^{t} \sum_r f_{x,i}^r \tag{25}$$

where τ_s, τ_m, τ_l are considered as 5, 10 and 50-time steps. Finally, since the dataset we used had user ratings for each movie, we also considered using average rating μ as a feature since the rating is treated as a popularity metric in the deep learning paradigm. So we will be replacing the least rated movie in the buffer when a new request arrives.

2) Action Space: In our proposed framework, the output of the Actor-network will determine the caching action, and we will also need to define them like we selected our features. Instead of bothering about the replacement behavior of the cache buffer, the action will simply output a caching policy which will be used by the servers to update their state. As a result, our proposed algorithm will output the caching policy, which will be followed in the next time step by the server to cache and replace items. With this approach, our action space will be greatly reduced and will be discrete. The policy vector can be denoted by $A_n = \{a_1, a_2, \ldots, a_n\}$ where a_n represents the policy which will be used to make caching decisions. For example, a_1 represents replacing cached items that are least recently requested. So, when the actor-network signals a_1 for the local server, the policy will cache a new item by replacing the least requested item throughout that time step. For our work, we have determined six action policies as defined in Table III, which are; replacing items that were requested least recently, which had the least request frequency, least request frequency in the last 5, 10, and 50-time steps, and the least data ratings.

TABLE III
ACTION SPACE SELECTION

Notation	Description
a_1	Cache with least recently requested item
a_2	Cache with least frequently requested item
a_3	Cache with least frequently requested item in last τ_s steps
a_4	Cache with least frequently requested item in last τ_m steps
a_5	Cache with least frequently requested item in last τ_l steps
a_6	Cache with least rated item

3) Reward Function: We have considered the objective function as our reward function as defined in (22).

$$\Psi^{\mathrm{t}} = \Psi_{\mathrm{GCHR}}^t + \Psi_{\mathrm{SNR}}^t \tag{26}$$

B. Caching Methodology

Finally, this is our proposed algorithm, which consists of the critic and actor networks implemented in the central servers and local buffers accordingly. Now, the workflow of the algorithm that we are proposing has been explained below: At a given time step, based on the user position and user requests, the SNR is calculated using Algorithm 1. As a result, if a connection is possible to a server, it will be known during that period. Then each user's request is handled by the nearest UAV-GBS pair. If the item is not available in either of the servers, then the data is fetched from the cloud server to serve the user, and the item is cached in either UAV or GBS, which has a better SNR value. The server will then process the content request and make a caching decision based on the current caching action policy as updated at the beginning of the time step. Then, the states, actions, and new states are sent to the central server at the end. In the central server, the critic network will evaluate the rewards based on the features and actions as well and calculate the TD error. As a result, this will allow the Critic network to update its parameters by minimizing the loss function. Finally, the TD error will be sent to the UAV-GBS pair to update their caching policies. The process will repeat in the next time step.

1) Actor Network: The actor-networks are positioned in the local servers, i.e., in the UAV and GBS, which takes caching state and request features as input. The network is denoted by Θ^π, which seeks the optimal caching policy by mapping the state C to action a. In each time step, servers choose their actions based on the current parameters of the network.

$$a^t = \pi\left(C^t \mid \Theta^\pi\right) \tag{27}$$

2) Critic Network: The local server estimates the values of the selected actions, and the critic network is used in this case. As

