Joint System Regret and Cost Optimization in 6G UAV-Assisted SAGIN using H-DRL with Quantum Inspired Experience Replay

Saurabh Kumar Mishra, Mridul Athya, Ayush Chauhan, Ajay Pratap , *Member IEEE*
Department of CSE, Indian Institute of Technology (BHU), Varanasi
Email: {saurabhkumarmishra.rs.cse23, mridul.athya.cse23, ayush.chauhan.cse23, ajay.cse}@iitbhu.ac.in

***Abstract*—The emerging 6G Space-Air-Ground Integrated Network (SAGIN) leverages UAVs and satellites to extend content delivery in remote and dynamic environments. However, rising user demand leads to cache misses and elevated latency and content retrieval costs from Content Service Providers (CSPs). To address this, we formulate a joint optimization problem aimed at maximizing cache hit ratio and minimizing retrieval cost under UAV storage and transmission constraints. Our solution integrates content popularity prediction using an LSTM-based model with a cache placement strategy learned via Deep Reinforcement Learning. The proposed heuristic policy effectively mitigates the challenges posed by the large discrete action space, while the experience replay mechanism leverages quantum-inspired principles—specifically Grover's iteration—to dynamically adjust experience importance and prioritize the most informative samples, thereby enhancing learning efficiency. Simulation results on the MovieLens 1M dataset show that our method achieves a 5.5% and 5.2% higher cache hit ratio than H-DDQN-PER and H-DDQN-LRU respectively, whereas 19.8% improvement over H-DDQN-FIFO, while also reducing CSP communication costs by 52.3% , 52.2% and 35% respectively, validating the effectiveness of involving the qubit interaction among experiences to train Q-network in effective manner for better cache decisions.**

***Index Terms*—Proactive caching, system regret, communication cost, long short term memory (LSTM), heuristic double deep Q-Network (H-DDQN), quantum-inspired experience replay (QiER), movielens 1M dataset**

I. Introduction

WIRELESS networks have seen a rapid surge in data usage in recent years [1]. This is mainly driven by the growing use of smart mobile devices that support high-quality multimedia features, along with the increasing popularity of data-heavy services like video streaming. In particular, mobile video is expected to make up the largest portion of mobile data usage in the near future due to rising global demand [1]. Despite improvements in mobile network technologies, current cellular systems are struggling to keep pace with this explosive growth in data traffic [2]. To address this issue, researchers have suggested using content caching — an idea previously explored in wired networks [3]—to make wireless data delivery more efficient. This approach is becoming more practical not only because storage devices are becoming cheaper, but also because a small set of highly popular content tends to generate most of the overall traffic [4].

However, providing efficient wireless content delivery in remote or underserved regions remains a significant challenge, as traditional terrestrial cellular infrastructure is often absent, unreliable, or economically infeasible to deploy in those areas. Moreover, natural disasters or extreme geographical conditions can severely impact terrestrial connectivity, resulting in persistent communication blackouts. While ground communication systems may offer high-speed links under normal conditions, their vulnerability to environmental disruptions introduces coverage blind spots and service unreliability. To overcome these limitations, space-air-ground integrated networks incorporating Unmanned Aerial Vehicles (UAVs) and Low Earth Orbit (LEO) satellites have emerged as powerful enablers. Satellites provide wide-area backhaul links, ensuring resilient connectivity without geographical constraints, while UAVs act as agile, low-latency intermediaries, capable of dynamically positioning themselves closer to end users to enhance content delivery.

The utility of UAVs is further amplified when equipped with caching capabilities, allowing them to store and deliver popular content locally without repeated access to core networks. This approach significantly reduces latency, alleviates backhaul congestion, and enhances the Quality of Experience (QoE) for users in both remote and dense urban scenarios. But, as there is limited capacity and energy limitations, UAVs wii not be able to cache all available content. Moreover, the vast and continuously growing amount of multimedia content available makes it infeasible to store everything in local storages. Hence, machine learning algorithms are required to determine what content should be cached. A major challenge in this context is understanding and predicting popularity of the content. The popularity distribution of the material has a significant impact on the best cache content location, but when making a caching decisions are made, it is often unclear what users will request in the future. In many scenarios, not even a reliable estimate of the content popularity is available beforehand—it must instead be learned and computed locally by the caching entity itself. This is particularly important

because local popularity patterns may significantly deviate from global trends observed by content providers.
Thus, UAV-based caching systems must learn local content popularity through observation and user interaction over time to support proactive and adaptive cache placement. Additionally, user preferences are not uniform; different users favor different content, and these preferences fluctuate with changes in the mobile user population. This means proactive caching should consider the diversity of local content popularity, ensuring that the cache reflects the tastes of the users currently in the area. Context also plays a vital role—users' content consumption behavior can vary based on factors like location, time, mood, age, or even device characteristics. Therefore, context-aware caching is crucial, enabling the UAV to adapt stored content dynamically according to the contextual patterns of user demand.
In this emerging paradigm, UAVs function not only as mobile base stations but as intelligent caching entities, fetching selected content from Content Service Providers (CSPs) via satellite or terrestrial relays and delivering it to users proactively. A promising direction involves cluster-assisted UAV delivery, where UAVs operate collaboratively in clusters, coordinating their positions and cache contents to efficiently serve user groups with minimal overlap and maximum coverage. This cooperative approach enhances delivery efficiency and scalability, especially in scenarios involving high user density or rapidly changing demand. Thus, integrating proactive content caching with UAV clustering and satellite backhaul forms a robust architecture to support seamless, scalable, and context-aware wireless content delivery in next-generation networks.

Getting Motivation by the above given problems and opportunities, in this article, we form a machine learning–driven proactive content caching strategy for user-centric request behaviour, leveraging historical interaction data and deep reinforcement learning to address content popularity shifts and spatial-temporal user dynamics, in addition with quantum operations to select best experiences to make the DRL model learn more better. The following are our principal contributions:

1) A structured data preprocessing and feature engineering pipeline is designed using the MovieLens 1M dataset. The pipeline includes temporal discretization of user interactions, stratified user–item sampling, genre-based one-hot encoding, and demographic normalization to transform the raw dataset into a temporally and semantically aware format suitable for learning.
2) A predictive layer is constructed using a LSTM (Long Short-Term Memory) trained on sequential synthesized user–item pairs. The model consumes composite feature vectors—incorporating user demographics, movie metadata, and temporal context—to estimate user requests, producing a dense preference matrix that mitigates the cold-start problem and enhances cache decision accuracy.
3) We develop a Heuristic content placement policy using a Double Deep Q-Network with Quantum-Inspired Experience Replay (DDQN-QiER), a deep reinforcement learning–based framework that models the content caching task as a Markov decision process. This approach statically learns to place high-utility content into cache, optimizing both long-term reward and hit rate. To further enhance the training efficiency of deep reinforcement learning, we use a quantum-inspired experience replay (QiER) approach (DRL-QiER), which improves training performance naturally without requiring deliberate hyperparameter tuning. Experiences are represented in quantum form in DRL-QiER, and quantum operations, such as a preparation operation, are used iteratively to change their probability amplitudes. (which prioritizes experiences based on importance) and a depreciation operation (which accounts for the replay frequency of selected experiences). These operations ensure that the significance of experiences is properly distinguished while maintaining experience diversity, ultimately improving caching decision-making and overall system performance.

The rest of this paper is arranged as follows. The related works are discussed in Section II. In Section III, we introduce the system model and describe the problem formulation for proactive caching in UAV-assisted SAGIN. In Section IV, we present the LSTM model for content popularity prediction and heuristic-based modified DDQN with quantum-inspired experience replay for solving the formulated problem. Section V shows the performance analysis and comparisons with baselines. Finally, the conclusion and future works is given in Section VI.

II. Related Works

Recent research has revealed that some well-known content is repeatedly requested by users, contributing to the most data traffic [14]. This puts substantial strain on the network, especially considering that UAVs serve ground users (GUs)by establishing a wireless backhaul connection to the content server. Because of its low capacity, this backhaul link frequently causes a bottleneck with the rapid development of data traffic, which eventually degrades the user experience. Popular content caching at the network edge has become a viable remedy for this. especially during peak traffic periods. Edge caching not only reduces the load on backhaul links but also improves responsiveness and quality of service. Moreover, caching can significantly extend UAV lifespan by enabling them to pre-store popular files and serve repeated requests locally, thereby minimizing the need for constant backhaul communication [15]–[17].

Building on this foundation, several advanced strategies have been proposed to further optimize content delivery and resource usage. In [8], the authors address CSP cost minimization in 6G SAGIN via UAV caching and GU clustering.

TABLE I: Summary of related work

Paper	SAGIN	UAV	Content Popularity Prediction	Proactive Caching	DRL	Quantum Experience Replay	Latency	Cost
[5]	X	✓	X	X	X	X	X	X
[6]	X	✓	X	✓	X	X	X	X
[7]	X	X	✓	X	✓	X	✓	✓
[8]	✓	✓	X	X	X	X	✓	✓
[9]	X	✓	X	X	✓	X	✓	X
[10]	X	X	✓	✓	✓	X	X	✓
[11]	X	X	X	X	✓	✓	X	X
[12]	X	✓	X	X	✓	X	✓	X
[13]	✓	✓	✓	✓	✓	X	X	X
Ours	✓	✓	✓	✓	✓	✓	✓	✓

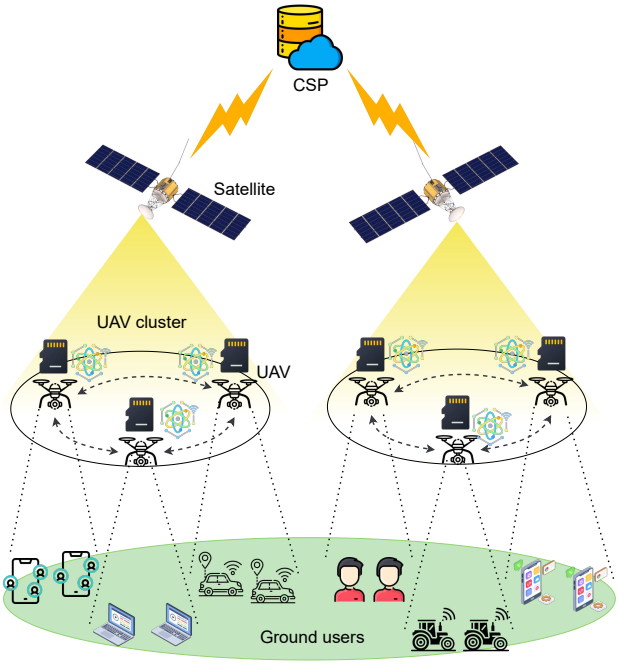

Fig. 1: System architecture.

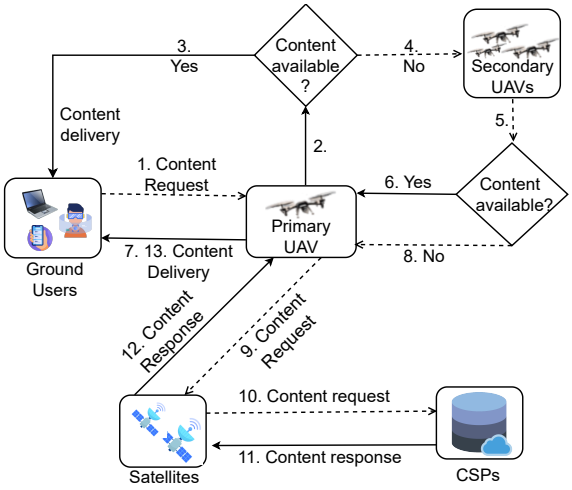

Fig. 2: Workflow.

