Combining Lyapunov Optimization and Deep Reinforcement Learning for D2D Assisted Heterogeneous Collaborative Edge Caching

Ziyi Teng, *Student Member, IEEE*, Juan Fang, *Member, IEEE*, and Yaqi Liu, *Student Member, IEEE*

***Abstract*—The problem of shared node selection and cache placement in wireless networks is challenging due to the difficulty of finding low-complexity optimal solutions. This paper proposes a new approach combining Lyapunov optimization and reinforcement learning (LoRL) to address content sharing in heterogeneous mobile edge computing (MEC) networks with base station (BS) and device-to-device (D2D) communication. Device in this network can choose to establish D2D links with neighboring devices for content sharing or send requests directly to the base station for content. Content access and energy consumption of shared nodes are modeled as a queuing system. The goal is to assign content sharing nodes to stabilize all queues while maximizing D2D sharing gain and minimizing latency, even in the presence of unknown network state distribution and user sharing costs. The proposed approach enables edge device to independently select associated nodes and make caching decisions, thereby minimizing time-averaged network costs and stabilizing the queuing system. Experimental results show that the proposed algorithm converges to the optimal policy and outperforms other policies in terms of total queue backlog trade-off and network cost.**

***Index Terms*—Edge cache, content sharing, device-to-device communication, deep reinforcement learning, Lyapunov optimization.**

I. Introduction

THE INCREASING number of smart devices joining wireless networks has led to a surge in wireless multimedia traffic [1]. However, a significant portion of this traffic consists of repeated requests for popular content such as news articles and TV shows. To address this issue, mobile edge computing (MEC) technology has emerged as a promising solution, allowing content retrieval from edge storage nodes like base stations or clouds. However, this approach often results in redundant data transmission within a short timeframe and is constrained by the cache capacity of edge devices.

To mitigate the limitations of storage capacity and duplication in edge devices, collaborative caching has been recognized as an effective solution [2]. The progression of integrated circuits has led to the integration of storage and computing abilities into edge devices. Consequently, content sharing via device-to-device (D2D) communication has become feasible. Furthermore, the integration of D2D communication with MEC caching can yield additional benefits such as improved spatial frequency reuse and boosted cellular network throughput. These benefits can lead to reductions in transmission and backhaul loads while augmenting the service probability of tasks [3], [4].

Extensive research has focused on optimizing user performance in Mobile Edge Computing (MEC) networks through collaborative caching and device-to-device (D2D) caching. Content cache placement strategies have been specifically studied [5], [6], [7], [8], [9], [10], [11]. However, there are still unresolved issues that require attention. Firstly, existing studies mainly concentrate on improving cache performance through effective placement strategies. While this approach proves beneficial, there is a limitation on the amount of content a device can transfer within a given time frame. When simultaneous content requests exceed the transmission capacity, it causes delays and increases retrieval latency. To tackle this challenge, researchers propose using a virtual queuing system to represent access requests [10], [11]. This allows for optimization of queue management to mitigate transmission delays. Secondly, in collaborative networks, forwarding content sharing node requests incurs energy consumption costs. Virtual queues effectively represent node energy consumption dynamics. Therefore, ensuring stability in the consumption queue becomes crucial for overall system stability.

The selection of shared content delivery nodes and cached content replacement in a D2D-assisted MEC network are key issues. When a user's local cache cannot fulfill a request, it is necessary to determine which cache node (e.g., neighboring user node, local/base station, neighboring base station) should handle the request and how to cache the content [3]. User requests can be routed to other users or accessible edge nodes, and dynamic queues represent the content access and energy consumption of all edge nodes. The objective is to select shared nodes for unsatisfied requests, ensuring the stability of the request and energy consumption queues. However, there are three main considerations when deciding on shared nodes and caching. Firstly, the data rate for content sharing via D2D links depends on user distance and channel conditions. User

Manuscript received 27 June 2023; revised 5 November 2023; accepted 28 January 2024. Date of publication 2 February 2024; date of current version 12 July 2024. This work is supported by Beijing Natural Science Foundation (4192007), and supported by the National Natural Science Foundation of China (61202076), along with other government sponsors. The associate editor coordinating the review of this article and approving it for publication was S. Hoteit. *(Corresponding author: Juan Fang.)*

The authors are with the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: tengziyi@emails.bjut.edu.cn; fangjuan@bjut.edu.cn; liuyq617@emails.bjut.edu.cn).

Digital Object Identifier 10.1109/TNSM.2024.3361796

preferences can be challenging to understand, but machine learning methods can be employed to learn and predict preferences for better decision-making. Secondly, content sharing incurs transmission costs, necessitating consideration of both request delay and network cache node energy consumption to ensure network stability. Lastly, the cache replacement policy should adapt to the evolving wireless network environment, facilitating optimal replacement decisions based on changing network conditions.

Note that when confronted with a large number of users, simultaneously selecting shared nodes and making caching decisions for each user in the system poses a significant challenge. However, this challenge can be addressed by employing a stochastic Lyapunov optimization method that does not rely on prior knowledge of the network state distribution. Nevertheless, in our case, the uncertainty surrounding the user's mobile location, network channel state, and user preference further complicates the problem. This increased complexity arises from the need to stabilize all the queues while also considering cache decision problems. Therefore, We introduce context-aware preference learning strategies and propose dynamic shared node selection and cache replacement methods that combine Lyapunov optimization and reinforcement learning (LoRL). Thus, the main contributions of this paper are as follows:

- Our approach integrates Lyapunov optimization theory and reinforcement learning to develop a novel method for shared node selection and cache replacement. By considering random fading channels and data arrival, our method enables intelligent decision-making for user delivery node selection and cache replacement. The primary objective is to minimize user request latency and energy consumption while ensuring the stability of user request and energy consumption queues.
- We analyze the factors influencing the establishment of D2D links between users, considering both physical conditions and user preference similarity. To improve the effectiveness of D2D links, we propose a user preference model based on **AutoEncoder**. This model captures and measures the similarity in user preferences, enabling us to optimize the formation of D2D links based on shared interests.
- We integrate Lyapunov and reinforcement learning optimization techniques to address network optimization problems. Our approach involves deriving an upper bound for the drift plus cost function using Lyapunov optimization, which guides the training of the reinforcement learning model. By doing so, we enable the model to maximize user service quality while ensuring the stability of the system.

II. Related Work

Existing research on edge cache optimization can be divided into: i) accuracy improvement of popularity prediction models; ii) joint optimization of edge caching and wireless resources; iii) collaborative caching.

A. Popularity Prediction

Due to the repetitive nature of content requests in the network, edge caching should cache content with high popularity. Cache placement policies based on popularity prediction have demonstrated caching effectiveness, and reactive caching [12], [13] or proactive caching [14], [15] by analysing past historical request information to obtain request patterns has been extensively investigated. Hassine et al. [16] used Auto-regressive and Moving Average (ARMA) models for centralised content popularity prediction. To overcome the sparse nature of user requests, Chen et al. [17] proposed a popularity prediction scheme based on weighted clustering and also described an explicit relationship between cache performance and popularity prediction accuracy. In addition, considering the private nature of user data, a federated learning approach is used for edge caching policy optimization [3], [18]. Due to the consumption caused by learning-based approaches, some online popularity prediction approaches that do not require a training phase have also been proposed [12], [19]. However, while the improvement of the accuracy of the popularity prediction model can improve the caching performance to a certain extent, the channel conditions as well as the network state in mobile edge networks can have an impact on the quality of service (QoS) of the users. Therefore, the relationship between cache performance and prediction accuracy is implicit, and the impact of popularity prediction errors on cache performance is difficult to estimate. The most popular (MP) algorithm with a priori popularity knowledge of the user request model is compared with the proposed algorithm in Qian et al. [21], and this is verified by the poor performance exhibited by the MP algorithm.

B. Joint Caching and Resource Optimization

The decentralization of cache capacity in MEC networks leads to strong coupling between cache strategies and wireless communication resource management. From the perspective of limited cache and wireless resources, study the caching problem. Existing research approaches to caching policies are classified as *optimization-based*, *reinforcement learning-based* and *deep learning-based*, and *game-theory-based*. **Optimization based** caching strategies are usually designed to maximize certain performance metrics within the constraints of network resources. For the optimization problem of complex joint wireless resources and caching, simple heuristic algorithms often require a long time and can only obtain sub-optimal solutions. Therefore, in existing research, a Lyapunov optimization method for online joint utility maximization and stability control framework has been proposed. This method decouples multi-stage stochastic optimization problems into continuous deterministic sub-problems for each stage, while providing theoretical guarantees for the long-term stability of the system [13], [20]. **Strategies based on reinforcement learning** and **deep learning** use observable user data or environmental states, such as user contextual information, channel gain or cache state, for online caching decisions and resource allocation. Wireless channels have a finite amount of data that can be transmitted per unit time, and proactive caching

strategies are investigated in order to maximise bandwidth utilisation [21], [22], [23], [24]. However, when the user request or environment state space is large, centralised Reinforcement Learning caching strategies are complex and difficult to handle, hence distributed reinforcement learning approaches are proposed [25]. **Game theoretic** caching strategies have been used for caching and computing resource allocation in MEC network environments, where service providers or users compete among themselves for limited computing and bandwidth resources to meet their own interests [26], [27].