Algorithm 2: : GBS-UAV Actor-Critic Algorithm for Cooperative Caching.

```
In GBS and UAV:
Initialize: Actor network $\pi_n(C_n \mid \Theta_n^\pi)$ and $\pi_m(C_m \mid \Theta_m^\pi)$
  with weights $\Theta_n^\pi$ and $\Theta_m^\pi$; Target Actor network $\pi'_n$ and
  $\pi'_m$ with weights $\Theta'_n \leftarrow \Theta_n^\pi$ and $\Theta'_m \leftarrow \Theta_m^\pi$; Critic
  network $Q_{n,m}(C1, C2, \ldots\ldots, C_{n,m}, a_{n,m} \mid \Theta_{n,m}^Q)$ with
  weights $\Theta_{n,m}^Q$; Target Critic network $\Theta'_{n,m} \leftarrow \Theta_{n,m}$;
  Cache Buffer state $C_n$ and $C_m$; Replay Buffer $B_{n,m}$
1: while $t \le T$ do
2:   Select action, $a_n^t = \pi_n(C_n \mid \Theta_n^\pi)$ and
     $a_m^t = \pi_m(C_m \mid \Theta_m^\pi)$ with current policy
3:   while $r \le R$ do
4:     if $C_n^r$ and $C_m^r$ not full then
5:       if $r_i^t$ in $C_n^r$ or $C_m^r$ then
6:         Update cache and hit ratio
7:       else
8:         Update in $C^r$ which has better SNR, $\gamma$
9:         Update cache hit ratio
10:      end if
11:    else
12:      Keep track of caching state $C_n^r$ and $C_m^r$
13:      if $r_i^t$ in $C_n$ or $C_m$ then
14:        Perform $a_n^t$ or $a_m^t$, then send the state $C_n^{r-1}$ or
           $C_m^{r-1}$, action, new state $C_n^r$ or $C_m^r$ to the cloud
           server and update the hit ratio.
15:      else
16:        Perform $a^t$ in C which has better SNR, $\gamma$ then
           send the state $C^{r-1}$, action, new state $C^r$ to the
           cloud server and update the hit ratio.
17:      end if
18:    end if
19:  end while
20:    The cloud server calculates reward $\psi_n^t$ and $\psi_m^t$ for G
       and U and store $(C_n^t, a_n^t, \psi_n^t, C_n^{t+1})$ and
       $(C_m^t, a_m^t, \psi_m^t C_m^{t+1})$ for all $r$ it received during $t$ in
       $B_{n,m}$
21:    Sample a random mini batch of S transitions
       $(C_{n,m}^s, a_{n,m}^s, \psi_{n,m}^s, C_{n,m}^{s+1})$ from $B_{n,m}$
22:    $y_{n,m}^s = \psi_{n,m}^s + \gamma Q_{n,m}$
       $\left(C_1^{s+1} \ldots C_{N,M}^{s+1}, \pi_{n,m}(C_{n,m}^{i+1}) \mid \Theta'_{n,m}\right)$
23:    Calculate TD based on present parameters: $\Phi_{n,m} =$
       $\frac{1}{S}\sum_s \left(y_{n,m}^s - Q_{n,m}\left(C_1^s \ldots C_{N,M}^s, a_{n,m}^s \mid \Theta_{n,m}^Q\right)\right)$
24:    Update the critic network $Q_{n,m}$ by minimizing the
       Huber loss: $z(\Phi_{n,m})$
25:    Update Critic target network:
       $\Theta'^Q_{n,m} \leftarrow \tau\Theta_{n,m}^Q + (1-\tau)\Theta'^Q_{n,m}$
26:    Send $\Phi_{n,m}$ to GBS and UAV
27:    For GBS: $\nabla_{\theta_n^\pi} J = \nabla_{\theta_n^\pi} \log \pi_n(C_n, a_n)\Phi_{n,m}$
28:    For UAV: $\nabla_{\theta_m^\pi} J = \nabla_{\theta_m^\pi} \log \pi_m(C_m, a_m)\Phi_{n,m}$
29:    Update GBS Actor target network:
       $\Theta'^\pi_n \leftarrow \tau\Theta_n^\pi + (1-\tau)\Theta'^\pi_n$
30:    Update UAV Actor target network:
       $\Theta'^\pi_m \leftarrow \tau\Theta_m^\pi + (1-\tau)\Theta'^\pi_m$
31:    $C_n^{r-1} \leftarrow C_n^r$
32:    $C_m^{r-1} \leftarrow C_m^r$
33: end while
```

its input, it takes all the caching states and actions performed by the GBS and UAV at the time, t. The action policy of a particular server is determined by the global states and actions performed by the UAV and GBS at time t. Then, the job of the critic is to calculate the TD-error,

$$\Phi_{n,m} = \frac{1}{S}\sum_s \left(y_{n,m}^s - \gamma Q_{n,m}\left(C_1^s, \ldots, C_{N,M}^s, a_{n,m}^s \mid \Theta_{n,m}^Q\right)\right)$$

$$y_{n,m}^s = \psi_{n,m}^s + Q_{n,m}\left(C_1^{s+1}, \ldots, C_{N,M}^{s+1}, \pi_{n,m}\left(C_{n,m}^{s+1}\right) \mid \Theta_{n,m}^Q\right) \tag{28}$$

where γ is the discount factor that enables a balance between recent and future accumulated reward, S is the size of the mini-batch that will be sampled, and $\Theta_{n,m}^Q$ is the network parameters.

After calculating the TD error the critic network is updated by the loss function in 29:

$$\text{loss, } z(\Phi_{n,m}) = \begin{cases} 0.5\left(\Phi_{n,m}\right)^2/\text{beta}, & \text{if } |\Phi_{n,m}| < \text{beta} \\ |\Phi_{n,m}| - 0.5 * \text{beta}, & \text{otherwise} \end{cases} \tag{29}$$

And for the GBS and UAV, the actor network is updated using the policy gradient:

$$\begin{aligned} \nabla_{\theta_n^\pi} J &= \nabla_{\theta_n^\pi} \log \pi_n(C_n, a_n)\Phi_{n,m} \\ \nabla_{\theta_m^\pi} J &= \nabla_{\theta_m^\pi} \log \pi_m(C_m, a_m)\Phi_{n,m} \end{aligned} \tag{30}$$

3) *Computational Complexity:*

$$\begin{aligned} &\mathcal{O}\left(T \cdot R^t \cdot M \cdot \left(\sum_{i=0}^{L_a} d_a^{(i)} \cdot d_a^{(i+1)}\right) \cdot \right. \\ &\left. N \cdot \left(\sum_{i=0}^{L_a} d_a^{(i)} \cdot d_a^{(i+1)}\right) \cdot \sum_{j=0}^{L_c} d_c^{(j)} \cdot d_c^{(j+1)}\right) \\ &= \mathcal{O}\left(T \cdot R^t \cdot (M+N) \cdot \left(\sum_{i=0}^{L_a} d_a^{(i)} \cdot d_a^{(i+1)}\right)\right. \\ &\left.\left. \cdot \sum_{j=0}^{L_c} d_c^{(j)} \cdot d_c^{(j+1)}\right)\right) \end{aligned} \tag{31}$$

where T is the number of time steps, R^t is the number of content requests at time t, M and N are the number of UAV and GBS respectively, $d_a^{(i)}$ and $d_a^{(i+1)}$ represents the number of nodes at the i^{th} and $(i+1)^{th}$ layer of the actor's neural network, alongside, $d_c^{(j)}$ and $d_c^{(j+1)}$ representing the number of nodes at the j^{th} and $(j+1)^{th}$ layer of the critic's neural network.