They adopt game theory and genetic algorithms to reduce latency and improve content placement efficiency under fixed power constraints. [6] proposed a context-aware proactive content caching (m-CAC) scheme with service differentiation in wireless networks. By leveraging user context such as location and preferences, their approach reduces latency using a heuristic to solve the MINLP-based caching problem. A Q-learning-based collaborative caching algorithm tailored for vehicular clusters is proposed in [7]. This algorithm dynamically adapts to content popularity and user mobility, effectively reducing both latency and backhaul congestion. [9] introduced a deep reinforcement learning (DRL)-based mobility-aware caching strategy, which also integrates UAV trajectory optimization using the DDPG algorithm. This joint optimization improves both content delivery efficiency and UAV movement planning. A DRL-based user association and caching framework in fog networks is presented in [10]. Utilizing Q-learning with experience relay, their method predicts content popularity and reduces content delivery delay. To enhance DRL training, [11] introduced a Quantum-Inspired Experience Replay (DRL-QiER) mechanism. By sampling experiences based on TD error and replay frequency, their method significantly improves learning efficiency compared to DRL-PER and DCRL. A DRL-based caching strategy for UAVs and user devices is developed in [18]. Their system pre-caches popular content in an end-to-end UAV-supported network, improving overall delivery efficiency. [19] presented a DRL-driven cache placement optimization algorithm for UAV networks. Their solution determines optimal caching strategies that reduce content delivery time and improve QoS. In another notable study, [5] applied proactive caching directly at the GUs to address UAV endurance limitations. Their framework jointly optimizes caching policies, UAV path planning, and communication scheduling to enhance service deliv-

ery. In [12], authors proposed Fed-IDCCO, a federated deep reinforcement learning framework for joint data caching and computation offloading in UAV-assisted IoV. The approach minimizes task delay and maximizes cache hit ratio while preserving user privacy and improving training efficiency over baseline methods. Finally, [13] proposed a reinforcement learning framework that integrates content recommendation and proactive caching at the wireless edge to maximize the net profit of mobile network operators. By decomposing the large state-action optimization into two subproblems—a recommendation agent (enhancing user stickiness and revenue) and a pushing agent (minimizing transmission cost)—their double deep Q-network with dueling architecture achieved significant gains over baseline strategies.

III. System Model and Problem Formulation

As illustrated in Fig. 1, the proposed Space-Air-Ground Integrated Network (SAGIN) is designed to cover a designated geographical remote area. The system consists of several components, a set of satellites, denoted by $\mathcal{S} = \{1, \ldots, s, \ldots, S\}$, a set of Unmanned Aerial Vehicles (UAVs), represented by $\mathcal{U} = \{1, \ldots, u, \ldots, U\}$, and a Content Service Provider (CSP) denoted by k. Each UAV in the network is equipped with local memory to cache frequently accessed contents, enhancing service delivery and reducing latency. The content set $\mathcal{F} = \{1, \ldots, f, \ldots, F\}$ is distributed across the CSPs, and there are N Ground Users (GUs) represented by the set $\mathcal{GU} = \{1, \ldots, i, \ldots, N\}$. Each GU is associated with a single UAV, which serves as its direct communication node for content delivery. Multiple users can share the same UAV as their primary connection point. To improve efficiency and reliability in content distribution, UAVs are grouped into S clusters.

Within each cluster, UAVs can cooperate to fulfill user requests by sharing cached content. If a requested content is not available on a user's primary UAV, the system allows the request to be forwarded to one of the secondary UAVs—that is, other UAVs within the same cluster that are not directly associated with the user. These secondary UAVs support the content delivery process by acting as backup providers, enabling seamless transfer of data and ensuring that user demands are met even in the absence of local availability on the primary UAV. The operation of the network proceeds in discrete updating periods, indexed by t. During each period, every UAV simultaneously serves a fixed subset of ground users, denoted by M_u, which remains unchanged throughout the duration of the period. UAVs are equipped with a single antenna and operate either as relay stations or direct service providers based on the availability of requested content.

The primary UAV initially determines if the content is locally cached when a user requests it. If the content is transmitted directly to the user. However, if the content is not available on the primary UAV, the request is forwarded to another UAV within the same cluster that has the content cached. This inter-UAV communication ensures reliable delivery and minimizes delays by leveraging the cooperative structure of the network.If no UAV in the current UAV cluster has the content then, the primary UAV passes the content to the GU after receiving it from a CSP via satellite as a relay.

A. Channel Model

The channel between satellite and UAV is [20], [21]:

$$CG_{s,u} = [h_{s,u}]^T, \tag{1}$$

where, $h_{s,u} = \sqrt{g_{s,u} d_{s,u}^{-\alpha^{(1)}}}$ is the satellite-to-UAV channel gain. $\alpha^{(1)}$ represents the path loss exponent between a satellite and a UAV, and $d_{s,u}$ indicates the distance between the s^{th} satellite and the u^{th} UAV and $g_{s,u}$ is the fading factor between satellite and UAV. Due to the page constraint we did not include the channel gain formula for CSP to satellite i.e., $CG_{k,s}$. However, it is considered based on Eq. (1) by changing the respective parameters [22].

B. Transmission Model

The data rate between CSP and satellite is given as [22]:

$$R_{k,s} = B_1 \log_2 \left(1 + \frac{CG_{k,s} P_k}{\sigma_s^2}\right), \tag{2}$$

where B_1 is the transmission bandwidth, P_k is the transmit power of CSP, $CG_{k,s}$ is channel gain between CSP and satellite, σ_s^2 is the noise power. The data rate between satellite s and UAV u ($R_{s,u}$) is considered based on Eq. (2) by changing the respective parameters [22].

C. Caching Strategy and Content Popularity Prediction

To efficiently manage content delivery in UAV-assisted networks, we define an association between UAVs and GUs as binary association matrix at time t, denoted by $\mathcal{X}(t)$, where each element $x_{u,i}(t)$ represents the association between UAV u and GU i as:

$$x_{u,i}(t) = \begin{cases} 1, & \text{if the } u^{\text{th}} \text{ UAV is the primary UAV of GU } i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Here, u denotes the primary UAV, and u' represents a secondary UAV within the same cluster. For simplicity, the request packets used by GUs are considered negligible in size compared to the content files, and we assume error-free transmissions. We assume that GU i requests the f^{th} content of size Z bits from its associated UAV u. Satellites have access to all contents from CSP and transmit them to UAVs. Due to limited storage, each UAV can cache up to F' contents at a time, where $F' \ll F$, and F is the total number of available contents.

We define the content popularity $q_i^f(t)$ as the accumulated request behavior for content f by GU i in the update period t. The actual content popularity $Q_u^f(t)$ of content f for all GUs primarily served by UAV u is defined as follows [23]:

$$Q_u^f(t) = \sum_{i \in \mathcal{GU}} x_{u,i}(t) q_i^f(t), \quad \forall f \in \mathcal{F} \tag{4}$$

It is necessary to forecast each content's overall popularity using the historical request data in order to implement proactive content caching during the updating period t. As a result, we indicate$\hat{Q}_u^f(t)$ as the total predicted popularity of content f for UAV u which is the accumulated predicted request for a particular content f for all GUs served by UAV u. Efficient content caching is a crucial aspect of modern wireless networks and edge computing systems. However, a real challenge lies in accurately predicting the popularity of content and ranking it accordingly. To address this, we define the ranking function as [23]:

$$\text{rank}(\hat{Q}_u^f(t), \mathcal{F}) = \sum_{f' \in \mathcal{F}} \mathbb{I}\left(\hat{Q}_u^f(t) \leq \hat{Q}_u^{f'}(t)\right), \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the condition inside holds true and 0 otherwise. This function counts how many other contents f' in $\mathcal{F}$ have an estimated popularity equal to or greater than content f.

The rank function is important because it helps decide which content should be saved in the cache. If the rank is low (closer to 1), it means the content is popular and should be stored. If the rank is high, it means the content is less popular and might not be kept in the cache. By leveraging this ranking mechanism, the caching system is dynamically updating stored content based on real-time popularity trends, thereby optimizing cache efficiency and network performance.

We define the cache placement vector for a particular UAV u in updating period t, denoted as $\boldsymbol{\Gamma_u(t)}$, as the binary vector representing the cached state of content f in updating period t, determined based on the predicted total popularity while satisfying the maximum cache capability constraint F' and the transmission capacity constraint $\boldsymbol{L_u(t)}$. Here, the transmission capacity constraint is the maximum number of contents that can be retrieved from the CSP depending on available battery and power resources of a UAV. Mathematically, $\boldsymbol{\Gamma_u(t)}$ is defined as , $\boldsymbol{\Gamma_u(t)} = \{\Gamma_{u,f}(t) \mid f \in \mathcal{F}\}$, where $\Gamma_{u,f}(t)$ is a binary indicator denoting whether the f-th content is stored in the cache of UAV u, with $\Gamma_{u,f}(t) \in \{0, 1\}$. Here, a value of 1 means the content is stored in the cache of UAV u, while 0 means the content is not stored in that UAV. Each UAV u has a cache storage limit, expressed as:

$$\sum_{f \in \mathcal{F}} \Gamma_{u,f}(t) \leq F'. \tag{6}$$

However, without considering the transmission constraint, the ideal cached content vector $\tilde{\boldsymbol{\Gamma}}_u(t)$ depends only on the predicted popularity ranking, defined by

$$\tilde{\Gamma}_u(t) \triangleq \left\{\tilde{\Gamma}_{u,f}(t) = \mathbb{I}\left(\text{rank}(\hat{Q}_u^f(t), \mathcal{F}) \leq F'\right) \mid f \in \mathcal{F}\right\}.$$

We define the cache update load as the number of contents that must be updated in the cache, which is given as:

$$\Delta\Gamma_u(t) = \sum_{f \in \mathcal{F}} \left|\boldsymbol{\tilde{\Gamma}_{u,f}(t) - \Gamma_{u,f}(t-1)}\right|, \tag{7}$$

Then, the updated cache placement vector $\boldsymbol{\Gamma_u(t)}$ is determined as follows:

- If $L_u(t) \geq \Delta\Gamma_u(t)$: The UAV has sufficient transmission resources to retrieve all required files from the CSP via satellite, then: $\boldsymbol{\Gamma_u(t) = \tilde{\Gamma}_u(t)}$, meaning that the files with the top F' predicted popularity ranking will be cached.
- If $L_u(t) < \Delta\Gamma_u(t)$: The UAV can retrieve at most $L_u(t)$ files from the CSP, then: $\boldsymbol{\Gamma_u(t) = \Gamma_{u,1}(t) + \Gamma_{u,2}(t)}$, in which :

$$\Gamma_{u,1,f}(t) = \mathbb{I}\left(\text{rank}\left(\hat{Q}_u^f(t), \mathcal{F}_{prev}\right) \leq F' - L_u(t)\right),$$
$$\Gamma_{u,2,f}(t) = \mathbb{I}\left(\text{rank}\left(\hat{Q}_u^f(t), \mathcal{F} \setminus \mathcal{F}_{prev}\right) \leq L_u(t)\right),$$

  where $\mathcal{F}_{prev} = \{f \in \mathcal{F} \mid \boldsymbol{\Gamma_{u,f}(t-1)} = 1\}$ denotes the set of contents cached at previous time slot. Here, $\boldsymbol{\Gamma_{u,1}(t)}$ represents the files retained from the previous cache at UAV u. Meanwhile, $\boldsymbol{\Gamma_{u,2}(t)}$ represents the $L_u(t)$ files retrieved from the CSP at time t.