C. Collaborative Caching

In MEC networks, collaborative caching is an effective approach to reduce network service load, improve service latency, and enhance spectrum usage efficiency by expanding cache capacity. Existing research divides collaborative caching into two types based on cache location: *Coordinated Multi-Point (COMP)* and *D2D* caching. COMP involves obtaining requested content from adjacent devices, base stations, or other caching devices. The optimization goal in this context is to jointly optimize content caching and delivery decisions, considering network constraints and aiming to minimize service latency or content retrieval costs [28], [29]. To address the joint optimization problem of user collaboration nodes and cache placement, a decoupling approach can be employed for dual-scale joint optimization [20], [30].

Decentralised cooperative sharing methods address the challenges of diverse network states in wireless networks. They aim to solve the node selection problem in centralised cooperative transmission effectively [18]. These methods decentralise decision-making, allowing nodes to independently select cooperative partners and make transmission decisions based on local information. In networks with caching capabilities on the user side, collaborative sharing can be performed between D2Ds by establishing D2D communication [31]. Furthermore, the uncertainty of user requests and movement patterns makes it challenging to establish D2D connections. Therefore, based on learning methods, the user's movement trajectory as well as user request patterns are predicted to enable dynamic delivery of content [8]. Further, user data information is private and has the property of not being willing to be shared, while an effective local caching policy requires knowledge of user preference information. Therefore, to maximise the benefits for users, D2D content sharing approaches with social awareness and incentives have been proposed in existing studies [6], [27]. Moreover, in addition to horizontal collaboration between network cache nodes, vertical inter-tier collaboration between cache nodes is also an important solution to achieve service demand by expanding cache capacity. Similarly, some D2D-assisted heterogeneous collaboration approaches have been proposed to maximise spectrum efficiency and reduce request latency [3].

D. Our Contribution

Based on the aforementioned categorization, similar to the works in [3], [8], and [12], this paper investigates heterogeneous collaborative caching strategies supporting D2D assistance. Similar to the contributions in [14] and [15], we predict user preference popularity by analyzing historical request information. Similar to the contributions in [23], [24], [25], [26], [27], and [28], we utilize reinforcement learning for dynamic cache decision optimization. However, what sets our work apart from these contributions is that we employ predicted user preferences for D2D shared node selection. Furthermore, we combine Lyapunov optimization with reinforcement learning for user-associated node selection, cache decision-making, and maintaining stability in the request latency queue and cost consumption within the requesting network nodes.

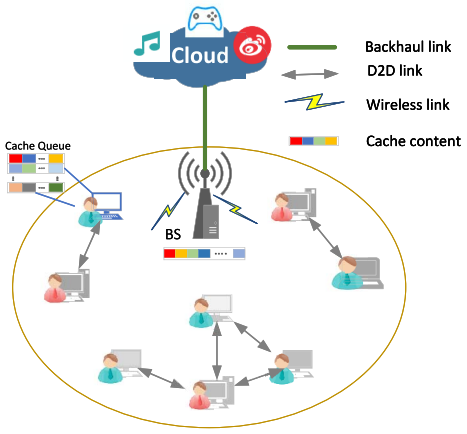

Fig. 1. D2D-assisted network architecture for heterogeneous collaborative edge caching.

III. System Model

A. Network Model

Consider a wireless D2D-assisted heterogeneous collaborative network architecture, as shown in Fig. 1. The network architecture includes three types of cache nodes, namely user equipment (UE), base stations (BS) and cloud servers. The cloud server connects to all BS via backhaul links to provide services to users, and the BS serve users via cellular links. In addition to the traditional BS-to-User use of the BS's wireless spectrum for content delivery, the considered network architecture allows for D2D links between users for content sharing. Given the abundant storage and computing resources of cloud servers, we assume that the cloud server has access to all the content that users may request within the storage area, denoted as $\mathcal{F} = \{1, 2, 3, \ldots, F\}$, where F represents the total number of contents [2], [3], [5], and its size is denoted as $(s_f^{1\times F})$. Each UE and BS has a restricted cache capacity to store content with high content popularity. The cache capacity of user device u is M_u, where $\forall u \in \mathcal{N} = \{1, 2, \ldots, N\}$ is the set of tags of users. To simplify the model, we consider the existence of one BS in the network, serving the UE. In particular, the BS have the limited cache capacity, denoted as M_B.

To meet generality, the capacity of the network's cache nodes is $M_u \leq M_B \leq F$. All cache nodes in the network architecture, except for cloud nodes, are represented as

$\mathcal{H} = \{\mathcal{N} \cup B\}$. In the network under consideration, operations are organized based on a time slot framework, represented as $\mathcal{T} = \{1, 2, 3, \ldots, t\}$. The time axis is divided into equal time intervals, referred to as slots, with a small duration denoted as Δt and $t \in \mathcal{T}$. Within a time slot t, all network parameters (e.g., user location, channel quality, content prevalence) remain constant. This time-slot-based organization enables the analysis and optimization of network performance within well-defined and consistent time intervals. Let us define the position of user u during time slot t as $l_{u,t}$, denoted by $l_{u,t} = \{x_{u,t}, y_{u,t}\}$. Here, $x_{u,t}$ and $y_{u,t}$ represent the coordinates of user u in the given time slot. It is essential to emphasize that within any time slot t, each user can make at most one request, and this request must be fulfilled prior to the commencement of the subsequent time slot.

B. D2D Sharing Mode

In the considered heterogeneous collaborative network, cache node content sharing has a restricted physical range, so it is assumed that the service range of the base station is R_B and the range of D2D is bounded by the radius R_u and satisfies $R_u \leq R_B$, i.e., $\pi R_u^2 \leq \pi R_B^2$, and cache nodes exceeding the service range cannot establish connections. In addition, user requests have the characteristics of being numerous and diverse. Therefore, in order to improve the effectiveness of D2D links, we will model the D2D connections from the physical domain and user similarity respectively in the following.

Physical domain: Due to physical limitations such as signal attenuation between D2D, only users within the user coverage can communicate [32]. Therefore, similar to [10], a graph $G_p = \{N, Y_p\}$ is introduced, where N represents the set of vertices of all users and $Y_p = \{(u, v) | e_{u,v}^p = 1, \forall u, v \in \mathcal{N}\}$ represents the edge set. When $e_{u,v}^p = 1$, it means that the distance between user equipment u and user equipment v is $L_{u,v}^t < R_u$, that is, within the communication range of the user, where $L_{u,v}^t = \sqrt{(x_{u,t} - x_{v,t})^2 + (y_{u,t} - y_{v,t})^2}$. Otherwise, $e_{u,v}^p$=0.

User similarity: Humans are herd animals and tend to have a herd mentality for content acquisition, and each person has a different request preference P_u. Therefore, the concept of cosine similarity is introduced to represent the relationship between users' preferences. The cosine similarity between user u and user v is

$$S_u^v = \frac{P_u \cdot P_v}{\|P_u\| \|P_v\|}. \tag{1}$$

A user with a high preference similarity indicates that the content stored in the cache is more similar, and therefore, the content requested by the user is more likely to be stored.

User sharing probability: Based on the physical map and user similarity obtained above, we get the connection probability $R_{u,v} = e_{u,v}^p \cdot S_v^u$ between users, and define $R_u = (R_{u,v})^{1 \times N}$. Finally, we normalize R_u by $\frac{R_{u,v}}{\sum_{u \in \mathcal{N}} R_{u,v}} \rightarrow R_u$.

In the next sections, user preferences are predicted using an stack AutoEncoder (SAE)-based algorithm, and then content delivery and latency models for D2D-assisted heterogeneous networks are investigated in Section III-D.

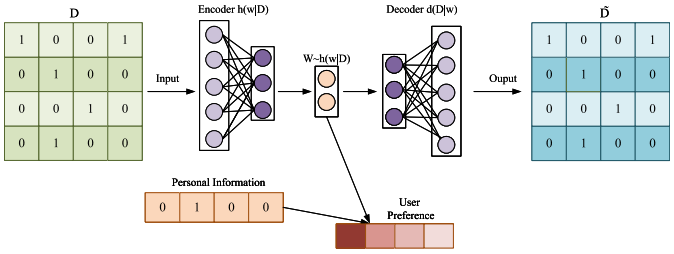

Fig. 2. A framework for learning user preferences based on contextual information.