Since the actor-networks are implemented at the GBS and UAV, there are (M+N) actor networks. Furthermore, there is only one critic network located at the central server. Finally, the asymptotic analysis shows that the time complexity is linear with T, the number of requests R^t, the number of layers in the actor networks (M+N), and the critic network. We also see similar linear time complexity of A3C in online-resource scheduling algorithms [39].

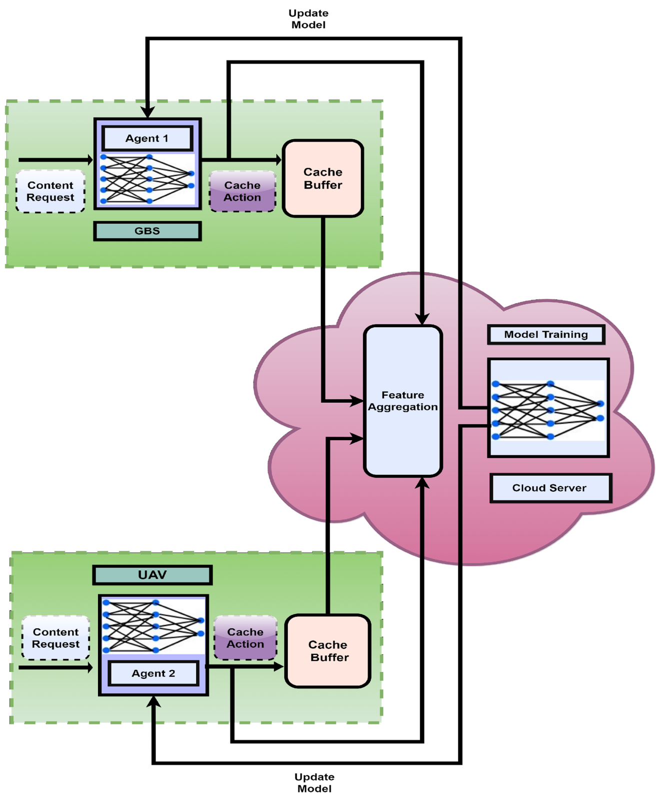

Fig. 2. Cooperative Caching between UAV and GBS.

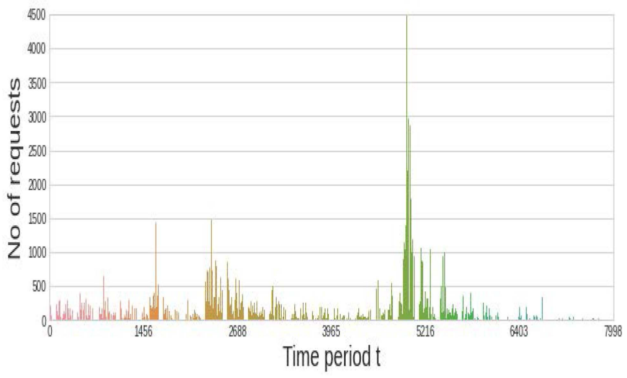

Fig. 3. Number of Movie requests in each time period, t.

VI. Simulation Results

The MovieLens dataset [40], [41] is used to implement the content caching policy. This dataset is mainly composed of rating data for certain movies by different users. The MovieLens 1 M dataset records 1 million ratings of users along with User ID, Movie ID, and timestamp. The movie rating process is analogous to the content request process. We have assumed that when an audience is rating a movie at a particular time, it means the user is also requesting the movie file at that specific time. For our work, we have considered the ratings given by the users as the requests made, which is a valid assumption also done in previous works [31], [42]. As a result, the rating of movies given in an hour is considered as the number of requests by each user in each time, t and shown in the Fig. 3.

TABLE IV
Environmental Parameters

Descriptions and Notations	Value
UAV-BS height,H	40m
Users,x	6040
GBS, N	3
UAV, M	3
GBS link carrier frequency, fc, GBS	2.1GHZ
UAV-BS transmit power, P_{UAV}	20dBm
GBS transmit power, P_{GBS}	40dBm
UAV link carrier frequency, fc, UAV	38GHZ
Free-space reference distance, d_o	5
Shadowing random variables, XσLoS, XσNLoS	8.3, 8.27
UAV, SNR	-93
GBS, SNR	-97
Environment dependent constant, Z,W	11.9, 0.13
Path loss Exponent, α	2

For the simulations, we used 8-core CPUs and 12-VRAM GPU. For training our RL models, we divided the 24,000 hours timesteps from the movie lens dataset into 16,800 hours training set and 7.2 hours test set. We trained our MAAC model for 30 hours and 650 episodes where it took the model, 27 hours, and 575 episodes to converge. During training, for our proposed model at first, we used Mean Squared Error (MSE) as our loss function, which resulted in a lengthy and unstable convergence. Afterward, we opted for Huber loss as our loss function which resulted in a gradual and smoother convergence.

A. Cache Node Selection Results

In Table IV, we have several parameters which we are mainly using for cache node selection. The first one is the height of the UAV-BS from the user, which is 40 m. We have considered the SNR threshold of UAV and GBS to be -93 dBm and -97 dBm, which is suggested as a good connection as in [43]. Then we have GBS link carrier frequency which is the frequency of the ground base station that operates at 2.1 GHz. We also have UAV-BS transmit power which gives the power of UAV transmission as 20 dBm and Ground base station transmitting power as 40 dBm. The UAV-BS links are operated in mmWave band with the carrier frequency of fc, UAV $=$ 38GHZ. Finally, we have shadow-random variables, environmental constants, and path loss exponents that are used to calculate the total path loss for Los and NLoS links.