Then, the network fulfills user requests for content in $\mathcal{F}$ via the following three phases:

- **Phase 1: Direct Cache Delivery**
  For content $f \in \mathcal{F}$ such that $\Gamma_{u,f}(t) = 1$, the requested content is proactively cached at UAV u and can be directly delivered to the user.
- **Phase 2: Cluster-Assisted Delivery**
  For content $f \in \mathcal{F}$ such that $\Gamma_{u,f}(t) = 0$, but $\Gamma_{u',f}(t) = 1$ for any UAV u' that belongs to the same cluster as UAV u, the requested content is first retrieved from UAV u' and then delivered to the user through UAV u.
- **Phase 3: Content Retrieval from CSP via Satellite**
  For content $f \in \mathcal{F}$ such that $\Gamma_{u,f}(t) = 0$ and $\Gamma_{u',f}(t) = 0$ for all UAVs u' in the same cluster — where a cluster is defined as a group of UAVs that are jointly served by a common satellite and are responsible for delivering content to their associated ground users — the requested content must be retrieved from the CSP via satellite transmission.

Given the cache placement vector $\boldsymbol{\Gamma_u(t)}$, the total cache hit ratio of these cached contents, denoted as $\mathcal{H}_u(t)$, is calculated as:

$$\mathcal{H}_u(t) = \frac{\sum_{f \in \mathcal{F}} \left(\Gamma_{u,f}(t) + (1 - \Gamma_{u,f}(t)) \max_{u' \in \mathcal{U}_s} \Gamma_{u',f}(t)\right) \hat{Q}_u^f(t)}{\sum_{f' \in \mathcal{F}} \hat{Q}_u^{f'}(t)}. \tag{8}$$

To evaluate the performance of the system, we define system regret as the cache miss in updating period t, denoted as $\mathcal{R}(t)$, which is the loss of the total cache hit ratio from cache vectors $\boldsymbol{\Gamma_u(t)}$ of contents that are cached for each UAV. For a single UAV u, the regret is given by:

$$\mathcal{R}_u(t) = 1 - \mathcal{H}_u(t). \tag{9}$$

TABLE II: List of Notation

Symbol	Meaning	Symbol	Meaning
$\mathcal{S}$	Set of Satellites	$\mathcal{U}$	Set of UAVs
C	Set of Content Service Providers	$\mathcal{F}$	Set of Content
$\mathcal{GU}$	Set of Ground Users	s	Index of Satellite
u	Index of UAV	k	Representation of a CSP
f	Index of Content	i	Index of User
M_u	Maximum limit of users a UAV can serve	$CG_{i,j}$	Channel Gain between i and j
$R_{i,j}$	Data rate between i and j	$\mathcal{U}_s$	Cluster for UAVs of satellite s
N	Total number of ground users	F	Total number of contents/files
$q^f_{i,u}(t)$	Accumulated request count for content f by GU i for UAV u in time slice t	$Q^f_u(t)$	Total actual popularity of content f for all $\mathcal{GU}$ of UAV u in time slice t
$\hat{Q}^f_u(t)$	Predicted content popularity of f for all $\mathcal{GU}$ of UAV u in time slice t	$L_u(t)$	Transmission capacity of a UAV
$\Gamma_{u,f}(t)$	Binary variable denoting content f is cached at UAV u in time slice t	F'	Storage capacity of a UAV
		$\mathcal{H}_u(t)$	Cache hit for a UAV u
$\mathcal{R}(t)$	System regret	$x_{u,i}$	Binary variable for User-UAV association
$\mathcal{X}$	User-UAV Association Matrix	$d_{i,j}$	Distance between i and j
$\mathcal{F}_{prev}$	Set of content cached at UAV u at time t-1	C_u	Total cost that UAV has to pay for content retrieval from CSP via satellite
$T_{i,j}$	Latency between i and j	$N_{i,j}$	Number of files transferred between i and j
$D_{k,u}$	Data transmitted between k and u	$C_{k,u}$	Cost for communication between k and u

Extending this to all UAVs in the system, the total system regret is expressed as:

$$\boldsymbol{\mathcal{R}(t)} = \sum_{u \in \mathcal{U}} \mathcal{R}_u(t). \tag{10}$$

Now, when a user request for content results in a cache miss, i.e., the requested file is unavailable at both the primary UAV and its neighboring UAVs within the same cluster, the content must be fetched directly from CSP via satellite. This retrieval involves data transmission over satellite and backhaul links, thereby incurring both latency and communication cost. To comprehensively evaluate system performance under such cache miss scenarios, we analyze the total latency experienced during CSP retrieval and quantify the associated communication cost required to cache the missing content at the UAV.

D. Latency and Communication Cost

If the required content f is not present in the cache of either the primary UAV u or the secondary UAV u', then the content has to be bought from CSP. The latency for bringing that content from CSP via satellite as a relay is formulated as [8]:

$$T_{k,u} = T_{k,s} + T_{s,u}, \tag{11}$$

where,

$$T_{k,s} = \frac{2d_{k,s}}{c} + \frac{N_{k,s}Z}{R_{k,s}}. \tag{12}$$

$$T_{s,u} = \frac{2d_{s,u}}{c} + \frac{N_{s,u}Z}{R_{s,u}}, \tag{13}$$

By Eqs. (11), (12), and (13) we write $T_{k,u}$ as [8]:

$$T_{k,u} = \frac{2(d_{k,s} + d_{s,u})}{c} + \frac{N_{s,u}Z}{R_{s,u}} + \frac{N_{k,s}Z}{R_{k,s}}, \tag{14}$$

where, $N_{s,u}$, $N_{k,s}$ represent the number of files transferred between satellite s and UAV u, and CSP k and satellite s respectively. When data is transmitted from CSP to UAV , the cost borne for the communication via backhaul(satellite) is given by [8]:

$$C_{k,u} = p^k D_{k,u}, \tag{15}$$

where, p^k is the cost per bit for data transfer and $D_{k,u}$ is the size of data to be transferred from CSP to UAV, defined as [8],

$$D_{k,u} = R_{s,u}T_{s,u} + R_{k,s}T_{k,s}. \tag{16}$$

Now, for completely caching the missing contents on UAV u, the total communication cost is formulated as:

$$C_u(t) = \sum_{f \in \tilde{\mathcal{F}}} \left((1 - \Gamma_{u,f}(t))(1 - \max_{u' \in \mathcal{U}_s} \Gamma_{u',f}(t)) C_{k,u} \right) \tag{17}$$

where $\tilde{\mathcal{F}}$ is the set of contents which are to be fetched from CSP via satellite in case of cache miss from the cluster, which is defined as $\tilde{\mathcal{F}} = \{f \in \mathcal{F} \mid \text{rank}\left(Q^f_u(t), \mathcal{F} \setminus \mathcal{F}_{prev}\right) \leq L_u(t)\}$. Extending this to all UAVs in the system, the total system cost is expressed as:

$$\boldsymbol{\mathcal{C}(t)} = \sum_{u \in \mathcal{U}} \mathcal{C}_u(t). \tag{18}$$

In practice, it is crucial to create a balance between minimizing the cache miss rate and reducing the communication cost incurred during content retrieval from the CSP via satellite

backhaul links. To address the tradeoff between the cache hit ratio and communication cost for content retrieval in case of cache miss, we formulate a joint optimization framework that aims to minimize the overall system cost by effectively managing content placement decisions. The objective is to reduce the system regret incurred due to suboptimal cache hits while simultaneously minimizing the retrieval cost from CSP in the event of cache misses. The optimization problem is expressed as follows:

$$\text{P1}: \quad \min_{\boldsymbol{\Gamma(t)}} \boldsymbol{\mathcal{R}(t)} + \boldsymbol{C}^{\text{norm}}\boldsymbol{(t)} \tag{19}$$

subject to the following constraints:

$$\Gamma_{u,f}(t) \in \{0,1\}, \qquad \forall u \in \mathcal{U}, f \in \mathcal{F}, \tag{19a}$$

$$\sum_{f \in \mathcal{F}} \Gamma_{u,f}(t) \leq F', \quad \forall u \in \mathcal{U}, \tag{19b}$$

$$\Delta\Gamma_u(t) \leq L_u(t), \quad \forall u \in \mathcal{U} \tag{19c}$$

, where $\boldsymbol{C}^{\text{norm}}\boldsymbol{(t)}$ is the normalized cost. To ensure consistent scaling across different metrics, we normalize the cost using min-max normalization, computed as $\boldsymbol{C}^{\text{norm}}\boldsymbol{(t)} = \frac{\boldsymbol{C}(t) - \min \boldsymbol{C}(t)}{\max \boldsymbol{C}(t) - \min \boldsymbol{C}(t)}$. This normalization is necessary because the system regret (i.e., cache miss) for each UAV is naturally bounded between 0 and 1, while the cost values can span a wider numerical range. By applying Min-Max normalization, we scale the cost into the same range $[0, 1]$ for balanced optimization.

IV. Proposed Solution

To address the above joint regret and cost minimization problem, we propose an integrated prediction–decision framework. This section is divided into two parts: (i) an LSTM-based content popularity prediction approach, and (ii) a Heuristic Deep Reinforcement Learning (DRL) inspired framework for cache placement.

A. Long Short Term Memory (LSTM)

The Long Short-Term Memory (LSTM) network [24], a prominent type of recurrent neural network (RNN), is extensively employed for modeling sequential data. It has demonstrated remarkable performance in a variety of tasks, including speech recognition [25], machine translation [26], syntactic parsing [27], and image captioning [28]. Owing to its effectiveness in capturing temporal dependencies, the LSTM architecture is particularly well-suited for our content request prediction task. In the following subsection, we briefly describe the structure of the LSTM network.

The input gate g_t, forget gate f_t, and output gate o_t are built as sigmoid units to selectively pass information, and each of the memory blocks that make up the LSTM network contains a memory cell that is controlled by these three gates. The input gate g_t decides how much new information is supplied to create the updated cell state c_t, while the forget gate f_t regulates how much of the prior cell state c_{t-1} is kept. Through recurrent connections, the cell output h_t and c_t are transmitted to the following time step after being modulated by the output gate o_t. The LSTM generates an output h_t and changes its cell state c_t at each time t. This output is used to forecast the content request $\hat{q}_i^f(t)$, based on the previous state c_{t-1}, previous output h_{t-1}, and the current input x_t (i.e., the normalized content request $\bar{q}_i^f(t-1)$). Given the limited number of user requests, which makes individual demand hard to forecast accurately, we instead predict the aggregate number of requests for each content item per time slot, where $Q_u^f(t)$ denotes the total **actual popularity** of content f across all users served by UAV u.

Let $\mathbf{p}_{i,f,t} \in \mathbb{R}^\rho$ represent a ρ-dimensional integrated feature vector that captures the historical request behavior of GU i for content f at time slot t. This vector is defined as: $\mathbf{p}_{i,f,t} = [q_i^f(t-\rho), q_i^f(t-\rho+1), \ldots, q_i^f(t-1)]$, where ρ denotes the number of preceding time slots considered. Hence, $\mathbf{p}_{i,f,t}$ aggregates the past ρ content requests made by GU i for content f.