C. User Preference Learning

In order to obtain the similarity between users, a hypothesis is made that, within a time frame, users send preference information to surrounding cache nodes. User preferences, i.e., the request popularity of a user, are uncertain and therefore obtaining numerical results for their popularity probability is very complex. Additionally, we assume that the preferences of users for the requested content follow an independent and identically distributed process. Unsupervised learning is a feasible approach to solve this problem. Therefore, in order to obtain the preference probability P_u of user u to further obtain the sharing probability $R_{u,v}$ between users, we propose a contextual information-based preference learning framework based on an unsupervised learning-hybrid filtering neural network model, as shown in Fig. 2.

The core of the proposed model is to minimise the discrepancy between the input and output, thus training a single hidden layer neural network to reconstruct the input data from the latent representations. The model consists of two main components: (i) an encoder that receives the input data and (ii) a decoder that outputs the results. The difference between the input and output is measured by the loss function F_L

$$F_L\left(D, \widetilde{D}\right) = \sum_{i=1}^{d} \left(\widetilde{D}_i - D_i\right), \tag{2}$$

where $\{D_1, D_2, \ldots\}$ is the input dataset, $D_{(i)} \in \mathbb{R}^d$ represents the dimension d of each element in the dataset, mainly including the user's contextual information, such as name, gender, age, movie, movie rating and etc. The input data is represented implicitly using the encoder through the activation function $h(x)$ for the input mapping as $h_{w,b}(D_i) \approx \widetilde{D}_i$, where $\{\widetilde{D}_1, \widetilde{D}_2, \ldots\}$ is the output of the corresponding encoder and the parameters w and b are the weight matrix and bias vector, respectively.

The training of user preferences means that the model is to be continuously updated and used to minimise the reconstruction error of the input data set. By training the SAE neural network, the hidden features encoding z of the training data are obtained and these features are used to calculate the user similarity. Since what content is needed in the future, i.e., the popularity of the user's request, may depend on the user's context. Therefore, by combining the hidden feature encoding z with the user's contextual information, the user's preferences, i.e., the user's own content request popularity, are obtained.

D. Content Transmission and Delay Model

In a heterogeneous collaborative network, a user's cache list in time slot t can be represented as $x_u = \{x_{u,f} \in \{0,1\}, u \in \mathcal{N}, f \in \mathcal{F}\}$. $x_{u,f} = 1$ means that content f is stored in the cache, otherwise, if $x_{u,f} = 0$, it means that the user does not store content f. When the user's requested content f cannot be satisfied locally, the user selects a node for content sharing from the network. We define $b_{u,t} = \{b^n_{u,t} \in \{0,1\} | u \in \mathcal{N}, n \in \mathcal{H}/u\}$ to denote the set of associated candidate nodes for user u. $b^n_{u,t} = 1$ indicates that user u selects cache node n to process user requests. Specifically, $b^0_{u,t} = 1$ indicates that users directly obtain content requests from the base station. Therefore, in D2D assisted heterogeneous networks, there are several methods for obtaining content.

1) The user's request is saved in the local cache list, i.e., the request is satisfied in the local cache. Therefore, the corresponding user request delay $d^L_{u,t}$ is 0.
2) If a user request is not saved in the local cache list, we can establish a D2D link to obtain the request from nearby users n. In line with the approach taken in many existing studies [3], [5], [6], we adopt orthogonal models to allocate non-overlapping radio resources for D2D transmission. This allocation scheme involves dividing the bandwidth of each node into equal sub-bands and assigning equal sub-bands to each node while ensuring no interference among them. Thus, the request delay $d^{D2D}_{u,t} = \frac{s_f}{r^n_{u,t}}$, where s_f denotes the size of the user request file and $r^n_{u,t}$ denotes the data transfer rate between user u and user n. $r^n_{u,t} = B^D log_2(1 + g^D_t h^D_t / L^n_{u,t})$, with parameters B^D, g^D_t, h^D_t denote the inter-user channel bandwidth, transmission power consumption and channel gain, respectively.
3) The user u can also send a request to the local base station for content retrieval. If the base station can fulfill the request, the delay for file f is given by $d^B_{u,t} = \frac{S_f}{r^B_{u,t}}$. The transmission rate is calculated using the formula $r^B_{u,t} = B^B \log_2(1 + \frac{g^B_t h^B_t}{L^B_{u,t}})$, where $L^B_{u,t}$ represents the path loss between user u and the base station at time slot t. Here, $r^B_{u,t}$ represents the transmission rate from user u to the base station at time slot t. It is determined by parameters such as the bandwidth B^B, transmission power consumption g^B_t, and channel gain h^B_t associated with the user-to-base station link.
4) Eventually, when content cannot be obtained through direct sharing or local caching, the base station forwards the request to the cloud server. The latency of fetching the content from the cloud server is the sum of the base station delay $d^B_{u,t}$ and the transmission time required to transfer the content from the cloud server. This can be expressed as $d^c_{u,t} = d^B_{u,t} + \frac{s_f}{r_c}$, where s_f is the size of the requested content and r_c is the constant transmission rate between the base station and the cloud server.

Therefore, the request latency for the user of the above fetch request method satisfies $d^L_{u,t} \leq d^{D2D}_{u,t} \leq d^B_{u,t} \leq d^c_{u,t}$. When the user makes the selection decision $b^n_{u,t}$, if the selected content sharing cache node cannot meet the request, it will obtain the request from the same or upper layer cache node by the above collaborative method.

IV. Queue Model and Caching Problem

A. Queuing Model

When user u initiates a request, if the request cannot be fulfilled locally, a content sharing node $b^n_{u,t}$ is selected. Based on the chosen delivery node $b^n_{u,t}$, the content access queue associated with node n is denoted as $Q_n(t)$. The dynamic backlog of all queues in the network is captured by the vector $Q(t) = \{Q_n(t) | n \in \mathcal{H}\}$. The arrival rate of $Q_n(t)$ represents the total size of content shared by node n as selected by users. Therefore, the arrival rate of queue is

$$A_n\left(b^n_{u,t}\right) = \sum_{u \in \mathcal{N}, n \in \{\mathcal{H}/u\}} b^n_{u,t} s_f. \tag{3}$$

The service rate of the queue $Q_n(t)$ is expressed as the average rate at which the current request is satisfied. Based on the above description, the dynamic request queue $\{Q_n(t)\}^{\mathcal{H}}_{n=1}$ at any time slot t can be dynamically described as

$$Q_n(t+1) = \max\{Q_n(t) + A_n\left(b^n_{u,t}\right) - r^n_{u,t}, 0\} \tag{4}$$

Specifically, at the initial stage, the request queue maintained by user n is set to $Q_n(0) = 0$. In addition, since the system state is random, the system-dependent queuing vector process $\{Q_n(t)\}_{t \in \mathcal{T}}$ is also random.

When users select nodes, there is a cost associated with content sharing. This cost is related to the delivery rate of the content from the user to the selected node. We assume that the transmission cost is proportional to the data rate, which is a general performance metric that can be converted to other metrics such as battery life, transmission delay, and interference. Following similar calculations in prior studies [10], we assume that the transmission cost is charged per unit of data rate. Therefore, at any given time slot t, the transmission cost from the selected node n to the user u can be expressed as:

$$C_n\left(b^n_{u,t}\right) = b^n_{u,t} \cdot p^{trans}_{u,n} \cdot r^n_{u,t}, \tag{5}$$

where $p^{trans}_{u,n}$ is denoted as the transmission power consumption from user u to node n. To deal with the network cost of the delivery nodes in the network, virtual cost queues $Y(t) = \{Y_n(t) | n \in \mathcal{H}\}$ are introduced. Specifically, at the initial stage, the cost queue maintained by user n is set to $Y_n(0) = 0$. Based on the above description, at any time slot t, the dynamic cost queue $\{Y_n(t)\}^{\mathcal{H}}_{n=1}$ can be dynamically described as

$$Y_n(t+1) = max\{Y_n(t) + C_n\left(b^n_{u,t}\right) - \epsilon v, 0\}, \tag{6}$$

where ϵ is a positive scaling factor. Dynamic queue $Y_n(t)$ can be seen as a random energy consumption $C_n(b^n_{u,t})$ and a fixed service rate ϵv.