In Fig. 4, we have a visualization that shows the position of the UAV in n^{th} time-step. At each period, based on the movement of users, the UAVs reallocate themselves to an optimal position. In Fig. 5, the UAVs rearrange their position to adjust to the change in user distribution. In the $(n+1)^{th}$ time-step, the users were more towards the center, as represented by the denser circles. As a result, the UAVs were also closer to the center. Later, due to the shift in users' position, the UAVs adjusted accordingly.

Afterward, we ran the simulation and found that most users got connected to the GBS, and those who didn't get connected to the GBSs got connected to the UAVs. If the SNR value is too

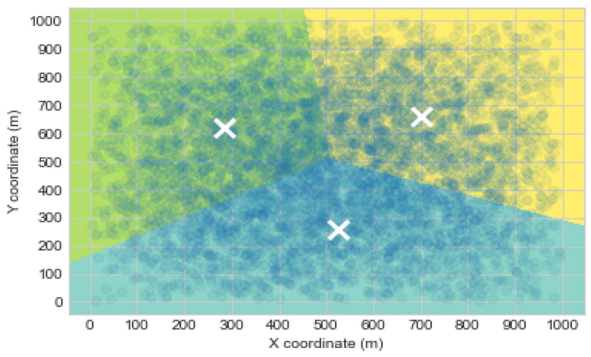

Fig. 4. User mobility and UAV position at n^{th} time-step.

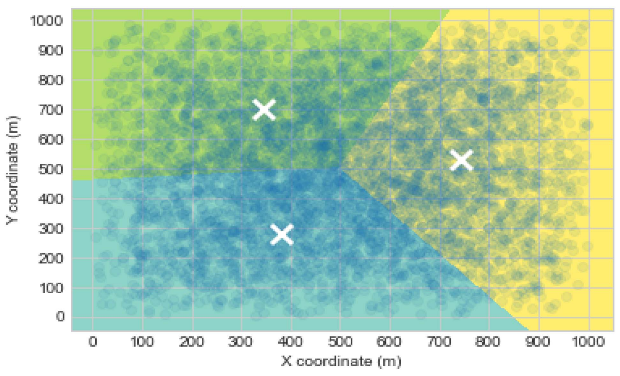

Fig. 5. User mobility and UAV position at $(n+1)^{th}$ time-step.

weak for both GBSs and UAVs, then the user has to retrieve its data from the cloud server.

B. Cache Hit Comparison

For our research, we compared our proposed algorithm to two other conventional caching techniques, as well as one of the state-of-the-art RL-based model used in edge caching.

- Least Recently Used (LRU): The method caches item based on the time they arrived, i.e., if the cache buffer is full and a new item is to be cached, then the algorithm replaces the item that arrived least recently.
- Least Frequently Used (LFU): In this method, the request frequency of an item has been tracked so that when a new item is requested, the algorithm replaces the least frequently requested item from the full cache server.
- Double Deep Q-Learning (DDQN): The RL-based model [44] is a more improved version of the commonly known Deep Q-Learning. The proposed architecture of this algorithm contains two identical neural networks where the main neural network has been used to find the max argument of the Q value, which is then further used to extract the action from the second neural network. The error of the model has been formulated as:

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a}{\operatorname{argmax}} Q\left(S_{t+1}, a; \boldsymbol{\theta}_t\right); \boldsymbol{\theta}'_t\right) \quad (32)$$

- Genetic Algorithm (GA): Inspired by biological evolution, GA is a powerful technique to tackle complicated multi-variable non-linear problems to obtain the globally optimal solution asymptotically [45]. The optimization problem that we have proposed can also be approximately solved using GA. The steps of the algorithm are as follows:
    1) Initialize: The input population, i.e the cache states C, terminating condition, probability of mutation Pr_{mt}, probability of crossover Pr_{cr}. The chromosomes are encoded as the movie items, D_i. For each chromosome, we define the fitness function as defined in (20).
    2) Evaluate fitness for each chromosome.
    3) Select chromosomes with higher fitness.
    4) Perform single-point crossover with Pr_{cr}.
    5) Perform bit mutation operation with Pr_{mt}.
    6) Generate a new population and continue the process till the termination condition is met i.e. achieving satisfactory convergence.

Both the conventional algorithm and GA caches items that are relatively popular. When the popularity of the requested item is uniform or when the distribution of the content popularity changes drastically such algorithms cannot adapt. Both DDQN and our proposed algorithm perform better here because their dynamic approach does not depend only on the popularity of content. As a result, it can adjust quickly when the popularity of content keeps changing or does not follow a pattern. However, due to the collaborative approach of our proposed model, the same item is not cached at both places. It is not the case for other methodologies. It also does a better job in caching the contents appropriately to achieve a higher global cache-hit ratio.

For our proposed model, we have considered the actor-network with two fully connected hidden layers of shapes 256 and 128, with a learning rate of 0.001. In the critic network, we have two fully connected hidden layers of shape 128 and 64 units, with a learning rate of 0.004. We have used ADAM as the optimizer for both networks. And finally, we initialized the replay buffer, $B_{n,m}$, with size 10000, and set the mini-batch, S sample size to 28.

1) Caching in Peak Time: To demonstrate the improved performance of our proposed algorithm when there is a surge in user requests, we chose period 4800–6000 from the Movie Lens data-set, discarding a period where the total number of user requests was less than 15. The average number of requests in a given hour during this period was 314.