To mitigate the influence of varying magnitudes in request values, we normalize this feature vector. Let $\bar{\mathbf{p}}_{i,f,t}$ denote the normalized version of $\mathbf{p}_{i,f,t}$, given by: $\bar{\mathbf{p}}_{i,f,t} = \left[\bar{q}_i^f(t-\rho), \bar{q}_i^f(t-\rho+1), \ldots, \bar{q}_i^f(t-1)\right]$, where each normalized request is computed as: $\bar{q}_i^f(t-\rho) = \frac{q_i^f(t-\rho)}{\max\{\mathbf{p}_{i,f,t}\}}$, and $\max\{\mathbf{p}_{i,f,t}\}$ denotes the maximum entry within the feature vector $\mathbf{p}_{i,f,t}$.

For each UAV u, we employ an LSTM network consisting of three hidden LSTM layers, along with an input and an output layer, to predict future content popularity. Let $\mathcal{L}_u(\cdot)$ be the output function of this LSTM network, parameterized by weights θ_u. At the start of time slot t, the normalized input vectors $\bar{\mathbf{p}}_{i,f,t}$ are generated using the aforementioned procedure. The predicted request (popularity) of content f by GU i at time t, denoted $\hat{q}_i^f(t)$, is then obtained as follows [29]:

$$\hat{q}_i^f(t) = \max\{\bar{\mathbf{p}}_{i,f,t}\} \cdot \mathcal{L}_u(\bar{\mathbf{p}}_{i,f,t}|\theta_u). \tag{20}$$

The LSTM parameters θ_u are trained by minimizing the squared loss between the predicted value $\hat{q}_i^f(t)$ and the actual number of requests $q_i^f(t)$, where the target is also normalized. The corresponding loss function is defined as [29]:

$$\text{Loss}(\theta_u) = \left(\mathcal{L}_u(\bar{\mathbf{p}}_{u,t,f}|\theta_u) - \frac{q_i^f(t)}{\max\{\mathbf{p}_{u,f,t}\}} \right)^2. \tag{21}$$

After computing the predicted popularity $\hat{q}_i^f(t)$ for each GU i associated with UAV u, the total predicted popularity of content f across all connected users at time slot t, denoted as $\hat{Q}_u^f(t)$, is obtained using Eq. 4.

Algorithm 1 illustrates a decentralized LSTM-based framework for predicting content popularity at each UAV. Each UAV u maintains its own LSTM network $\mathcal{L}_u(\cdot)$, trained using local historical data of connected users. At the beginning of

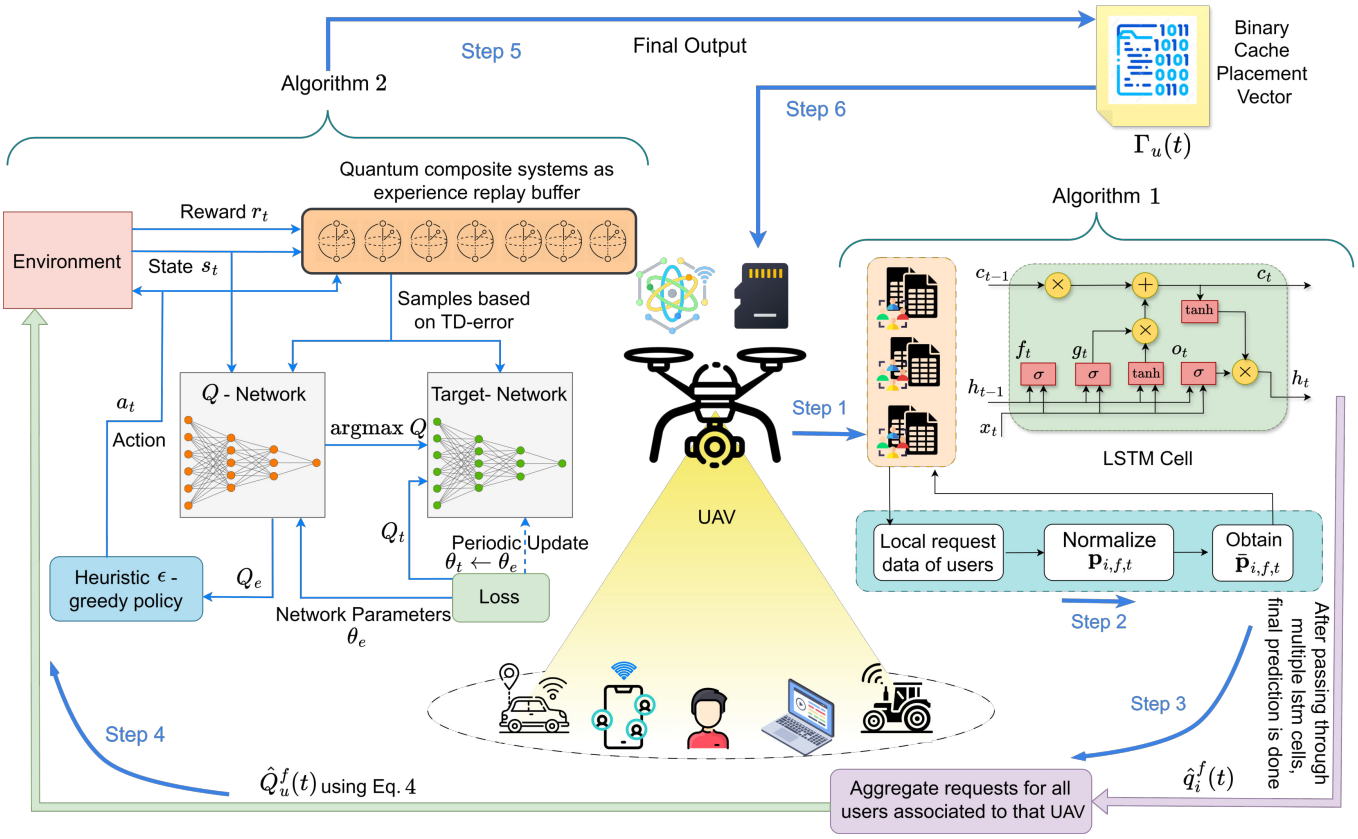

Fig. 3: UAV sequential workflow

Algorithm 1: LSTM prediction for content popularity

Initialize: LSTM networks $\mathcal{L}_u(\bar{\mathbf{p}}|\theta_u)$ with weights θ_u

1 **for** $t = \rho + 1$ *to* T **do**

`// Phase 1: Content popularity prediction at start of time slot t`

2 **for** $f = 1$ *to* F **do**

3 Generate and normalize feature vectors $\mathbf{p}_{i,f,t}$ using historical requests and obtain $\bar{\mathbf{p}}_{i,f,t}$;

4 Use the LSTM network to predict the popularity based on eq. [20];

`// Phase 2: Train LSTM network at the end of time slot t`

5 **for** $u = 1$ *to* U **do**

6 Train LSTM network L_u by minimizing the loss function [21];

Output : Predicted content popularity $\hat{q}_i^f(t)$

each time slot t, the UAV generates a normalized feature vector $\bar{\mathbf{p}}_{i,f,t}$, integrating user- and content-specific attributes for each content f (line-3). This vector is input into the LSTM network to predict the popularity $\hat{q}_i^f(t)$, as defined in eq. (20) (line-4). After observing the actual number of content requests, the model is trained by minimizing the squared error loss between predicted and actual normalized popularity values eq. (21) (line-6).

B. RL framework for content cache placement

Reinforcement learning (RL) adeptly maintains a balance between exploration and exploitation through learning strategies, obviating the necessity for preexisting data samples. By continuously observing environmental feedback, RL acquires instructive information for actions, maximizing intelligent agent rewards. Consequently, RL exhibits characteristics of iterative experimentation and delayed reward, typically amenable to modeling as an MDP. For challenges associated with intricate state spaces and discontinuities, conventional methods face difficulties in attaining the optimal policy function. Deep Reinforcement Learning (DRL) through interactive engagement with the environment, excels in furnishing nearly optimal solutions for complex decision problems.

The RL framework can be primarily described as consisting of an agent, an environment, a state set S, an action set A, state transitions P, and rewards $\mathcal{R} : S \times A \to \mathbb{R}$. The agent, by observing the state of the environment, makes decisions through learning and interaction with the environment within discrete time steps. Starting from an initial state s_0, the agent

observes the environment's state and takes action $a_0 \in A$, and the agent computes the reward $R(s_0, a_0)$ as the state transitions from s_0 to s_1.

The action-value function $Q_\pi(s,a)$ is used to represent the expected return under a policy $\pi(s) : S \to A$, which indicates the expected cumulative rewards:

$$Q_\pi(s,a) = E[R(s_0,a_0) + \gamma R(s_1,a_1) + \gamma^2 R(s_2,a_2) + \cdots [s,a] \quad (22)$$

where the discount factor $\gamma \in [0,1]$ is used to gradually converge the rewards.

For a given state s, the optimal value function represents the maximum possible cumulative reward that can be obtained among all policies. Therefore, the Bellman equation for the optimal value function is given by:

$$Q'_\pi(s,a) = R(s,a) + \max_a \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) Q'_\pi(s',a'). \quad (23)$$

The optimal strategy is:

$$\pi'(s) = \arg\max_a \sum_{s' \in \mathcal{S}} P(s'|s,a) Q'_\pi(s',a'). \quad (24)$$

The objective of the RL agent is to maximize the expected cumulative reward by selecting actions. To utilize DRL for solving problem 19, the RL framework for the cache placement decision problem is defined as follows.

1) State: The state of the system at time slot t is defined as:

$$s_t = [\boldsymbol{\Gamma(t), \Gamma(t-1)}], \quad (25)$$

where $\boldsymbol{\Gamma(t-1)}$ is the cache status of all UAVs in the previous time slot denoted as $\boldsymbol{\Gamma(t-1)} = \{\Gamma_1(t-1), \Gamma_2(t-1), \ldots, \Gamma_U(t-1)\}$, and $\boldsymbol{\Gamma(t)} = \{\Gamma_1(t), \Gamma_2(t), \ldots \Gamma_U(t)\}$ is the cache status of all the UAVs in the current time slot t.

2) Action: The action at time slot t, denoted as a_t, is defined as the cache decision binary variable $\Gamma_{u,f}(t) = \{0,1\}$, which represents the cache decision in vector for each UAV for each content.

3) Reward: At time slot t, the payoff is the negative of the sum of expected system regret of the system of all UAVs in case of cache hit and the normalized transmission cost for caching the content in UAV storage from CSP via satellite as relay when cache miss happens(19), which is observed at the end of time slot t, is given by :

$$r_t(s_t, a_t) = -[\mathcal{R}(t) + C^{\text{norm}}(t)] \quad (26)$$

C. Content Placement using Heuristic-Randomized DDQN Framework

According to the the analysis of state and action space, the possible action a_t representing the binary cache variable $\Gamma_{u,f}(t)$ for a content f at UAV u are 0 and 1. Therefore, The total number of feasible joint actions is then given by $|\mathcal{A}| = \prod_{u=1}^{U} \binom{F}{F'}$, which is super-exponential in both U and F, rendering exhaustive search or tabular reinforcement learning **intractable** [30]. To address this difficulty, we exploit the $\epsilon-$ greedy policy as heuristic-randomized policy, defined by [31] :

$$\pi_H(s_t) = \begin{cases} \text{top}_{F'} \left(\underset{f \in \mathcal{F}}{\text{argsort}} \left(Q(s_t, a; \theta)[u, f] \right) \quad \forall u \in \mathcal{U} \right), & \text{with prob. } (\\ \text{Random}(\tilde{\mathcal{A}}), & \text{with prob. } \epsilon \end{cases} \quad (27)$$

where $\tilde{\mathcal{A}}$ is any possible cache state which is feasible constrainted with (19b).