B. Problem Formulation

In this paper, the overall objective is to make dynamic node selection and caching decisions that maximise D2D sharing gains while minimising user request latency. The specific problem under consideration can be formulated as

1) **D2D shared gain**: If the user's request cannot be satisfied locally (i.e., $x_{u,f} = 0$), then by selecting the content sharing node $b^n_{u,t}$ establishes a D2D link with adjacent user n to meet user requests, where $n > 0$. Therefore, the gain obtained through the D2D sharing method can be expressed as

$$G^t_{u,1} = \sum_{n\in\mathcal{N}} s_f \cdot R_{u,n} \cdot b^n_{u,t} \cdot x_{n,f} \cdot \big(1 - x_{u,f}\big), \quad \forall u \in \mathcal{N}, \forall n \in \mathcal{H}, \forall t \in \mathcal{T}, \forall f \in \mathcal{F}. \tag{7}$$

2) **Content fetch gain**: To account for the fact that a User Equipment (UE) can only share content with a chosen node during a time slot, we introduce an average queue delay that is directly proportional to the serving UE. Consequently, the network's benefit from user acquisition requests can be expressed as follows:

$$G^t_{u,2} = \begin{cases} \tau e^{-d^L_{u,t}}, & Local\ Cache \\ \tau e^{-d^{D2D}_{u,t}}, & D2D\ Communication \\ \tau e^{-d^B_{u,t}}, & Communication\ to\ BS \\ \tau e^{-d^c_{u,t}}, & Cloud\ Service \end{cases} \tag{8}$$

Where τ represents the introduced parameter, we observe a negative exponential function that highlights the inverse relationship between user request delay and channel gain. In simpler terms, when the user request delay is low, a larger gain is obtained. By using this formulation, we can analyze the impact of user request delay on the network's benefit.

Based on the above description, the gain of the obtained content is represented as

$$\mathbb{G}\big(b^n_{u,t}, x_{u,f}\big) = \sum_{u}^{N} \Big\{ \lambda_1 G^t_{u,1} + \lambda_2 G^t_{u,2} \Big\}, \tag{9}$$

where λ_1 and λ_2 is the two introduced parameters, which satisfy the condition $\lambda_1 + \lambda_2 = 1$, where $0 \le \lambda_1, \lambda_2 \le 1$. These parameters represent the weights or proportions assigned to D2D gain and delay, respectively. Therefore, the goal of this article is to optimize the problem $\mathcal{P}_1$, can be expressed as

$$\mathbf{max}\ \mathbb{G}\big(b^n_{u,t}, x_{u,f}\big) \tag{10}$$

For ease of understanding, the notation used in this article is summarized in Table I.

TABLE I
MODELING PARAMETERS AND NOTATIONS

t	Index of the time slot.
$\mathcal{N} = \{1, 2, \ldots, N\}$	Set of user labels.
$\mathcal{H} = \{\mathcal{N} \cup \mathcal{B}\}$	All the cache node set in network and B is base station node.
$\mathcal{F} = \{1, 2, \ldots, F\}$	Set of all contents.
M_i, M_B	Cache capacities of UE-i and the MEC server, respectively.
$l_{u,t} = \{x_{u,t}, y_{u,t}\}$	User u position in time slot t and $x_{u,t}$ and $y_{u,t}$ is the coordinate of user u, respectively.
R_u,R_B	Service scope of users and base stations.
P_u	User preferences.
g^n_t, g^B_t	Channel gain.
S^v_u	Preference similarity between user u and user v.
x_u	User cache state.
$b^n_{u,t} = 1$	Indicates that user u selects cache node n to process user requests.
s_f	Request the size of file f.
$r^n_{u,t}$	The transmission rate processed by node n for user u.
$d^L_{u,t}, d^D_{u,t} 2D, d^B_{u,t}, d^c_{u,t}$	Transmission delay.
$A_n(b^n_{u,t})$	Arrival rate of queue n.
$Q(t)$, $Y(t)$	Dynamic of the queue backlog and virtual cost queue for delivery nodes.
$h_u(t)$	Channel gain between user u and other nodes.
$C_n(b^n_{u,t})$	The transmission cost from the selected node n to the user u.
$p^{trans}_{u,n}$	Transmission power consumption from user u to node n.
ϵv	fixed service rate .

V. DYNAMIC CONTENT CACHE AND NODE SELECTION ALGORITHM

A. Stochastic Lyapunov Optimization

Directly solving the aforementioned problem $\mathcal{P}_1$ becomes a challenging task without prior knowledge of the system's status, queue backlog, and network cost distribution. In addition, the choice of content sharing delivery nodes creates an imbalance in the request queue and delivery cost of caching nodes in the network. In order to tackle the aforementioned challenges, we employ stochastic Lyapunov optimization methods to regulate the selection of content sharing nodes. The primary objective of this approach is to minimize the average network latency while ensuring the stability of both the request and cost queues within the network, all without relying on any prior knowledge. We started by defining the functions used in our analysis.

Definition 1 (Quadratic Lyapunov Function): In order to jointly control the request and network cost queues of any time slot *t*, the total queue is defined as $Z(t) = \{Q_n(t), Y_n(t)\}$, where $\{Q_n(t)\}^{\mathcal{H}}_{n=1}$ and $\{Y_n(t)\}^{\mathcal{H}}_{n=1}$. The quadratic Lyapunov function $L(Z(t))$ of the random queuing process is equal to half the sum of the squares of the backlogs of all current queues, which is

$$L(Z(t)) = \frac{1}{2}\left(\sum_{n\in\mathcal{H}} Q_n(t)^2 + \sum_{n\in\mathcal{H}} Y_n(t)^2 \right). \tag{11}$$

The Lyapunov function is a scalar measure of the total queue backlog in the network, with a smaller $L(Z(t))$ indicating a lower queue occupancy in the network.

Definition 2 (Conditional Expectation Lyapunov Drift): At any time slot *t*, the conditional expectation Lyapunov drift ΔH_t represents the expectation of the time slot difference of the Lyapunov function, namely

$$\Delta H_t = E\{L(Z(t+1)) - L(Z(t)) | Z(t)\}, \tag{12}$$

which ΔH_t describes the variation of the quadratic Lyapunov function, i.e., the degree of fluctuation of the function. A smaller ΔH_t indicates a more stable queue in the network system. Therefore, we choose to minimise ΔH_t for each time slot *t* to stabilise the whole system network. However, if one wishes to stabilize the request and energy consumption queues in the network while minimizing average latency, one must add the expected cost $E\{\mathbb{G}(b^n_{u,t}, x_{u,f})\}$ to ΔH_t [10] and

then, transform the function that minimises ΔH_t, into a cost function that maximises the dift-plus-cost

$$V \cdot E\{\mathbb{G}(b_{u,t}^n, x_{u,f})|Z(t)\} - \Delta H_t, \tag{13}$$

where the weights $V \geq 0$ to balance the impact on network cost and network stability. In the following, to obtain an upper bound on the drift plus cost in (13) within an arbitrary time slot t. Firstly, we have

$$\begin{aligned} Q_n(t+1)^2 = {} & Q_n(t)^2 + 2Q_n(t)\big(A_n(b_{u,t}^n) - r_{u,t}^n\big) \\ & + \big(A_n(b_{u,t}^n) - r_{u,t}^n\big)^2, \end{aligned} \tag{14}$$

$$\begin{aligned} Y_n(t+1)^2 = {} & Y_n(t)^2 + 2Y_n(t)\big(C_n(b_{u,t}^n) - \epsilon v\big) \\ & + \big(C_n(b_{u,t}^n) - \epsilon v\big)^2. \end{aligned} \tag{15}$$

Secondly, by combining equation (4), (6), equation (12) can be rewritten as

$$\begin{aligned} \Delta H_t = \sum_{n\in\mathcal{H}} \frac{1}{2} E\Big\{ & \big(A_n(b_{u,t}^n) - r_{u,t}^n(b_{u,t}^n)\big)^2 + Q_n(t) \cdot \\ & \big(\big(A_n(b_{u,t}^n) - r_{u,t}^n\big)|Q_n(t)\big) + \frac{1}{2}\big(C_n(b_{u,t}^n) - \epsilon v\big)^2 \\ & + Y_n(t) \cdot \big(\big(C_n(b_{u,t}^n) - \epsilon v\big)|Y_n(t)\big)\Big\}. \end{aligned} \tag{16}$$