For the peak hours in Fig. 6, the LRU struggled to provide a good hit ratio. There is an increase initially, but the hit rate gradually falls after a few time steps and remains constant afterwards. The reason for this is the randomness of LRU. Out of all the algorithms, LRU is most prone to caching newer items. As a result, it couldn't capture the complex context of the user requests.

In Fig. 7, there is a rapid increase in the cache-hit ratio, which then stabilizes for the rest of the simulation. It is because the number of requests per hour increased. There is a greater chance for users to request the items more frequently, and since LFU caches based on the frequency of items, it provided a better hit ratio than LRU.

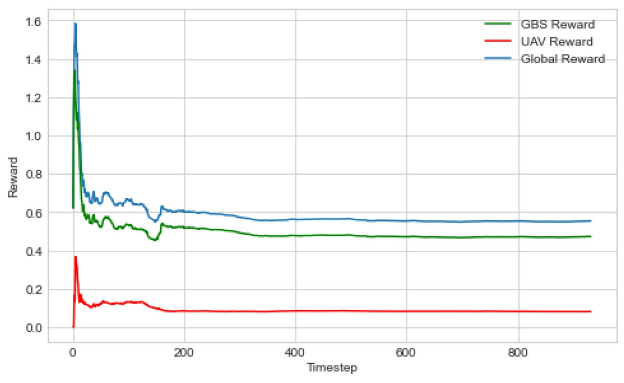

Fig. 6. Comparing Cache hit ratio using LRU during peak hours.

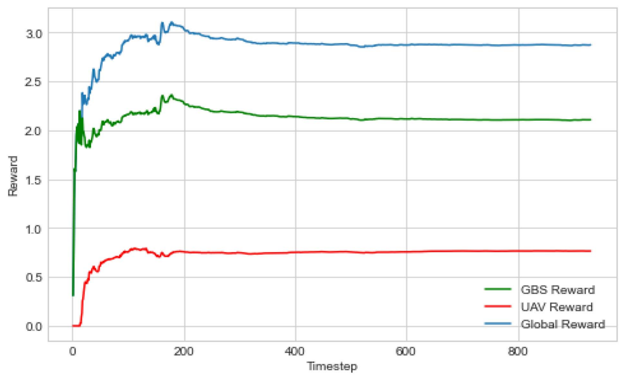

Fig. 7. Comparing Cache hit ratio using LFU during peak hours.

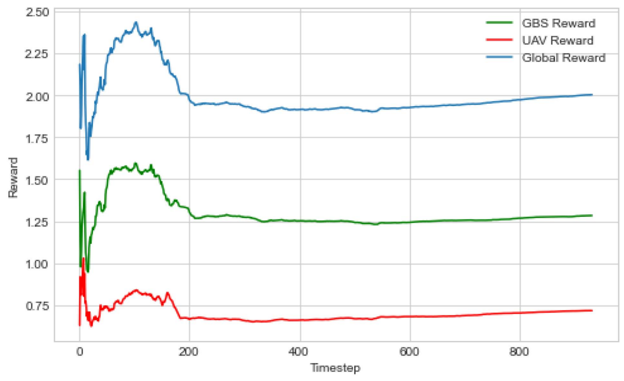

Fig. 8. Comparing Cache hit ratio using GA during peak hours.

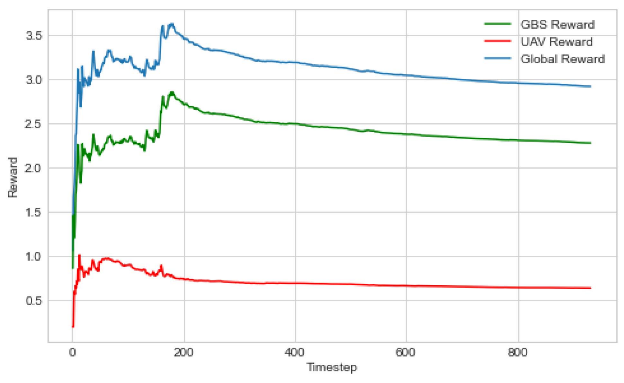

Fig. 9. Comparing Cache hit ratio using DDQN during peak hours.

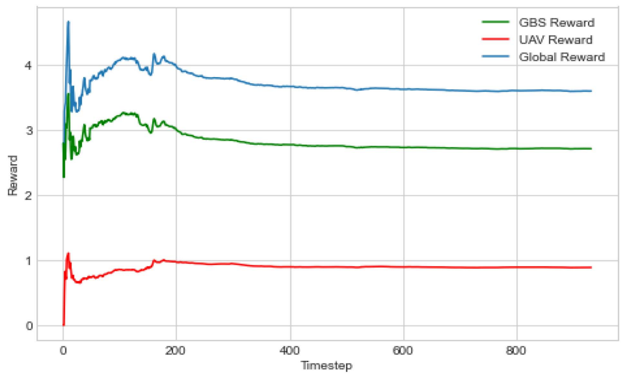

Fig. 10. Comparing Cache hit ratio using Proposed algorithm during peak hours.

There is a sharp dip in the cache hit ratio initially as seen in Fig. 8. This is because the cached items were initialized with movies that were not popular during the peak hour, due to the change in distribution GA failed to cache during the first few time steps. With each generation, fitter chromosomes were identified and as result the cache hit ratio boosted accordingly.