To solve the problem, DDQN introduces an interaction between the Q-network and the target network, effectively decoupling action selection from Q-value [32]. The Q network is used for action selection, and the target network is used to calculate the target value Q.In contrast to DQN, the target value calculation procedure is modified by first selecting the action A that corresponds to the maximum Q value using the Q network, and then using the target network to determine the Q value corresponding to the action A selected by the Q network, so that the Q value selected each time is not necessarily always the maximum, and avoid always selecting the overestimated suboptimal action. θ_e represents the Q-network , which is continuously updated and θ_t is periodically replaced with the latest θ_e to stabilize learning and reduce variance in the target value in order to improve the network convergence. The update of the parameter θ follows the gradient descent method, based on the mean squared loss function [33].

D. Experience replay

Traditional experience replay uniformly samples transitions from a replay buffer to break correlations between samples, but it fails to differentiate their importance. As a result, infrequent yet highly informative transitions may be sampled less frequently, reducing their contribution to learning. To address this issue, We employ a Quantum-inspired Experience Replay (QiER) mechanism to enhance the training efficiency of the Double Deep Q-Network (DDQN) agent. QiER introduces a priority-based sampling method that evaluates the importance of each transition based on its TD-error, or temporal difference error. The difference between the goal and forecasted Q-values is measured by the TD-error. A larger TD-error indicates that the corresponding transition is more informative for learning, and hence should be assigned a higher priority [33].

We maintain a fixed-size experience buffer of capacity D [34], where each new transition e_t is assigned an index $k \in \{1, \ldots, D\}$ and a corresponding priority value P_k, resulting in a prioritized buffer represented as $\{\langle e_t, P_k \rangle\}$. The experience replay process then consists of two stages: storing transitions into the buffer and sampling a minibatch from it based on the computed priorities. Transitions with higher TD-errors are sampled with greater probability, ensuring that more

valuable experiences are emphasized during training, thereby improving convergence and policy performance.

E. Quantum computations

In quantum computing, the basic unit of information is the quantum bit, or qubit. Unlike classical bits, a qubit exists in a superposition of two basic states, $|0\rangle$ and $|1\rangle$ [35]. This superposition can be expressed as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{28}$$

where α and β are complex numbers, which satisfy $|\alpha|^2 + |\beta|^2 = 1$. When a qubit in state $|\psi\rangle$ is measured, it collapses to $|0\rangle$ with probability $|\alpha|^2$, or to $|1\rangle$ with probability $|\beta|^2$. These coefficients can also be written using the inner product as $\alpha = \langle 0|\psi\rangle$ and $\beta = \langle 1|\psi\rangle$ [35]. Transformations in quantum systems are done using unitary operations. A unitary operation changes a qubit from its current state $|\psi\rangle$ to a new state $|\psi'\rangle$:

$$|\psi'\rangle = U\,|\psi\rangle \tag{29}$$

where U is a unitary matrix satisfying $U^\dagger U = UU^\dagger = I$. For example, a Hadamard gate transforms $|0\rangle$ to $(|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle$ to $(|0\rangle - |1\rangle)/\sqrt{2}$, which is formulated as [35], $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Our proposed framework, DRL-QiER, is illustrated in Fig. 4. In each learning cycle, the agent interacts with the environment and stores the resulting transition e_t at time step t. This transition is then converted into a quantum representation, specifically assigned to the k-th qubit in the experience buffer, where k is its index in the replay buffer. Next, this qubit is initialized into a superposition state using a preparation step (IV-G1). Transitions are then selected based on their importance, with higher-probability samples being more likely to be drawn. These selected transitions make up the minibatch used to train the Q-network. After each training step, the quantum states of the chosen transitions are adjusted. This adjustment involves two key unitary operations: one updates the quantum states to reflect the new TD-errors, and the other reduces the selection chance of overused transitions. This process continues in a loop until the learning algorithm converges, with implementation details discussed in the following sub-sections.

F. Quantum representation of experiences

In experience replay, each experience is modeled as a quantum bit (qubit), where the two basis states, $|0\rangle$ and $|1\rangle$, correspond to the decisions of *discarding* and *selecting* the experience, respectively. During training, the agent interacts with its environment, which is typically represented as an MDP, or Markov Decision Process. Every each step t, given the current state s_t, the agent selects an action a_t following a specific exploration strategy (in our case, heuristic randomized policy *(see 27)*). As a result, the environment moves to a new state s_{t+1} and returns a reward r_t. These elements

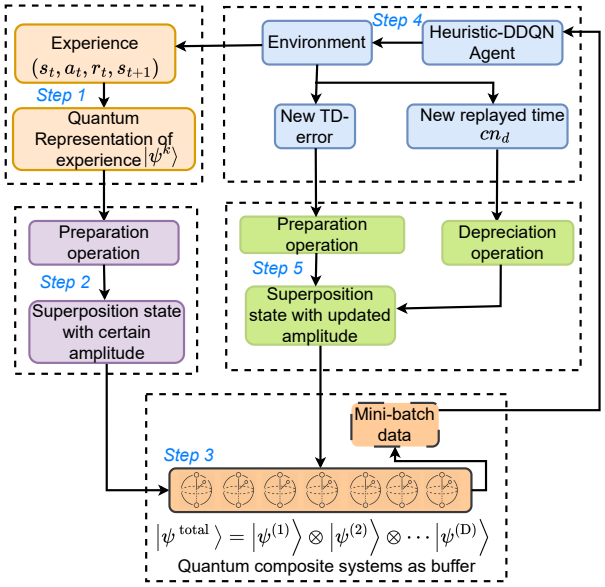

Fig. 4: Quantum operation and experience sampling framework

form a transition tuple (s_t, a_t, r_t, s_{t+1}), which is stored in a replay buffer and assigned an index k indicating its position in replay buffer. To convert this transition into a quantum form, we represent the possibility of *selecting* or *discarding* it using two basis states, $|1\rangle$ and $|0\rangle$, respectively. This transition is then treated as a qubit with the state defined by [34]:

$$\left|\psi^{(k)}\right\rangle = b_0^{(k)}\,|0\rangle + b_1^{(k)}\,|1\rangle \tag{30}$$

where the coefficients $b_0^{(k)}$ and $b_1^{(k)}$ are complex probability amplitudes satisfying $|b_0^{(k)}|^2 + |b_1^{(k)}|^2 = 1$. Specifically, the likelihood of rejecting the transition is given by: $|b_0^{(k)}|^2 = |\langle 0|\psi^{(k)}\rangle|^2$, while the likelihood of selecting it is $|b_1^{(k)}|^2 = |\langle 1|\psi^{(k)}\rangle|^2$. These coefficients reflect how important an experience is, and before its importance is fully known, it is reasonable to initialize the qubit in a known state and let it evolve towards a meaningful configuration.

A commonly used initial configuration in quantum computing is the uniform superposition state, expressed as [34], $|\psi_0\rangle = \frac{\sqrt{2}}{2}(|0\rangle + |1\rangle)$. The uniform state assigns equal probability to both eigenstates, reflecting a condition of maximum uncertainty. As a result, it serves as a suitable initial state for representing each experience. To modify the qubit state in (30), a rotation operator is applied. This operator is a key component of Grover's iteration [36] [37], and is defined as [34]:

$$U_\Phi = e^{-i\Phi Y} = \begin{bmatrix} \cos(\Phi) & -\sin(\Phi) \\ \sin(\Phi) & \cos(\Phi) \end{bmatrix} \tag{31}$$

where Φ is the rotation angle, and the Pauli-Y matrix is given by: $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$. Applying the operator U_Φ to the initial state $|\psi_0\rangle$ results in a new state $|\psi_f\rangle$, which shifts the

amplitude towards $|1\rangle$, thereby increasing the likelihood of selecting the experience.

For the entire memory buffer consisting of D experiences, the overall quantum state is the tensor product of all the individual qubit states is given as [34]:

$$\left|\psi^{\text{total}}\right\rangle = \left|\psi^{(1)}\right\rangle \otimes \left|\psi^{(2)}\right\rangle \otimes \cdots \otimes \left|\psi^{(D)}\right\rangle. \tag{32}$$

G. Quantum operations on experiences

To handle quantum representations of experiences, three key operations are employed: preparation, depreciation, and quantum observation. The preparation operation adjusts probability amplitudes based on updated TD-errors, steering quantum systems toward target states. Depreciation adapts the significance of experiences based on visitation frequency. Experience selection uses quantum observation to construct training minibatches. Grover iteration is adopted in both preparation and depreciation to amplify target state probabilities without affecting others. Unlike traditional methods operating on the full buffer, Grover's iteration is applied individually to each experience, enabling adaptive amplitude tuning and preventing the loss of crucial experience distinctions during replay.

1) Preparation operation: To enhance the efficiency of experience replay within the DRL-QER framework, it is essential to first differentiate the relevance of each experience. As a single quantum rotation alters the probability amplitude of a qubit, we define a fundamental rotation operator as in [34]:

$$U_\sigma = e^{-i\sigma Y} = \begin{bmatrix} \cos(\sigma) & -\sin(\sigma) \\ \sin(\sigma) & \cos(\sigma) \end{bmatrix} \tag{33}$$

Here, $\sigma \in \mathbb{R}$ denotes a small rotation angle. Using the exponential approximation, a sequence of basic unitary rotations $U_\Sigma = (U_\sigma)^m$ can be used to simulate a larger cumulative rotation. Moreover, since $e^{-i\sigma Y} e^{-i(-\sigma)Y} = I$, inverse rotations are conveniently represented as U_σ^{-1} or $U_\sigma^\dagger$. Therefore, rotations in either direction (clockwise or counterclockwise) can be realized by repeating the base operator.

The preparation operation for a single experience (specifically, the k^{th} transition in the buffer) involves applying multiple basic quantum rotations in the counterclockwise direction. These operations are designed to amplify the *selecting* amplitude of valuable experiences while simultaneously suppressing the *discarding* amplitude of less informative ones. Hence, the overall state evolution of this quantum system can be formulated as:

$$U_\Sigma^{(k)} = (U_\sigma)^{m_k}, \quad \left|\psi_f^{(k)}\right\rangle = U_\Sigma^{(k)} |\psi_0\rangle \tag{34}$$

where m_k indicates how many rotations are applied to the k^{th} qubit. Since TD-error δ_t reflects the importance of experience e_t, we relate m_k to this error. The priority score for the k^{th} experience is computed as $P_k = |\delta_t| + \epsilon$, where $P_{\max}$ denotes the maximum priority among all experiences. To translate priority into a rotation angle, we define a linear mapping such that the highest priority $P_{\max}$ corresponds to a maximum rotation angle $\Sigma_{\max}$, and the angle for P_k becomes $\Sigma_k = \Sigma_{\max} \times (P_k/P_{\max})$ [34]. Given that the Grover's iteration works by applying unitary operations until the system approaches a desired quantum state, we split $\Sigma_{\max}$ into μ parts and assign each part a rotation size σ. The initial state is rotated by angle ι, and thus the effective rotation angle m_k becomes [34]:

$$m_k = \text{Floor}\left(\mu \times P_k/P_{\max} - \iota/\sigma\right) \tag{35}$$

where $\text{Floor}(x)$ yields the greatest integer less than or equal to x, and $\mu, \iota \in \mathbb{R}$ are hyperparameters. The sign of m_k determines the direction of rotation with respect to the uniform superposition state angle ι. A positive m_k implies counterclockwise rotation (favoring $|1\rangle$), while a negative m_k leads to clockwise rotation (toward $|0\rangle$). The rotation increment σ in (34) is crucial and must be chosen carefully. While a fixed value offers convenience, dynamically adapting σ to the training process—based on TD-errors, experience frequency, or training progression—is often more effective. In this work, σ is modeled as a function of the training episode (TE) [34],

$$\sigma = \frac{\zeta_1}{1 + e^{\frac{\zeta_2}{\text{TE}}}} \tag{36}$$

with $\zeta_1, \zeta_2 \in \mathbb{R}$ as hyperparameters. By repeating this preparation strategy for all experiences in the buffer, each transition attains its corresponding quantum state. Valuable experiences will gradually evolve toward $|1\rangle$ via counterclockwise rotations, whereas less important ones will shift closer to $|0\rangle$ through clockwise adjustments.