Therefore, at any slot t, the drift-plus-cost function in (13) is upper-bounded by

$$\begin{aligned} & V \cdot E\{\mathbb{G}(b_{u,t}^n, x_{u,f})|Z(t)\} - \Delta H_t \\ & \leq V \cdot E\{\mathbb{G}(b_{u,t}^n, x_{u,f})|Z(t)\} \\ & \quad - \Bigg(B + \sum_{n\in\mathcal{H}} Y_n(t) \cdot \big(\big(C_n(b_{u,t}^n) - \epsilon v\big)|Y_n(t)\big) \\ & \qquad + \sum_{n\in\mathcal{H}} Q_n(t) \cdot \big(\big(A_n(b_{u,t}^n) - r_{u,t}^n\big)|Q_n(t)\big)\Bigg), \end{aligned} \tag{17}$$

where B is a constant independent of V and the $Q_n(t)$ and $Y_n(t)$ in the total queue $Z(t) = \{Q_n(t), Y_n(t)\}$ are independent of each other. Therefore, define $B = B_1 + B_2$, where B_1 and B_2 can be obtained separately from

$$\begin{aligned} & \frac{1}{2} E\Bigg\{ \sum_{n\in\mathcal{H}} \big(A_n(b_{u,t}^n) - r_{u,t}^n\big)^2 \Bigg\} \\ & \leq \frac{1}{2} \sum_{n\in\mathcal{H}} E\Big[\big(A_n(b_{u,t}^n)\big)^2 + \big(r_{u,t}^n\big)^2\Big] \\ & \leq \frac{1}{2} N \cdot \big(s_f\big)^2 + \sum_{n\in\mathcal{H}} \big(r_{u,t}^n\big)^2 = B_1, \end{aligned} \tag{18}$$

$$\begin{aligned} & \frac{1}{2} E\Bigg\{ \sum_{n\in\mathcal{H}} \big(C_n(b_{u,t}^n) - \epsilon v\big)^2 \Bigg\} \\ & \leq \frac{1}{2} \sum_{n\in\mathcal{H}} E\Big[\big(C_n(b_{u,t}^n\big)^2 + (\epsilon v)^2\Big] \\ & \leq \frac{1}{2} \sum_{n\in\mathcal{H}} \big(p_{u,n}^{trans} \cdot r_{u,t}^n\big)^2 + N \cdot (\epsilon v)^2 = B_2. \end{aligned} \tag{19}$$

The terms of the second inequality in (18) and (19) relate to the content sharing node $b_{u,t}^n$ selected by user u. Instead of minimizing the dift-plus-cost function in (13), we minimize its upper-bound function. Therefore, in order to minimize the right-hand side of the inequality (17), it is necessary to consider the current historical request queue $Q(t)$ and energy cost queue $Y(t)$ in time slot t. This can be achieved by selecting the appropriate shared delivery Node b_t^n. Then, we obtain the following optimization problem $\mathcal{P}_2$ max $\Omega(b_{u,t}^n, x_{u,f}|Z(t))$, where the objective function

$$\begin{aligned} & \Omega\big(b_{u,t}^n, x_{u,f}|Z(t)\big) \\ & = V \cdot E\{\mathbb{G}(s^t, a^t)|Z(t)\} - \sum_{u}^{N} \sum_{n\in\mathcal{H}} \\ & \quad Q_n(t) \cdot A_n\big(b_{u,t}^n|s_u^t\big) - Y_n(t)\ C_n\big(b_{u,t}^n|s_u^t\big) \\ & = \sum_{u}^{N} \Bigg(\sum_{n\in\mathcal{H}} \Big(s_f \cdot R_{u,n} \cdot b_{u,t}^n \cdot x_{n,f} \cdot (1 - x_{u,f}) + \tau e^{-d_{u,t}^n} \\ & \quad - Q_n(t) \cdot b_{u,t}^n \cdot s_f - Y_n(t) \cdot b_{u,t}^n \cdot p_{u,n}^{trans} \cdot r_{u,t}^n \Big)\Bigg). \end{aligned} \tag{20}$$

$$s.t. b_{u,t}^n, n \in \{\mathcal{N} \cup B\}/u, u \in \mathcal{N} \tag{19a}$$

$$\sum_{n\in\{\mathcal{N}\cup B\}} b_{u,t}^n = 1 \tag{19b}$$

B. Deep Reinforcement Learning for Shared Delivery Node Selection and Cache Replacement

Notice that the problem $\mathcal{P}_2$ is linear and, therefore, decomposable. In particular, we can then decompose this problem into N subproblems, given by

$$\begin{aligned} & \Omega^u\big(b_{u,t}^n, x_{u,f}|Z(t)\big) \\ & = \sum_{n\in\mathcal{H}} \Big(V \cdot \Big(s_f \cdot R_{u,n} \cdot b_{u,t}^n x_{n,f} \cdot (1 - x_{u,f}) + \tau e^{-d_{u,t}^n} \Big) \\ & \qquad - Q_n(t) \cdot b_{u,t}^n \cdot s_f - Y_n(t) \cdot b_{u,t}^n \cdot p_{u,n}^{trans} \cdot r_{u,t}^n \Big), \end{aligned} \tag{21}$$

for all users $u \in \mathcal{N}$, which can be solved in parallel by the users separately. Therefore, solving $\mathcal{P}_2$ is synonymous with finding the optimal content delivery node and caching policy, i.e.,

$$\mathcal{P}_3 \text{ arg max } \Omega^u\big(b_{u,t}^n, x_{u,f}|Z(t)\big). \tag{22}$$

It is worth noting that users need to make dynamic node selection and caching decisions under constantly changing channel conditions. We transform the joint optimization problem $\mathcal{P}_3$ of content delivery node selection and cache replacement into a Markov decision process (MDP), as shown below:

State: The state of user at time t can be expressed as $s_u^t = \{Y_u(t), Q_u(t), R_u, h_u(t)\}$. $Y_u(t), Q_u(t)$ denote the energy consumption queue and request queue of user u, respectively. In addition, R_u is the probability of a user establishing a D2D connection with neighbouring user, and $h_u(t)$ is denoted as the channel gain of the user requesting content to other delivery nodes.

Action: After receiving the status s_u^t, select the content sharing transmission node and replace the file. Therefore, the

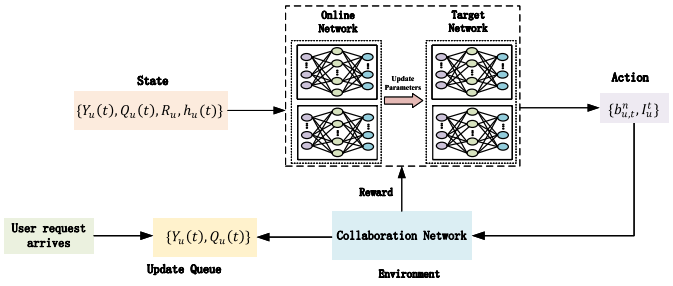

Fig. 3. DDPG architecture for solving the proposed MDP problem.

action can be expressed as $a_u^t = \{b_{u,t}^n, I_u^t\}$, where $b_{u,t}^n$ is the indication identifier indicating the node (e.g., including UE's neighbouring nodes and base station). I_u^t is the indicator whether the file in cache list need to be replaced.

Reward value of the system: Our goal is to maximize revenue from D2D and content sharing while reducing user request latency and maintaining the stability of the request and energy queues in the system. Therefore, we set the reward as the optimization problem $\mathcal{P}_3$, i.e., $r_u = \Omega^u(b_{u,t}^n, x_{u,f}|Z(t))$.

In general, obtaining the optimal action, including content node selection decision $(b_{u,t}^n)^*$ and cache replacement I_u^t, involves exploring a vast number of possible decisions, which can be as high as $2^{(N+M_u)}$ options. This results in significant computational complexity even when N is very small. As a result, a reinforcement learning approach is employed to enable online shared node selection and cache decisions. Specifically, we use the deep deterministic policy gradient (DDPG) reinforcement learning algorithm to dynamically make node selection and cache decisions, thus solving the formulated MDP problem.

Fig. 3 illustrates the architecture for addressing the proposed MDP problem in DDPG. The DDPG algorithm employs two independent deep neural networks (DNNs), which follow the actor-critic paradigm, including an online network and a target network. In each selection period t, the state s_u is sent to the actor network of the online network. To ensure comprehensive exploration of the environment while maintaining a balance between exploration and exploitation, a Gaussian noise vector is incorporated into the action policy function during its output. Following the execution of action a_u, the ensuing reward r_u and the subsequent state $s'_u(t)$ are observed. The observed Transition state $(s_u(t), a_u(t), r_u(t), s'_u(t))$ is saved to the experience pool to facilitate subsequent network learning.

The target network, which acts as a delayed replica of the online network, progressively tracks the acquired knowledge and updates the parameter configuration of the target network model through a process of soft updates. Throughout the training of the system, the agent randomly samples from the experience replay pool. In an effort to ensure the proximity of the Q-value output produced by the critic network to the actual value, the Q-value estimated by the critic network is employed within the target network. Additionally, the mean square error loss function is utilized to guide the training of the actor network. The model is trained Episode times, and in each epoch, the target network implements a soft update mechanism to modify network parameters. The detailed LoRL Algorithm is shown in Algorithm 1.