In Fig. 9, there is a slight improvement in the global cache hit ratio compared to LFU. The simulation was done after the DDQN network was converged. The model was able to perform well for the first few hundred time-steps. However, there is a slight fall in the global cache-hit performance afterwards. It was mainly due to the performance decline of UAVs, which gave poor results compared to LFU. DDQN was good at optimizing the local servers individually. Therefore, it achieved a better GBS reward, but due to a lack of cooperation between GBS and UAV, there were cached contents which were duplicated. When a user requested content, it was more likely to retrieve it from the GBS, which cached it. However, the same content was also cached in the UAV, which was inefficient as the UAV could have stored different content. So, next time if the content was missing in the GBS, the user could have retrieved it from UAVs which would have ensured a better global cache-hit ratio.

In Fig. 10, the proposed algorithm did not have a stable cache hit ratio in the first few hours, but after the model had adjusted its parameters, the hit ratio rose gradually and then remained constant afterward. Here, the algorithm was able to handle duplicity, and as a result, the UAV performance didn't decline afterward as it did in DDQN.

Finally, in Fig. 11, we can observe that our proposed model achieved the highest global cache-hit ratio throughout the simulation. Hence, it has outperformed other models as it was able to make a better and more efficient caching action due to the combination of multiple features that the MAAC was able to utilize. It has also cooperated to ensure an efficient content placement which resulted in a greater global cache hit ratio.

2) For Different Capacities: In the figures below, we have compared the cache hit ratios for different storage capacities. Using the peak hours, the simulations are illustrated in Figs. 12–14.

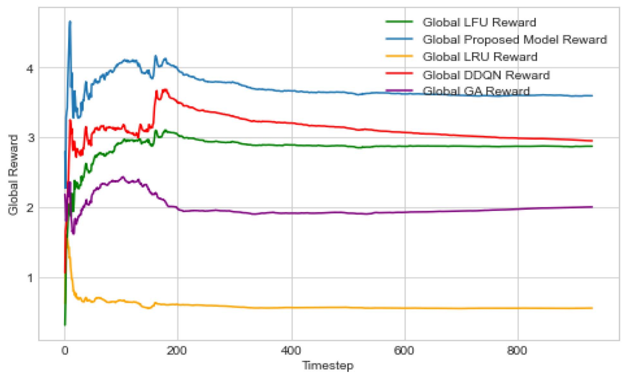

Fig. 11. Comparing global cache hit ratio for all algorithms in peak hours.

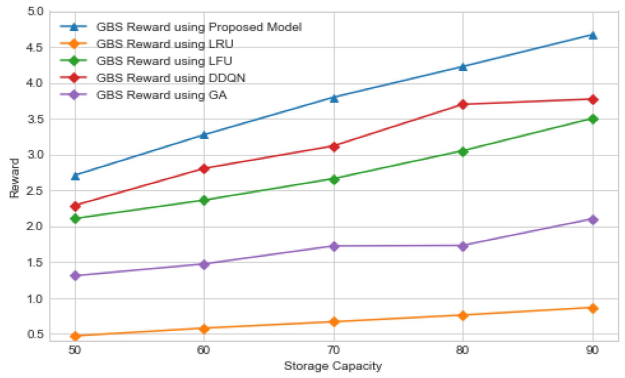

Fig. 12. Comparing Cache hit ratio for different GBS Storage Capacity.

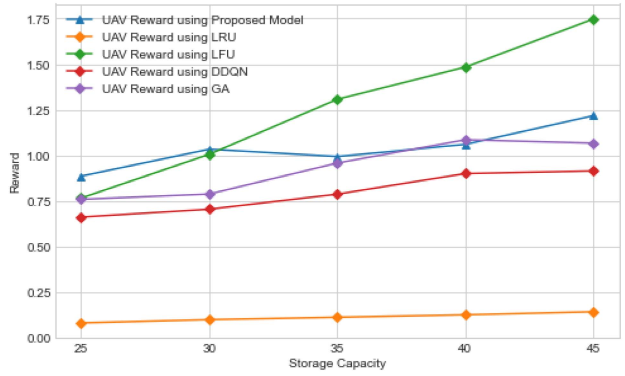

Fig. 13. Comparing Cache hit ratio for different UAV Storage Capacity.

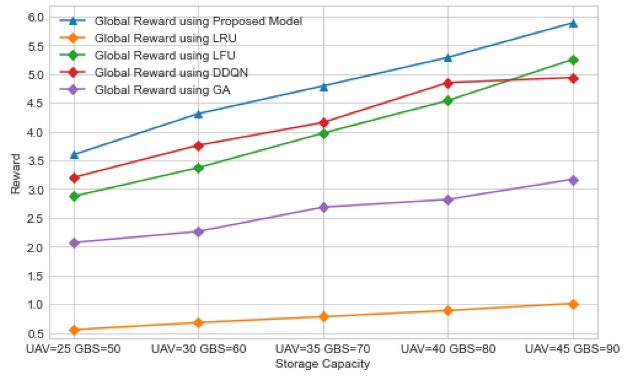

Fig. 14. Comparing Global cache hit ratio for different GBS and UAV Storage Capacity.

In Fig. 12, there is a linear rise in all the algorithms' cache hit ratios as we increase the GBS's storage capacity. Our proposed algorithm achieved the best hit ratio in all cases. However, the LRU struggled to get a high GBS reward even when the storage capacity increased. Meanwhile, there is a consistent gap between all the algorithms.