2) Depreciation operation:: Following preparation, the probability of selecting experiences aligns with their TD-errors. However, frequently replayed transitions may lead to overtraining [38], aggravated by limited buffer size. In RL, this reflects the exploration–exploitation tradeoff [39], where sample diversity helps prevent convergence to suboptimal solutions. To address this, a depreciation operation iteratively adjusts the probability amplitudes of selected experiences, accounting for changes in their TD-errors and novelty. Another unitary operator used for experience depreciation is defined as:

$$U_{\boldsymbol{\omega}} = \begin{bmatrix} \cos(\boldsymbol{\omega}) & -\sin(\boldsymbol{\omega}) \\ \sin(\boldsymbol{\omega}) & \cos(\boldsymbol{\omega}) \end{bmatrix}, \tag{37}$$

where $\boldsymbol{\omega} \in \mathbb{R}$. This rotation is applied during Grover iterations to selected experiences, updating their quantum states via:

$$\left|\psi_f^{(k)}\right\rangle \leftarrow U_{\boldsymbol{\omega}} \left|\psi_f^{(k)}\right\rangle. \tag{38}$$

The angle $\boldsymbol{\omega}$ has to be adapted according to the particular scenerio. In experience replay, where older experiences are periodically replaced once the buffer is full, maintaining balanced replay frequency is critical. A large $\text{RT}_{\max}$ (maximum number of replays across experiences) indicates imbalance. To mitigate this, a smaller $\boldsymbol{\omega}$ slows depreciation, preserving

lower-priority experiences, while a larger value accelerates their decline. Moreover, the depreciation factor $\boldsymbol{\omega}$ should be dynamically adjusted based on the training episode (TE). In the initial stages of training, the significance of individual experiences is unclear. Over time, certain experiences consistently exhibit high TD-errors, regardless of how often they are used for learning. Thus, it is beneficial to emphasize the acceptance probabilities of frequently replayed experiences early on, and gradually reduce their influence to prevent overfitting in later stages. This adaptive control is achieved by increasing $\boldsymbol{\omega}$ with TE. Thus, we formulate ω, the depreciation factor as,

$$\boldsymbol{\omega} = \frac{\tau_1}{\mathrm{RT}_{\max}\left(1 + e^{\tau_2/\mathrm{TE}}\right)}, \tag{39}$$

where $\tau_1, \tau_2 \in \mathbb{R}$ are hyperparameters.

3) Experience selection by quantum observation: Experiences are taken from the buffer and fed into the network to enhance the learning process. The selection probabilities are calculated using quantum observation and the quantum measurement concept. The probability of selection for the kth qubit in state $|\psi_f^{(k)}\rangle$ corresponds to the likelihood of measuring $|1\rangle$, given by $\left|\langle 1|\,\psi_f^{(k)}\rangle\right|^2$. To ensure a valid probability distribution over all stored transitions, these individual probabilities are normalized across all transitions in the buffer. Thus, the final selection probability for the kth transition is defined as:

$$b_k = \frac{\left|\langle 1|\psi_f^{(k)}\rangle\right|^2}{\sum_i \left|\langle 1|\psi_f^{(i)}\rangle\right|^2}. \tag{40}$$

Each transition in the buffer is associated with a fixed selection probability. Sampling is conducted repeatedly based on these fixed probabilities, enabling prioritized experience replay within the quantum framework. *Remark 1:* Sampling is performed with replacement; that is, once a transition is selected, it remains in the buffer but is reinitialized to the uniform state. This reflects the quantum measurement principle, where an observation collapses the quantum state. Accordingly, subsequent quantum operations (e.g., preparation and depreciation) begin from the uniform superposition rather than the prior quantum state [34].

After formulating the necessary mathematical formulations and quantum-inspired operations, we now systematically consolidate the procedure into a coherent algorithmic workflow. The overall process is captured in **Algorithm 2**, which integrates the LSTM-Heuristic-DDQN-QiER framework. Each training episode (lines 1–2) begins by initializing the environment. For every time slot t (line 3), the eigenvector $\phi(s_t)$ representing the current state is derived (line 4). We use heuristic-ϵ-greedy selection policy for cache placement action a_t (line 5) using eq.(27). After executing a_t, the next state s_{t+1} and the immediate reward r_t are observed (lines 6-7). A new qubit is initialized (line 8), and using maximum TD-error $\delta_{\max}$, Grover iterations are performed to evolve the quantum state $|\psi_f^{(k)}\rangle$ (lines 9-10), which is stored along with the experience tuple in the experience replay (ER) buffer (line 11). When the mini-batch is full ($\tilde{k}\ \%\ mini == 0$, line 12), experiences are sampled probabilistically based on quantum amplitudes (line 13). For each mini-batch (lines 16–21), TD-errors, priorities and replaying counts are updated (lines 18-20), quantum operations (Preparation and Depreciation) are performed (line 21) and δ_{max} and RT_{max} are updated (line 22). The Q-network is updated via MSE loss (line 23), target network is synchronized (line 24), and used qubits are reset (line 25). This iterative process continues, adapting both caching decisions and learning priorities.

4) Time complexity analysis: In the following, we analyze the computational complexity of the proposed Heuristic-DDQN-QiER algorithm. In each episode, the system has U UAVs and with a maximum cache capacity of F' out of total F contents. The Q-network and target-network has two hidden layers with (L_1, L_2) neurons. At each step t, the agent takes an action in $\mathcal{O}\left(U(F\log F + F') + U(F + F')\right)$ (line 5) and the reward calculation has a complexity of $\mathcal{O}(U \cdot F)$ (line 6). The initialization of the quantum state and parameter assignment for m_k (lines 8–9) are constant-time operations, except for the preparation operation, which depends on m_k. Also, to insert the transition in a buffer is a constant time operation. The experience replay step includes computing replay probabilities, incurring a cost of $\mathcal{O}(k\log k)$, where $k \leq D$, with D being the buffer size (lines 13 and 16). Sampling a transition and computing the TD-error from the model prediction requires a forward pass through the neural network, with complexity $\mathcal{O}(U \cdot F)$ (line 18). The Grover iteration over the selected qubit involves $(m_d + cn_d)$ operations depending on the selected priority and replay count (line 21). Hence, the time complexity of code from (line 12-21) is $\mathcal{O}(\frac{D}{mini}(D + mini(DlogD + U \cdot F + m_d + cn_d)))$ (let denote by ϕ). The neural network weight update using backpropagation involves a complexity of $\mathcal{O}(U \cdot F \cdot L_1 + L_1 \cdot L_2 + L_2 \cdot U \cdot F)$, accounting for input, hidden, and output layer edge weight updates respectively (line 23). All further remaining operations are of constant time. Assuming each episode consists of T steps and the total number of episodes is J, the overall time complexity of the Heuristic-DDQN-QiER algorithm is $\mathcal{O}\left(JT\left(U(F\log F + F') + U(F + F') + U \cdot F + m_k + \phi\right)\right)$. Hence, the overall time complexity fits in bound of $\mathcal{O}(JT(UF\log F + \frac{D^2\log D}{mini}))$.

V. Performance analysis

The proposed Heuristic-DDQN-QiER framework encodes classical experiences into quantum-inspired representations and applies quantum-like operations. While grounded in quantum principles, it is entirely simulated on classical hardware [34], requiring no quantum device. In the simulation of **Algorithm 2**, a predefined value of the maximum TD-error, $\delta_{\max}$, is initially set. Throughout training, $\delta_{\max}$ is

Algorithm 2: Heuristic-DDQN-QiER Algorithm

Initialize: θ_e, θ_t, N, T, γ, ϵ, update frequency M, experience buffer size D, Δ, mini-batch size $mini$, σ, ω, $\delta_{\max}$, $cn = [cn_1, cn_2, \ldots, cn_D] = \mathbf{0}$, $k = 1$, $\tilde{k}$

Input : Content popularity matrix $\hat{\boldsymbol{Q}}(\boldsymbol{t})$ using Algo. (1)

for *episode* $= 1$ *to* J **do**
- Initialize environment and get the initial state s_t
- **for** *step* $= 1$ *to* T **do**
  - Acquire eigenvector $\phi(s_t)$ representing current state information;
  - Use ϵ-greedy to select cache placement action a_t using (27)
  - Take action a_t, and watch the next state at s_{t+1}.
  - Calculate reward $r_t(s_t, a_t)$ using (26)
  - Set the uniform state for the kth qubit upon initialization $|\psi_0\rangle$;
  - Determine m_k by setting $P_k = |\delta_{\max}|$. Eq. (35);
  - Execute the preparation operation on the kth qubit using Grover iteration, obtaining its final state: $|\psi_f^{(k)}\rangle = (U_\sigma)^{m_k} |\psi_0\rangle$ by Eq. (34)
  - Store transition $(s_t, s_{t+1}, a_t, r_t, done)$ along with its quantum representation $|\psi_f^{(k)}\rangle$ in ER;
  - **if** $\tilde{k}$ % $mini == 0$ **then**
    - Calculate the probability of sampling each experience by observing its quantum state, $[b_1, b_2, \ldots, b_k]$ using Eq. (40);
    - Update the preparation factor σ and the depreciation factor ω;
    - **for** $i = 1$ *to* $mini$ **do**
      - Sample a transition with index $d \in \{1, \ldots, k\}$ based on $[b_1, \ldots, b_k]$;
      - The d^{th} qubit should be reset to $|\psi_0\rangle$;
      - Compute TD-error: δ_i as in [34]
      - Obtain priority $P_d = |\delta_i|$ and determine m_d using Eq. (35);
      - Update the replayed time $cn_d += 1$;
      - Perform Grover iter. including preparation and depreciation operation : $|\psi_f^{(d)}\rangle = (U_\omega)^{cn_d}(U_\sigma)^{m_d} |\psi_0\rangle$
      - Update $\delta_{\max} = \max(\delta_{\max}, |\delta_i|)$ and update $RT_{\max} = \max(cn_1, \ldots, cn_D)$;
    - Compute loss using MSE Eq. , update θ_e using backpropagation, reset Δ;
    - Copy weights into target network: $\theta_t \leftarrow \theta_e$;
    - Remove the k^{th} qubit from buffer and reset $cn_k = 0$;
  - Set $k \leftarrow (k + 1)$ % D;
  - Set $\tilde{k} \leftarrow (\tilde{k} + 1)$ % $mini$;
  - Set $s_t = s_{t+1}$;

Output : Optimal cache placement vector $\boldsymbol{\Gamma(t)}$

dynamically updated whenever a larger TD-error is observed, ensuring that newly generated transitions receive the highest priority.