Algorithm 1 Combines Lyapunov Optimization and Reinforcement Learning (LoRL) for Solving $(\mathcal{P}_3)$

input: Parameters V.
output: Control actions $a_u^t = \{b_{u,t}^n, I_u^t\}$.
initialization: $\{h_u\}_{u=1}^{\mathcal{H}}$, $Y_u(t)\}_{u=1}^{\mathcal{H}}$, $\{Q_u(t)\}_{u=1}^{\mathcal{H}}$, $Y_u(0) = 0$, $Q_u(0) = 0$.
for UE-$u \in \mathcal{N}$ in Parallel: **do**
  Training AutoEncoder model, and then obtain user preferences P_u.
end for
for each epoch in episode **do**
  for $t = 0, 1, 2, 3, ..., T$ **do**
    for UE-$u \in \mathcal{N}$ in Parallel: **do**
      User u sends a request file f.
      IF $x_{u,f} = 1$:
        continue.
      Calculate the similarity S_u^v (12) between user u and neighboring users.
      Calculate the connection probability between users R_u, $\frac{R_{u,v}}{\sum_{u \in \mathcal{N}} R_{u,v}} \rightarrow R_u$.
      Observer the state $s_u^t = \{Y_u(t), Q_u(t), R_u, h_u(t)\}$.
      Observer reward feedback $r_u(t)$ (21), and obtain new observations $s'_u(t)$.
      Construct transition $(s_u(t), a_u(t), r_u(t), s'_u(t))$ and then store the transition into experience pool.
      Update actor-critic model parameters
    end for
  end for
end for

C. Complexity Analysis

The computational complexity analysis of the proposed LoRL scheme is as follows: The execution of the LoRL algorithm consists of two parts, namely, joint user association and cache decision, and policy update. Between these two parts, a joint decision action generation is performed in each time frame, while policy updates are less frequent. Therefore, we focus on analyzing the complexity of policy decision generation in each time frame.

Careful observation reveals that within each time slot, the algorithm's complexity includes the computation of similarity for each user (12), the probability of connections between users, and the updating of DDPG model parameters. Specifically, the complexity of computing similarity and user connection probability is O(2N), where N is the number of users.

Additionally, the time complexity of the DDPG algorithm, which mainly consists of initialization, memory replay, and four deep neural networks, is as follows. The state is initialized at the beginning of each training episode, with a time complexity of K. Additionally, both the actor network and the critic network are designed as fully connected networks,

assuming the actor network has N_A fully connected layers, and the critic network is composed of N_C fully connected layers. Therefore, the time complexity of DDPG is

$$2\sum_{i=0}^{N_A-1} u_{a,i}u_{a,i+1} + 2\sum_{j=0}^{N_C-1} u_{c,j}u_{c,j+1} + v_a u_i + K = \mathcal{O}\left(\sum_{i=0}^{N_A-1} u_{a,i}u_{a,i+1} + \sum_{j=0}^{N_C-1} u_{c,j}u_{c,j+1}\right) + \mathcal{O}(K) \tag{23}$$

Based on the description above, the time complexity of the proposed LoRL algorithm is $\mathcal{O}(\sum_{i=0}^{N_A-1} u_{a,i}u_{a,i+1} + \sum_{j=0}^{N_C-1} u_{c,j}u_{c,j+1}) + \mathcal{O}(K) + \mathcal{O}(2N)$.

VI. Simulation Results

A. Parameter Setting

In our simulation study, we utilized the MovieLens 1M dataset [34] to model the request behavior in the network. This dataset contains user ratings for a total of 3952 movies. Each record in the dataset includes a user ID, a movie ID, a rating, and a timestamp. Since user ratings are typically provided after viewing, we treated these ratings as request records for our simulation. To calculate user preferences, we divided the dataset into two parts. The period from January 1, 2000 to April 13, 2002 was used as a historical training set to obtain user preferences. The remaining data served as the test set to evaluate the performance of our algorithm. The content database $\mathcal{F}$ consists of the 3952 movies contained in the dataset. In order to reflect the degree of queue backlog in the system network, we set the total number of requests to be 10,000 under different numbers of users. We set the default cache sizes of user nodes and base stations to 40M and 100M, respectively. These cache sizes determine the amount of content that can be stored locally at each node.

In our simulation, we modeled the user's movement trajectory using the Random Waypoint model [35], which is a widely applied and proven effective approach in simulating user mobility. The Random Waypoint model has also been commonly used in other studies, particularly in research related to caching strategies [36], [37]. This model generates random coordinate locations for the user at each time slot within a specified region. To determine the range of D2D connections, we assumed that users can establish connections within a physical range limit of 100 units. Users can connect to the base station within a range of 500 units. The path loss in the network was modeled using the formula $36.8 + 36.7\log(d)$, where d represents the distance between user cache nodes. The small-scale fading was modeled using the unit variance of Rayleigh fading. Other network parameters included a channel bandwidth of 20 MHz and a background noise level of -95 dBm. In addition, all users run a DRL agent with a three-layer neural network. All these agents use the Adam optimizer with adaptive learning rates to learn their respective training parameters, starting from a learning rate of 10^{-2}. For specific simulation parameter settings of DDPG, please refer to Table II.

TABLE II
Model Simulation Parameters

Parameter	Value
learning rate	0.01
action dim	400
critic dim	200
batch size	128
gamma	0.99
Exploration rate	0.1
Hidden layer	3
Memory Capacity	2000
Activation function	ReLU

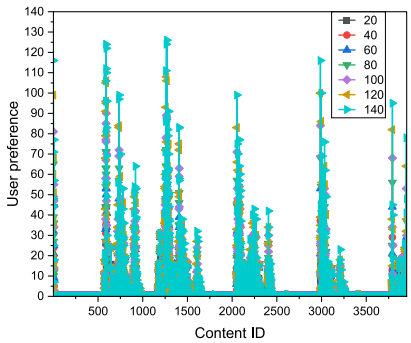

Fig. 4. Variation in request preference at different numbers of users.

B. Content Request Preference Analysis

In our experiment, we simplified the dataset by extracting the user IDs with the highest number of requests. We then selected a specific number of users for simulation purposes. User preference in this context refers to users' focus on requesting certain types of files. To illustrate how user preference changes with different numbers of users, we provide a simple example. In the example, we consider the numbers of users to be [20, 40, 60, 80, 100, 120, 140]. Fig. 4 shows the distribution of requested content IDs at different numbers of users. From the figure, we observe that the content popularity exhibits certain preferences for different numbers of users. The requested content IDs are clustered within specific ranges, i.e., [500-1000], [1250-1500], [2000-2500], [3000-3250], [3725-3952]. This indicates that certain content types or categories are more popular among users, and their preferences can be observed based on the content IDs requested.

C. Baseline Schemes and Performance Metrics

To evaluate the performance of the proposed LoRL algorithm under different parameters, we consider the following baseline scheme:

1) *LRU:* Randomly select content sharing nodes within the communicable range, and the earliest stored content will be replaced with new content.
2) *MLPLRU* [38]: Randomly select content sharing nodes within the communicable range, and an improved online cache replacement strategy based on Pareto Least Recently Used (PLRU) Algorithm and Least Recently Used K (LRU-K) Algorithm.
3) *LoRL-with no preference:* LoRL-with no preference refers to the variant of LoRL algorithm that disregards

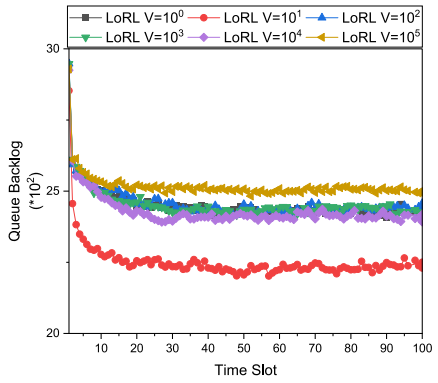

Fig. 5. Dynamics of the time-averaged sum of queue backlogs in LoRL for different weights V during the observation period $t = 1, \ldots, 100$ slots.

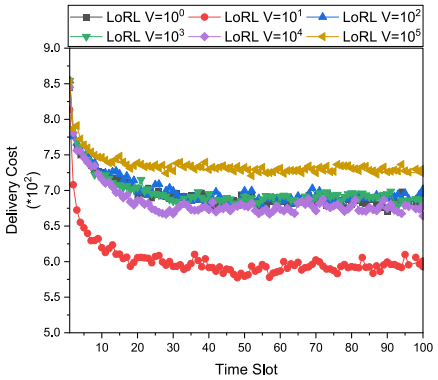

Fig. 6. Dynamics of the time-averaged network cost in LoRL for different weights V during the observation period $t = 1, \ldots, 100$ slots.

user preferences and solely focuses on the connection probability between users.