In Fig. 13, there was not enough increase in the hit ratio for LRU. However, the the proposed algorithm isn't the best performing when it came to UAV. This is because we first checked for a connection in GBS and cached our item, which had a better SNR value. Since GBSs are also closer to most requests, the most requested items are cached in GBSs and not in UAVs. As a result, to achieve a higher global cache-hit ratio, the proposed algorithm is sacrificing its performance for the case of UAV.

Finally, in Fig. 14, we have shown the change in global rewards with multiple UAV and GBS storages. LRU performed the worst among all the algorithms. Our proposed model achieved the highest reward in all sizes. However, there is a smaller gap with LFU, initially, because of the drop in UAV's reward to adjust cooperation. It seems DDQN couldn't scale as it's performance didn't keep on increasing as we increased storage. To sum up, all the algorithms showed linear trend as we increased the storage capacity

VII. Conclusion

In this research, we have used the computation, caching, communication, and control of mobility to cache contents intelligently at edge networks. It is thus noteworthy to mention that a clustering approach for dynamic allocation of UAVs based on user mobility is considered where the users' positions have been mapped using the random-waypoint model. Finally, a cooperative Multi-Agent Actor-Critic based reinforcement learning algorithm has been trained and compared where the servers are intelligent enough to make caching decisions cooperatively and outperform conventional algorithms as well as the state-of-the-art caching methodology in various scenarios. To conclude, this research can be further explored to investigate the communication and cooperation between multi GBS-UAV architecture where the goal would be to reach global cache-hit maximization.

References

[1] "Cisco's global cloud index study: Acceleration of the multicloud era." Feb. 2018, Accessed: May 02, 2021. [Online]. Available: https://blogs.cisco.com/news/acceleration-of-multicloud-era
[2] "Cisco annual internet report - cisco." Mar. 2020, Accessed: Jan. 08, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html
[3] S. Kazmi et al., "Infotainment enabled smart cars: A joint communication, caching, and computation approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8408–8420, Sep. 2019.
[4] A. Ndikumana et al., "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.
[5] A. Ndikumana and C. S. Hong, "Self-driving car meets multi-access edge computing for deep learning-based caching," in *Proc. IEEE Int. Conf. Inf. Netw.*, 2019, pp. 49–54.

[6] K. Thar, N. H. Tran, T. Z. Oo, and C. S. Hong, "DeepMEC: Mobile edge caching using deep learning," *IEEE Access*, vol. 6, pp. 78260–78275, 2018.

[7] T. Taleb, K. Samdanis, B. E. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge architecture and orchestration," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1657–1681, Thirdquarter 2017.

[8] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. IEEE 52nd Annu. Conf. Inf. Sci. Syst.*, 2018, pp. 1–6.

[9] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[10] Y. Chen, K. Liao, M. Ku, and F. P. Tso, "Mobility-aware probabilistic caching in UAV-assisted wireless D2D networks," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[11] Y. Zhang et al., "Cooperative edge caching: A multi-agent deep learning based approach," *IEEE Access*, vol. 08, pp. 133212–133224, 2020.

[12] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. 6th Int. Conf. Adv. Future Internet*, 2014, pp. 48–55.

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[14] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[15] M. Kohler and R. Fies, "ProActive caching- a framework for performance optimized access control evaluations," *Proc. IEEE Int. Symp. Policies Distrib. Syst. Netw.*, 2009, pp. 92–94.

[16] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5341–5351, Aug. 2021.

[17] Z. Piao, M. Peng, Y. Liu, and M. Daneshmand, "Recent advances of edge cache in radio access networks for Internet of Things: Techniques, performances, and challenges," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1010–1028, Feb. 2019.

[18] X. Jin, J. Zhang, X. Sun, P. Zhang, and S. Cai, "Computation offloading and resource allocation for MEC in C-RAN: A deep reinforcement learning approach," in *Proc. IEEE 19th Int. Conf. Commun. Technol.*, 2019, pp. 902–907.

[19] R. Amer, W. Saad, H. ElSawy, M. M. Butt, and N. Marchetti, "Caching to the sky: Performance analysis of cache-assisted comp for cellular-connected UAVs," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2019, pp. 1–6.

[20] F. Cheng, G. Gui, N. Zhao, Y. Chen, J. Tang, and H. Sari, "UAV-relaying-assisted secure transmission with caching," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3140–3153, May 2019.

[21] M. M. Azari, F. Rosas, A. Chiumento, and S. Pollin, "Coexistence of terrestrial and aerial users in cellular networks," in *Proc. IEEE Globecom Workshops*, 2017, pp. 1–6. .

[22] E. Hyytiä, H. Koskinen, P. Lassila, A. Penttinen, J. Roszik, and J. Virtamo, "Random waypoint model in wireless networks," *Netw. algorithms: Complexity Phys. Comput. Sci.*, pp. 16–19, 2005.

[23] N. Cheng et al., "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[24] Y. Zhou, N. Cheng, N. Lu, and X. S. Shen, "Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture," *IEEE Veh. Technol. Mag.*, vol. 10, no. 4, pp. 36–44, Dec. 2015.

[25] C. Zhong, M. C. Gursoy, and S. Velipasalar, "Deep reinforcement learning-based edge caching in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 1, pp. 48–61, Mar. 2020.

[26] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile D2D networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1610–1622, Mar. 2019.

[27] N.B. Melazzi, G. Bianchi, A. Caponi, and A. Detti, "A general, tractable and accurate model for a cascade of LRU caches," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 877–880, May 2014.