TABLE III: Hyper-Parameter Configurations for the Process of Learning

Parameters	Values
Capacity of QiER buffer D	10000
Initial ϵ-greedy factor ϵ	1
Learning rate α_{lr}	0.001
Maximum training episodes $t_{e_{max}}$	5000
Size of mini-batch $mini$	64
Annealing speed dec_ϵ	0.995/episode
Discount factor γ	0.99
Step threshold N_{max}	10
Preparation sub-factor ζ_1 [34]	0.8
Preparation sub-factor ζ_2	2 x 1.0
Depreciation sub-factor τ_1 [34]	0.5
Depreciation sub-factor τ_2	1.0
Parameter μ of m	12
Parameter ι of m [34]	$\pi/4$

A. Dataset Description and Pre-processing

We evaluate our proposed algorithm using the MovieLens 1M dataset [40], which comprises 1,000,209 ratings from 6,040 users on 3,952 movies, recorded between 2000 and 2003. Each entry includes a user ID, movie ID, rating (1–5), and timestamp, along with user demographics like gender, age, occupation, and zip code. In our setup, user ratings are interpreted as content requests to a wireless caching system, aligning with prior works (see [41], [42]. For simulation efficiency, the dataset was preprocessed by selecting the top 1,000 active users and 100 most-rated movies through stratified sampling to maintain rating distribution. Feature engineering involved one-hot encoding 18 movie genres, binary encoding gender (M=0, F=1), and normalizing zip codes via index mapping. The cleaned user, movie, and rating datasets were merged using inner joins to construct a user-item interaction matrix. To mitigate sparsity and facilitate supervised learning, the test set was created by forming a Cartesian product of selected users and movies, with missing interactions imputed as zeros. This preprocessing pipeline ensured data consistency, enriched feature representation, and allowed the recommender model to simulate realistic user behavior in cache-based content delivery networks.

B. Simulation setup

- **Wireless network setting :** We consider a hierarchical content delivery network architecture comprising $K = 1$ CSPs, $S = 2$ satellites, and $U = 8$ UAVs. The network serves $N = 1000$ GUs and supports a content library of $F = 100$ contents. Each UAV can cache up to $F' = 33$ contents, and the caching cost per content is fixed at $Y = 10$. Each content has a size of $Z = 100 \times 1024$ bits. The transmission bandwidth between the CSP and satellite, satellite and UAV, and UAV-to-UAV is set to

$B = 10$ MHz. The UAV transmission load is limited to $L_u = 11$ contents for each UAV. The transmission power is set to 50 dBm for both the CSP and satellite links, with corresponding path-loss exponents $\alpha_1 = \alpha_2 = 2$, and fading factors $g_{k,s} = 0.64$ and $g_{s,u} = 0.68783$, respectively. The distances between the CSP and satellite, and satellite and UAV, are set to 780 km. The channel noise power is calculated using a thermal noise model with a noise spectral density of -174 dBm/Hz. The noise powers for CSP-satellite and satellite-UAV links are σ_k and σ_s, respectively, computed accordingly. The cost per bit for content transmission is $P_k = 0.05$, and a constant weight multiplier for energy cost is set to 10^{-8}, to normalise cost and make it dimenionless.

- **LSTM Setting :** We employ a neural network comprising one input layer, three hidden LSTM layers with 24, 24, and 12 units respectively, and a fully connected output layer with a linear activation function. The network is trained using a learning rate of 0.0005 and a batch size of 32.
- **DDQN agent setting :** In DDQN utilizes a primary Q-network and a target Q-network, both having identical architectures. Each network comprises an input layer corresponding to the flattened system state, followed by two fully connected hidden layers with 64 neurons each and ReLU activation functions. The dueling architecture separates the estimation of the state value and the advantage function: a single neuron outputs the state value, and an output layer of size equal to the action space computes the advantage values. These are combined to produce Q-values using a mean-subtracted advantage aggregation. The networks are optimized utilizing a fixed learning rate of the Adam optimizer 0.001, and training is stabilized using a soft update mechanism for the target network.
- **Quantum replay buffer setting :** We record the experiences in a tuple $(s_t, a_t, r_t, s_{t+1}, \left|\psi_f^{(k)}\right\rangle)$ using a quantum-inspired experience replay. The batch size $mini$ and the replay buffer size D are set to 64 and 10,000, respectively. The value of hyperparameter used in preparation operation, depreciation operation and many more discussed in section IV-E is shown in table (III).

C. Benchmark schemes

To show the superiority of our model, we compare our proposed model with following benchmark schemes:

- *H-DDQN-PER:* This model uses a prioritized experience replay (PER) buffer, which means experiences (or transitions) with higher learning potential—often based on the magnitude of the TD (temporal-difference) error—are sampled more frequently during training.
- *H-DDQN-LRU:* This version employs an Least Recently Used (LRU) strategy for managing the experience replay buffer. Older and less recently used experiences are removed to make room for new ones. It ensures that recent and potentially more relevant transitions are retained for training.
- *H-DDQN-FIFO:* The experience replay buffer works like a queue. The oldest experiences are discarded first to accommodate new ones. It does not prioritize experiences based on their usefulness but maintains a simple chronological order.

D. Comparison and result analysis

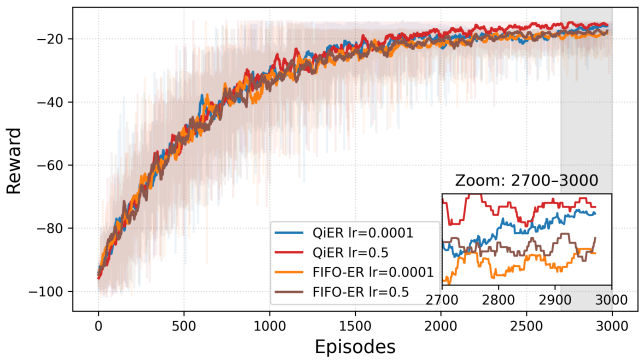

Fig. 5: Rewards vs. episodes

Fig. 5 presents the reward across training episodes for various learning rates and model variants in the context of cache placement optimization at UAVs using binary decision variables. The proposed Heuristic-DDQN-QiER model with a learning rate of 0.5 demonstrates superior learning stability and maximum rewards, effectively optimizing the binary cache placement vector at each UAV. Also, H-DDQN-QiER variants with lower learning rate 0.0001 perform equally well. However, the H-DDQN-FIFO-ER baseline with learning rates 0.0001 and 0.5 exhibit similar reward growth in the starting episodes but end up stabilising at lesser reward in the terminal phase as it can be seen in the zoomed part. This confirms the advantage of the QiER mechanism in enabling efficient and intelligent cache decisions in UAV-assisted networks over FIFO mechanism for Q-network learning.

As shown in Fig. 6, the proposed H-DDQN-QiER mechanism consistently achieves the lowest average CSP cost across varying numbers of contents, outperforming all other variants. The trend supports the fact that when no. of contents are increased, the demand for those contents has come into play, but UAV had limited cache capacities leading to frequent cache misses and increasing the CSP cost. Despite this, QiER maintains a substantial cost advantage, demonstrating reductions of $\approx$ 52.3% , 52.2% and 35% compared to PER, LRU and FIFO- ER, respectively. These results validate the efficiency and robustness of the QiER-based Q-network training approach in reducing CSP costs for UAV-enabled content caching frameworks.

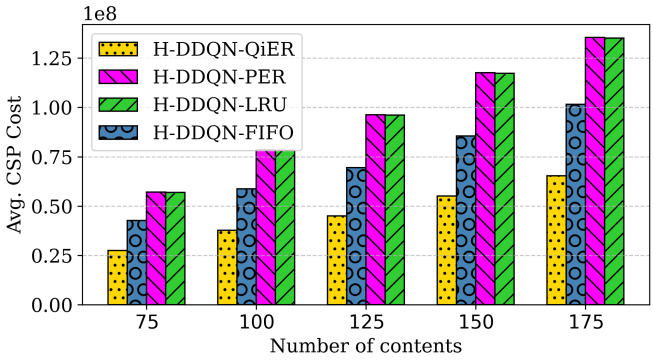

Fig. 6: Average CSP cost vs no. of contents

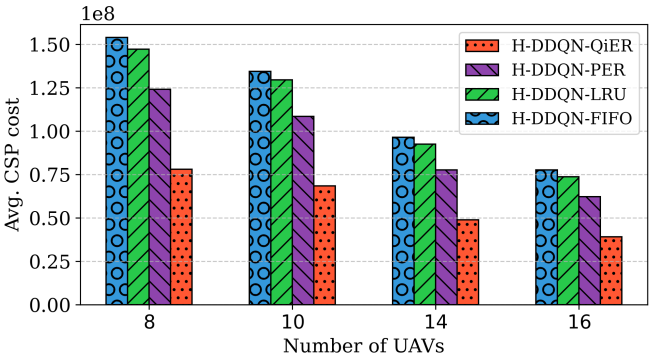

Fig. 8: Avg. CSP cost vs No. of UAVs

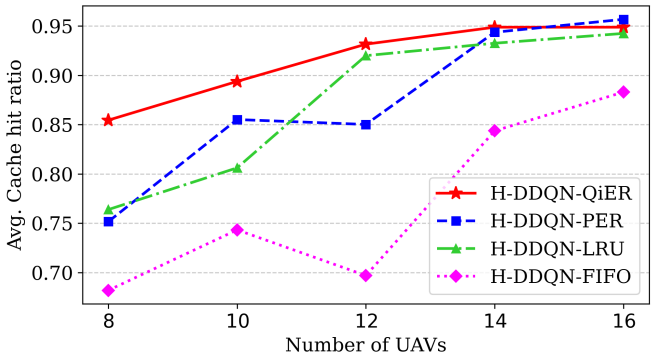

Fig. 7: Avg cache hit ratio vs. UAVs

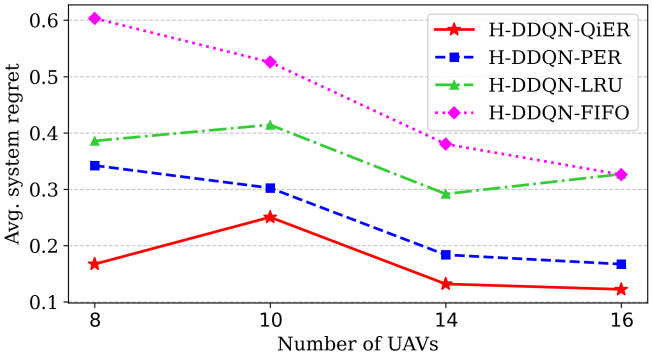

Fig. 9: Avg. System regret vs. No. of UAVs

As shown in Fig. 7, H-DDQN-QiER achieves the highest average cache hit ratio across all UAV counts, consistently outperforming all other variants. The trend is seen in the way that, as the number of UAVs increase (↑), the cache hit ratio increases (↑), because more UAVs in a cluster increase the chance of cluster-assisted content delivery, hence reducing CSP cost for retrieving the same content from backhaul links from CSP via satellite. Specifically, H-DDQN-QiER achieves a 5.5% and 5.2% higher cache hit ratio than H-DDQN-PER and H-DDQN-LRU, respectively, and a significant 19.8% improvement over H-DDQN-FIFO. This is evident from Fig. 8. In addition, the system regret, which is essentially the cache miss, should decrease (↓). This is also validated by Fig. 9 that increasing UAVs reduces system regret.

Fig. 10 shows that the average CSP cost decreases as the transmission capacity increases. This is because transmission capacity determines the maximum number of files a UAV can retrieve from the CSP upon a cache miss. A higher transmission capacity says that more no. of contents can be retrieved from CSP to accomodate in cache vector of a UAV, reducing the number of cache misses, which previously required costly CSP access. Among the models, H-DDQN-QiER consistently achieves lower CSP cost, especially at higher value, due to its efficient cache replacement strategy that reduces dependency on CSP content retrieval.