4) *DAC:* A delay-aware D2D caching (DAC) [39] algorithm with the goal of request latency.
5) *GHM:* The greedy heuristic method (GHM) [10] searches within a limited number of user/file pairs to maximize the target D2D gain and delay value.

To evaluate these solutions, we use the following performance metrics: (i)hit rate (satisfied by local cache, D2D sharing, or BS); (ii) average delay; (iii) queue backlog; (iv) network cost; (v) D2D offloading rate.

D. The Impact of Weight V

Figures 5-7 verify the analysis results related to Lyapunov optimization established in (17). Figures 5 and 6 show the dynamics of the time-averaged network cost and sum of queue backlogs during different values of weight V, respectively, in LoRL during the period $t = 1, \ldots, 100$. It is observed that LoRL converges to stable delivery cost and queue backlog levels around time slot $t = 30$. We also see that when weight $V = 10$, there is lower network cost and queue backlog. Figure 7 illustrates the target value, which is the time average of the sum of transmission cost and queue backlog, as a function of the weight V in Lyapunov optimization. The graph demonstrates a proportional relationship between the sum of queue backlog and network cost. This can be attributed to

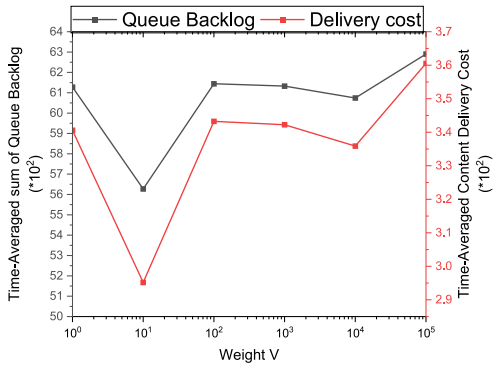

Fig. 7. Time averages of the content transfer cost and the sum of queue backlogs in LoRL as functions of weight V.

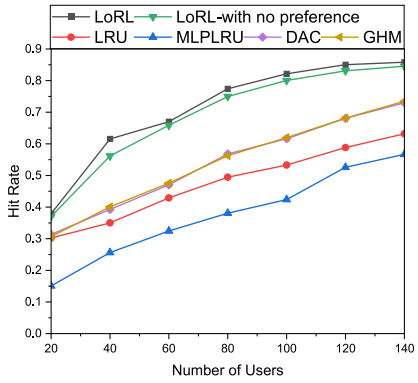

Fig. 8. The cache hit ratio varies with the number of users.

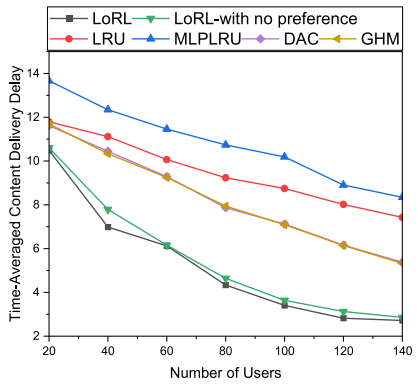

Fig. 9. Time-average transmission delay varies with the number of users.

the fact that a higher queue backlog prompts the selection of delivery nodes with lower transmission delays to minimize retrieval costs, thereby resulting in an increase in delivery costs.

E. The Impact of User Numbers

Results in Figures 8-12 describe the performance of the cache strategy in terms of cache hit ratio, time-averaged latency, time-averaged delivery cost, and time-averaged queue backlog under different numbers of users. The Figure 8 shows that the higher the number of users, the higher the cache hit rate.As depicted in Figures 9-11, it is evident that with the expansion of the network user scale, the queue backlog,

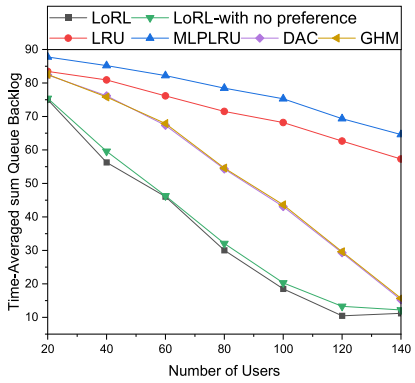

Fig. 10. Time-average sum of queue backlogs with varies with the number of users.

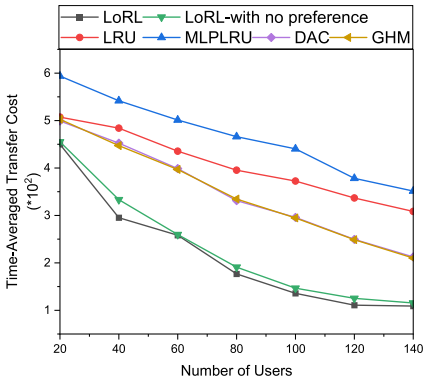

Fig. 11. Time-average network cost varies with the number of users.

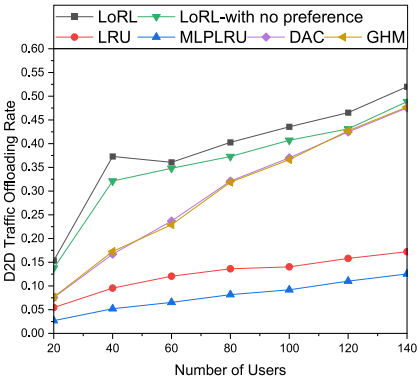

Fig. 12. The Device-to-Device (D2D) offloading rate varies with the number of users.

transmission cost, and average request delay decrease accordingly. This can be attributed to the increased opportunities for D2D communication resulting from a larger number of users, thereby enhancing the effectiveness of collaborative caching. It is worth noting that it is observed from Figure 12 that as the number of users increases, the D2D offloading rate has a slow decreasing trend, which is mainly because our proposed mechanism combines user preference node selection and cache decision, local The cache decision is more biased towards local requests, increasing the probability of local cache hits, thereby reducing the proportion of D2D offloading. Furthermore, it can be seen from these figures that although the proportion

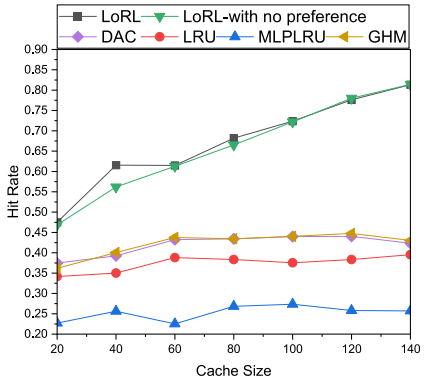

Fig. 13. The cache hit ratio varies with cache size.

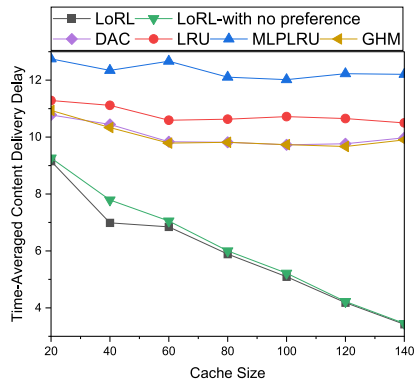

Fig. 14. Time-averaged transmission delay varies with cache size.

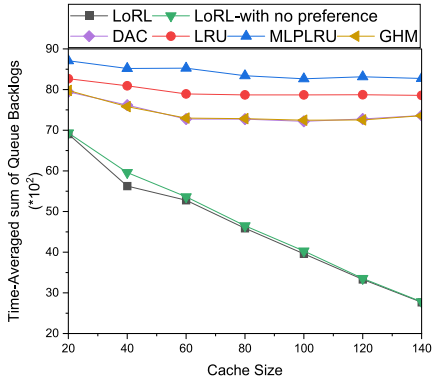

Fig. 15. Time-averaged sum of queue backlogs varies with cache size.

of D2D offloading decreases, the proposed mechanism still outperforms other strategies.

F. The Impact of Cache Size

In order to compare the performance of LoRL with other caching strategies, we present the results in Figures 13-17 with respect to cache size. These figures display the hit rate, request latency, time-averaged network cost, and the sum of queue backlog and D2D offload ratio. It is evident that the cache hit ratio improves with larger cache sizes. Conversely, the combined metrics of queue backlogs, transmission delays, and network costs decrease as the user cache size increases. As can be seen from these figures, LRU and MLPLRU

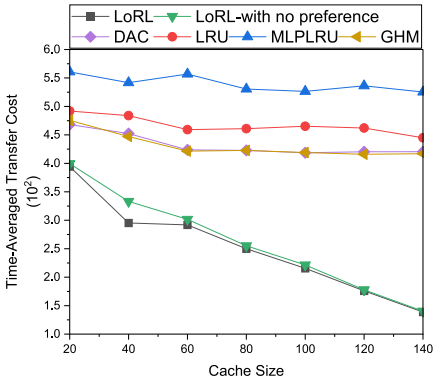

Fig. 16. Time-averaged network cost varies with cache size.