[28] C.-T. Chan, S.-C. Hu, P.-C. Wang, and Y.-C. Chen, "A FIFO-based buffer management approach for the ATM GFR services," *IEEE Commun. Lett.*, vol. 4, no. 6, pp. 205–207, Jun. 2000.

[29] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov./Dec. 2018.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[31] W. Jiang, G. Feng, S. Qin, and Y. Liu, "Multi-agent reinforcement learning based cooperative content caching for mobile edge networks," *IEEE Access*, vol. 7, pp. 61856–61867, 2019.

[32] X. Xu, M. Tao, and C. Shen, "Collaborative multi-agent multi-armed bandit learning for small-cell caching," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2570–2585, Apr. 2020.

[33] T. Nie, J. Luo, L. Gao, F.-C. Zheng, and L. Yu, "Cooperative edge caching in small cell networks with heterogeneous channel qualities," in *Proc. IEEE 91st Veh. Technol. Conf.*, 2020, pp. 1–6.

[34] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[35] L. Ramaswamy, L. Liu, and A. Iyengar, "Cache clouds: Cooperative caching of dynamic documents in edge networks," in *Proc. 25th IEEE Int. Conf. Distrib. Comput. Syst.*, 2005, pp. 229–238.

[36] C. Shen, T.-H. Chang, J. Gong, Y. Zeng, and R. Zhang, "Multi-UAV interference coordination via joint trajectory and power control," *IEEE Trans. Signal Process.*, vol. 68, pp. 843–858, 2020.

[37] S. Hayashi and Z.-Q. Luo, "Spectrum management for interference-limited multiuser communication systems," *IEEE Trans. Inf. Theory*, vol. 55, no. 3, pp. 1153–1175, Mar. 2009.

[38] H. Touati, A. Chriki, H. Snoussi, and F. Kamoun, "Cognitive radio and dynamic TDMA for efficient UAVs swarm communications," *Comput. Netw.*, vol. 196, 2021, Art. no. 108264.

[39] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang, "Intelligent resource scheduling for 5G radio access network slicing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7691–7703, Aug. 2019.

[40] "Movielens 1m dataset - Grouplens," Feb. 2003, Accessed: Jan. 08, 2021. [Online]. Available: https://grouplens.org/datasets/movielens/1m/

[41] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016. [Online]. Available: https://doi.org/10.1145/2827872

[42] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2862–2877, May 2021.

[43] "UAV Navigation in depth: How to measure the quality of the data link - UAV navigation." May 2019, Accessed: Jan. 08, 2021. [Online]. Available: https://www.uavnavigation.com/company/blog/uav-navigation-depth-how-measure-quality-datalink

[44] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, Mar. 2016, vol. 30, pp. 2094–2100. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10295

[45] P. Guo, X. Wang, and Y. Han, "The enhanced genetic algorithms for the optimization design," in *Proc. IEEE 3rd Int. Conf. Biomed. Eng. Inform.*, 2010, pp. 2990–2994.

Sadman Araf received the B.S. degree in computer science and engineering from BRAC University, Dhaka, Bangladesh. Afterwards, he was with BRAC University as a Lecturer. He is currently a Data Scientist for one of the leading telecommunication operator in Bangladesh. His research interests include edge computing, computer vision, reinforcement learning, and quantum computing.

Adittya Soukarjya Saha received the B.S. degree with a major in computer science and engineering and a minor in business studies and the M.Sc. degree in computer science and engineering from BRAC University, Dhaka, Bangladesh. He is currently working toward the Ph.D. degree in computer science with North Carolina State University, Raleigh, NC, USA. He was with BRAC University as a Lecturer. His research interests include edge computing, artificial intelligence, educational data mining, reinforcement learning, and human-computer interaction.

Sadia Hamid Kazi received the B.Sc. degree in engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, the M.Sc. degrees in computer information systems from Georgia State University, Atlanta, GA, USA, and the Ph.D. degree in information technology from International Islamic University Malaysia, Malaysia. She is currently the Chairperson of the Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh. She is also the Coordinator and Instructor of Cisco Networking Academy Program under BRAC University.

Nguyen H. Tran (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering from the HCMC University of Technology and Kyung Hee University, in 2005 and 2011, respectively. He was an Assistant Professor with Department of Computer Science and Engineering, Kyung Hee University, from 2012 to 2017. Since 2018, he has been with the School of Computer Science, The University of Sydney, where he is currently a Senior Lecturer. His research interests include distributed computing, machine learning, and networking. He received the best KHU thesis award in engineering in 2011 and several best paper awards, including IEEE ICC 2016 and ACM MSWiM 2019. He receives the Korea NRF Funding for Basic Science and Research 2016-2023 and ARC Discovery Project 2020-2023. He is the SOAR Fellow 2022. He was the Editor of IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING from 2016 to 2020, and the Associate Editor of IEEE Journal of Selected Areas in Communications 2020/2021 in the area of distributed machine learning/Federated Learning.

Md. Golam Rabiul Alam (Member, IEEE) received the B.S. degree in computer science and engineering and the M.S. degree in information technology, and the Ph.D. degree in computer science and engineering from Kyung Hee University, Seoul, South Korea, and completed postdoc afterwards. He is currently a Professor with BRAC University, Dhaka, Bangladesh,. His research interests include healthcare informatics, body area networks, ambient intelligence, and persuasive technology. He is a Member of IEEE IES, CES, CS, SPS, CIS, and ComSoc. He is also a Member of KIISE. He was the recipient of several best paper awards from prestigious conferences.