Fig 11 shows that the average cache hit ratio varies slightly with the number of users, but H-DDQN-QiER consistently achieves the highest hit ratio when compared to other three variants. The variation is due to the fact that when no. of users increases, the request for diverse content also increases, but since UAV has a limited cache capacity and can cache some contents only, leading to cache miss for remaining uncached contents.

Fig. 12 illusatrated the comparison of cache hit ratio for different cache capacity of UAVs with different variant replay

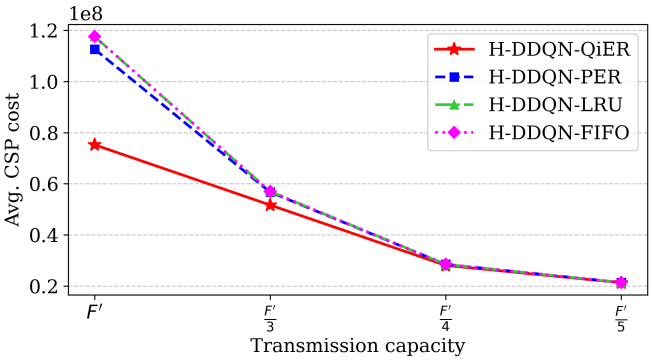

Fig. 10: Avg. CSP cost vs Transmission Capacity

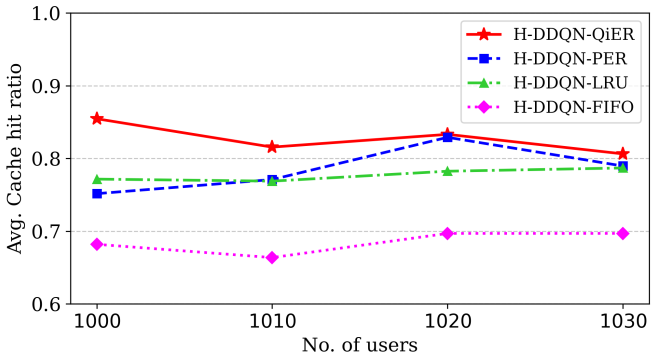

Fig. 11: Cache Hit Ratio vs. Users

buffer of H-DDQN. It is clearly evident that the proposed H-DDQN-QiER outperforms all other variants. The trend is supported by the fact that, when cache capacity of a UAV increases ↑ , it can accomodate more no of. contents in its cache vector, leading to less system regret (i.e. higher cache hit or equivalently lesser cache miss) and the lowers the CSP retrieval cost. This explanation is also supported by fig. 12, fig. 14 and fig. 13.

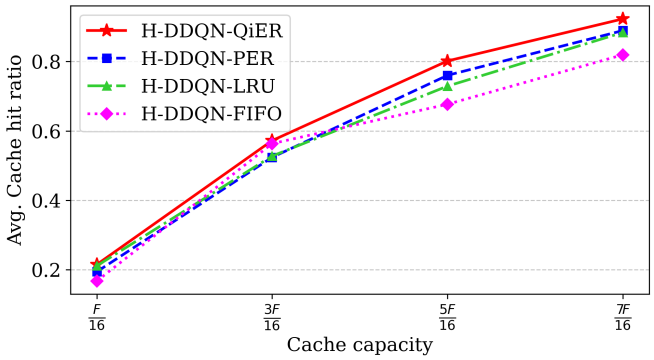

Fig. 12: Avg. Cache hit ratio vs. Cache capacity

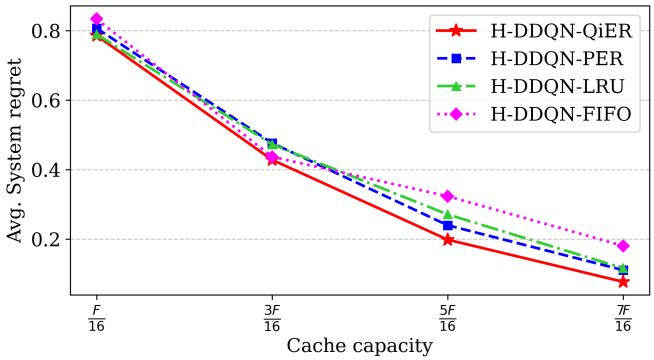

Fig. 13: Avg. System regret vs. Cache capacity

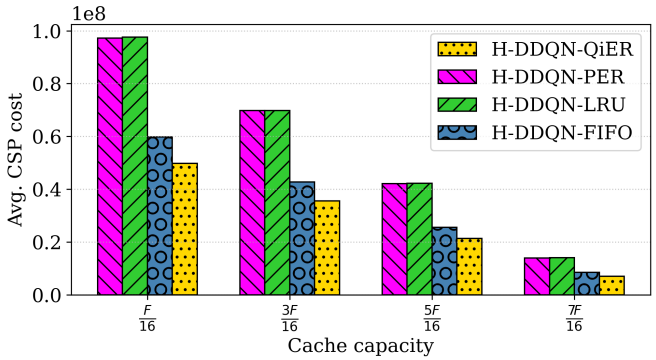

Fig. 14: Avg. Cost for content retrieval from CSP via satellite vs. Cache capacity

VI. Conclusion and Upcoming Projects

In this work, we considered a UAV-satellite-assisted edge caching scenario and proposed a proactive data caching approach using a heuristic-based modified Deep Double Q-Network with quantum inspired experience replay (Heuristic-DDQN-QiER). We jointly minimize the system regret and the CSP cost to maximize the UAV cache ratio. Numerous experiments are built using real-world datasets, such as Movielens, and the outcomes confirm the effectiveness of the and the suggested method's advantages over a number of other methods.

For upcoming projects, we plan to prolong the current model by incorporating dynamic user mobility and adaptive UAV association to better reflect real-world scenarios. Furthermore, integrating moving satellites and enabling inter-satellite communication would support cooperative caching and broader content distribution. The application of SDN could provide centralized control and optimization over the caching process.

References

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," n.d. Accessed: 2025-04-04.

[2] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[3] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *2010 Proceedings IEEE INFOCOM*, pp. 1–9, IEEE, 2010.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 1, pp. 126–134, IEEE, 1999.

[5] X. Xu, Y. Zeng, Y. L. Guan, and R. Zhang, "Overcoming endurance issue: Uav-enabled communications with proactive caching," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1231–1244, 2018.

[6] S. Müller, O. Atan, M. Van Der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2016.

[7] X. Bi and L. Zhao, "Collaborative caching strategy for rl-based content downloading algorithm in clustered vehicular networks," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9585–9596, 2023.

[8] S. K. Mishra, C. Goyal, C. Agrawal, and A. Pratap, "Minimizing costs for content service providers in 6g space-air-ground integrated networks," in *2024 IEEE Space, Aerospace and Defence Conference (SPACE)*, pp. 479–483, 2024.

[9] S. Anokye, D. Ayepah-Mensah, A. M. Seid, G. O. Boateng, and G. Sun, "Deep reinforcement learning-based mobility-aware uav content caching and placement in mobile edge networks," *IEEE Systems Journal*, vol. 16, no. 1, pp. 275–286, 2021.

[10] S. Yan, M. Jiao, Y. Zhou, M. Peng, and M. Daneshmand, "Machine-learning approach for user association and content placement in fog radio access networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9413–9425, 2020.

[11] Q. Wei, H. Ma, C. Chen, and D. Dong, "Deep reinforcement learning with quantum-inspired experience replay," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9326–9338, 2021.

[12] J. Huang, M. Zhang, J. Wan, Y. Chen, and N. Zhang, "Joint data caching and computation offloading in uav-assisted internet of vehicles via federated deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, 2024.

[13] D. Liu and C. Yang, "A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users," *IEEE Access*, vol. 7, pp. 83120–83136, 2019.

[14] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 1, pp. 126–134 vol.1, 1999.

[15] S. Chai and V. K. N. Lau, "Online trajectory and radio resource optimization of cache-enabled uav wireless networks with content and energy recharging," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1286–1299, 2020.

[16] M.-H. T. Nguyen, T. T. Bui, L. D. Nguyen, E. Garcia-Palacios, H.-J. Zepernick, H. Shin, and T. Q. Duong, "Real-time optimized clustering and caching for 6g satellite-uav-terrestrial networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 3009–3019, 2023.

[17] S. Yang, X. Xia, and B. Lorenzo, "Caching and computation offloading for intelligent transportation systems enabled by satellite-terrestrial networks," in *2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 722–729, IEEE, 2023.

[18] D. Wang, Q. Liu, J. Tian, Y. Zhi, J. Qiao, and J. Bian, "Deep reinforcement learning for caching in d2d-enabled uav-relaying networks," in *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 635–640, IEEE, 2021.

[19] C. Wu, S. Shi, S. Gu, L. Zhang, and X. Gu, "Deep reinforcement learning-based content placement and trajectory design in urban cache-enabled uav networks," *Wireless Communications and Mobile Computing*, vol. 2020, no. 1, p. 8842694, 2020.

[20] C. et al., "Trajectory design and link selection in uav-assisted hybrid satellite-terrestrial network," *IEEE Comm. Letters*, vol. 26, no. 7, pp. 1643–1647, 2022.

[21] M.-H. T. Nguyen, T. T. Bui, L. D. Nguyen, E. Garcia-Palacios, H.-J. Zepernick, H. Shin, and T. Q. Duong, "Real-time optimized clustering and caching for 6g satellite-uav-terrestrial networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 3009–3019, 2024.

[22] X. Dai et al., "Priority-aware task offloading and resource allocation in satellite and hap assisted edge-cloud collaborative networks," in *2023 15th ICCSN*, pp. 166–171, 2023.

[23] Q. Chen, W. Wang, F. R. Yu, M. Tao, and Z. Zhang, "Content caching oriented popularity prediction: A weighted clustering approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 623–636, 2021.

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

[25] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International conference on machine learning*, pp. 1764–1772, PMLR, 2014.

[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[27] O. Vinyals, Ł. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," *Advances in neural information processing systems*, vol. 28, 2015.

[28] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, PMLR, 2015.

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

[30] M. Yang, D. Gao, W. Zhang, D. Yang, D. Niyato, H. Zhang, and V. C. M. Leung, "Deep reinforcement learning-based joint caching and routing in ai-driven networks," *IEEE Transactions on Mobile Computing*, vol. 24, no. 3, pp. 1322–1337, 2025.

[31] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2197–2211, 2020.

[32] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 connectionist models summer school*, pp. 255–263, Psychology Press, 2014.

[33] Z. Guo, F. Tang, X. Chen, L. Luo, and M. Zhao, "Deep-reinforcement-learning-based content caching in satellite–terrestrial assisted airborne communications," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 22779–22789, 2024.

[34] Q. Wei, H. Ma, C. Chen, and D. Dong, "Deep reinforcement learning with quantum-inspired experience replay," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9326–9338, 2022.

[35] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.

[36] D. Dong, C. Chen, H. Li, and T.-J. Tarn, "Quantum reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1207–1220, 2008.

[37] L. K. Grover, "Quantum computers can search arbitrarily large databases by a single query," *Physical review letters*, vol. 79, no. 23, p. 4709, 1997.

[38] T. De Bruin, J. Kober, K. Tuyls, and R. Babuška, "The importance of experience replay database composition in deep reinforcement learning," in *Deep reinforcement learning workshop, NIPS*, 2015.

[39] S. Ishii, W. Yoshida, and J. Yoshimoto, "Control of exploitation–exploration meta-parameter in reinforcement learning," *Neural networks*, vol. 15, no. 4-6, pp. 665–687, 2002.

[40] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[41] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, 2016.

[42] 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 Network*, vol. 32, no. 6, pp. 50–57, 2018.