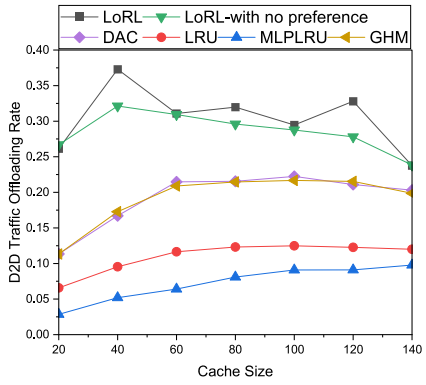

Fig. 17. The Device-to-Device (D2D) offloading rate varies with cache size.

are not very effective in minimizing content delivery delay, transmission cost, and queue backlog. The reason is that LRU and MLPLRU do not consider the popularity of file content, and only update the cache according to local requests. Also, these algorithms randomly select shared nodes from within the exchange range. In addition, both the GHM algorithm and the DAC algorithm are greedy node selection algorithms, which can improve the content sharing rate by optimizing the D2D node selection, but it does not consider the local cache decision, so its local cache hit rate is low. We also observed that the DAC and GHM algorithms are inefficient, leading to performance degradation at the expense of queue backlog and network transmission costs. The main reason is that when the number of users is fixed, the cache nodes that users can establish D2D links are limited, so GHM is compared to DAC advantage is not particularly obvious. In particular, LoRL has certain performance advantages compared to LoRL with no preference, and can better select shared nodes with high D2D gains, making the network have lower queue backlogs and transmission costs.

VII. Conclusion

In this paper, we introduce a novel approach for selecting shared nodes and making cache decisions in D2D-assisted heterogeneous collaborative edge computing networks. Our approach involves formulating a joint optimization problem and utilizing a Lyapunov optimization algorithm to decouple the problem. To enable intelligent user association and caching decisions, we propose a content caching algorithm based on deep deterministic policy gradient (DDPG) [33]. The algorithm aims to minimize user request latency while ensuring the stability of request and energy consumption queues in the system. To evaluate the effectiveness of our proposed algorithm, we conduct an extensive study and compare it with five baseline schemes. The results demonstrate that our algorithm surpasses the baseline schemes in terms of average content download latency and system queue stability. In our future work, we intend to delve deeper into the fine-grained characteristics of users and requests. This will enable us to make more informed decisions regarding user association and caching, ultimately enhancing the overall performance of the system.

References

[1] *Traffic Estimates for the Years 2020 to 2030*, ITU-Rec. M.2370-0, Int. Telecommun. Union, Geneva, Switzerland, 2015.

[2] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4309–4324, Oct. 2017.

[3] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. M. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 154–169, Jan. 2020.

[4] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1710–1732, 3rd Quart., 2018.

[5] B. Chen and C. Yang, "Caching policy for cache-enabled D2D communications by learning user preference," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6586–6601, Dec. 2018.

[6] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, Aug. 2021.

[7] M. S. Ali, P. Coucheney, and M. Coupechoux, "Learning in noisy-potential games for resource allocation in D2D networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2761–2773, Dec. 2020.

[8] L. Li et al., "Deep reinforcement learning approaches for content caching in cache-enabled D2D networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 544–557, Jan. 2020.

[9] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative D2D local model aggregations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3851–3869, Dec. 2021.

[10] A. Asheralieva and D. Niyato, "Combining contract theory and Lyapunov optimization for content sharing with edge caching and device-to-device communications," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1213–1226, Jun. 2020.

[11] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable Online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[12] 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 Things J.*, vol. 7, no. 10, pp. 9413–9425, Oct. 2020.

[13] Y. Jiang, M. Ma, M. Bennis, F.-C. Zheng, and X. You, "User preference learning-based edge caching for fog radio access network," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1268–1283, Feb. 2019.

[14] Y. Jiang, Y. Wu, F.-C. Zheng, M. Bennis, and X. You, "Federated learning based content popularity prediction in fog radio access networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 3836–3849, Jun. 2021.

[15] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3520–3535, Jun. 2017.

[16] N. B. Hassine, R. Milocco, and P. Minet, "ARMA based popularity prediction for caching in content delivery networks," in *Proc. Wireless Days*, 2017, pp. 113–120.

[17] Q. Chen, W. Wang, F. R. Yu, M. Tao, and Z. Zhang, "Content caching oriented popularity prediction: A weighted clustering approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 623–636, Jan. 2021.
[18] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.
[19] S. Li, J. Xu, M. van der Schaar, and W. Li, "Popularity-driven content caching," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[20] J. Kwak, Y. Kim, L. B. Le, and S. Chong, "Hybrid content caching in 5G wireless networks: Cloud versus edge caching," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3030–3045, May 2018, doi: 10.1109/TWC.2018.2805893.
[21] Y. Qian, R. Wang, J. Wu, B. Tan, and H. Ren, "Reinforcement learning-based optimal computing and caching in mobile edge network," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2343–2355, Oct. 2020.
[22] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.
[23] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.
[24] Q. Chen, W. Wang, W. Chen, F. R. Yu, and Z. Zhang, "Cache-enabled multicast content pushing with structured deep learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2135–2149, Jul. 2021.
[25] S. Liu, C. Zheng, Y. Huang, and T. Q. S. Quek, "Distributed reinforcement learning for privacy-preserving dynamic edge caching," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 749–760, Mar. 2022.
[26] W. Huang, W. Chen, and H. V. Poor, "Request delay-based pricing for proactive caching: A stackelberg game approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 2903–2918, Jun. 2019.
[27] Z. Xu et al., "Stable service caching in MECs of hierarchical service markets with uncertain request rates," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4279–4296, Jul. 2023, doi: 10.1109/TMC.2022.3149870.
[28] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.
[29] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 19–31, Jan. 2021.
[30] T. Zhang, Y. Wang, W. Yi, Y. Liu, C. Feng, and A. Nallanathan, "Two time-scale caching placement and user association in dynamic cellular networks," *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2561–2574, Apr. 2022.
[31] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2525–2553, 3rd Quart., 2019.
[32] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. M. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE J. Sel. Areas Commun.*., vol. 36, no. 8, pp. 1768–1785, Aug. 2018.
[33] T. P. Lillicrap et al, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[34] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2015.
[35] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model," *Wireless Netw.*, vol. 10, no. 5, pp. 555–567, 2004.
[36] B. Banerjee and C. Tellambura, "Study of mobility in cache-enabled wireless heterogeneous networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2017, pp. 1–6.
[37] C. Casetti, C.-F. Chiasserini, M. Fiore, C.-A. La, and P. Michiardi, "P2P cache-and-forward mechanisms for mobile Ad Hoc networks," in *Proc. IEEE Symp. Comput. Commun.*, Sousse, Tunisia, 2009, pp. 386–392, doi: 10.1109/ISCC.2009.5202293.
[38] F. Miao, D. Chen, and L. Jin, "Multi-level PLRU cache algorithm for content delivery networks," in *Proc. 10th Int. Symp. Comput. Intell. Design (ISCID)*, vol. 1, 2017, pp. 320–323.
[39] Y. Li, M. C. Gursoy, and S. Velipasalar, "A delay-aware caching algorithm for wireless D2D caching networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2017, pp. 456–461.

Ziyi Teng (Student Member, IEEE) received the B.S. degree from the North China University of Water Resources and Electric Power, Zhengzhou, China, in 2019. She is currently pursuing the Ph.D. degree with the Beijing University of Technology, Beijing, China. Her research interests in mobile edge computing. She is a student member of CCF.

Juan Fang (Member, IEEE) received the M.S. degree from the Jilin University of Technology, Changchun, China, in 1997, and the Ph.D. degree from the College of Computer Science, Beijing University of Technology, Beijing, China, in 2005. In 1997, she joined the College of Computer Science, Beijing University of Technology, where she has been a Professor since 2015. She currently works with the Faculty of Information Technology, Beijing University of Technology. Her research interests include high performance computing, heterogeneous intelligent computing, and edge computing.

Yaqi Liu (Student Member, IEEE) received the B.S. degree from the Beijing University of Technology, Beijing, China, in 2020, where she is currently pursuing the Ph.D. degree. Her research interests in mobile edge computing. She is a student member of CCF.