Dynamic Edge Caching via Online Meta-RL

Yinan Mao*, Shiji Zhou†, Haochen Liu§, Zhi Wang*†, Wenwu Zhu†
* *Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China*
† *Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen, China*
§ *School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore*

***Abstract*—The content request patterns perceived by edge devices are becoming highly dynamic, especially for emerging short video platforms compared to traditional video platforms. This calls for caching policies that can *continuously* adapt to dynamic environments, challenging previously popular reinforcement learning (RL)-based policies. A straightforward solution, i.e., repeatedly restarting and training RL agents, would fail to converge timely while meeting the observed adaptation process. Offering transferable knowledge is considered a possible method to speed up the adaptation process. Unfortunately, it fails to outperform the RL-based approach as an alternative solution in these scenarios. To alleviate this drawback, we 1) design a *sequential-pair* meta-learning for edge caching that captures the meta-knowledge of dynamic changes from sequential-pair-wise intervals, which are segmentations from the whole dynamic episode, and 2) develop an *online* meta-RL-based solution called Online Meta Actor-Critic (OMAC), which updates the meta-knowledge in an online manner. To evaluate the proposed framework, we conduct trace-driven experiments to demonstrate the effectiveness of our design: it improves the average cache hit rate by up to 37.4% (normalized) compared with other baselines.**

***Index Terms*—edge content delivery, dynamic caching policy, meta-learning, reinforcement learning**

I. Introduction

Video streaming has achieved skyrocketing growth in recent years, accounting for 80% of the whole internet traffic and being triple by the end of 2022 [1], [2]. Moving content distribution to the *edge* of the internet can alleviate the backbone workload and improve the quality of experience for streaming users [3]. Such video content serving is usually referred to as *edge content delivery*.

With the rapid revolution of video platforms, the request patterns in edge devices have shown to be *much more dynamic* compared with the traditional ones [4]. For instance, our measurement study of KuaiShou (c.f. Sec. III), one of the most prominent short video platforms, shows the content request pattern difference between days measured by Kullback–Leibler (KL) divergence increases by 35.1%, compared with traditional video platforms. This significant value indicates that the request pattern varies over time.

Such dynamic request patterns have challenged existing content caching methods in edge content delivery: **Traditional methods with prior assumptions**, including Least Recently Used (LRU), Least Frequently Used (LFU), and their variants [5]–[7], fail to adapt to the dynamic environment as they are based on simple rule-based policies with some manually set features. **Reinforcement learning (RL)-based caching policies** [8]–[10] have been verified to achieve a higher hit rate than conventional rule-based policies. However, they are designed with the assumption of a stationary and static environment. In contrast, this assumption contradicts the dynamics of edge request patterns and has been illustrated by our measurements (c.f. Sec. III-A). When it is applied in dynamic edge content delivery, this assumption cannot be ensured, which usually causes the performance drop, as illustrated in Fig. 1(top) and observed in Fig. 5. One possible reason is that the dynamic request pattern would gradually impair the stationary assumption, leading to a continuous degrade of real-time performance. **Remedy studies on RL-based methods**, including ones using manual features [11]–[13] and others using a recursive network architecture to extract the dynamic features [14], [15], still suffer from the dramatically changing environment of edge scenarios.

To get rid of the outdated data influence caused by edge dynamics, a straightforward solution is to restart and train the RL model repeatedly with new edge samples. However, the RL model needs a pretty long adaptation time to attain a good performance (c.f. Sec. III-B); therefore, we need a new framework to speed up this process. We consider offering some transferable knowledge for the RL model when a new dynamic task is upcoming. To face the challenges above, *meta-RL* is expected to be a promising solution as it has been verified to achieve great adaption for new environments in other field works [16], [17]. However, directly using meta-RL would even perform worse than vanilla RL methods in edge caching, as illustrated in Tab. II. One possible explanation could be that plain meta-RL is designed for multi-task adaptation in static environment, which would ignore the dynamic feature in the edge caching.

How to make meta-RL suitable for this dynamic edge environment? We give two key design objectives here: 1) The meta-learning should capture the transferable temporal knowledge to enable adaptation to a dynamic environment; 2) The meta-knowledge should be updated continually to maintain timeliness. To achieve 1), we reformulate meta-RL by *segmenting* the dynamic episode into a sequence of short intervals, and then leverage meta-RL to gain the knowledge of the dynamic changes from *sequential-pair* intervals, as shown in Fig. 1(bottom). For 2), we propose an *online* meta-learning paradigm based on the Online Gradient Descent (OGD) to

Corresponding author: Zhi Wang, wangzhi@sz.tsinghua.edu.cn. This research was funded by Shenzhen Science and Technology Program (Grant No. RCYX20200714114523079 and JCYJ20220818101014030). We would like to thank Kuaishou for sponsoring the research.

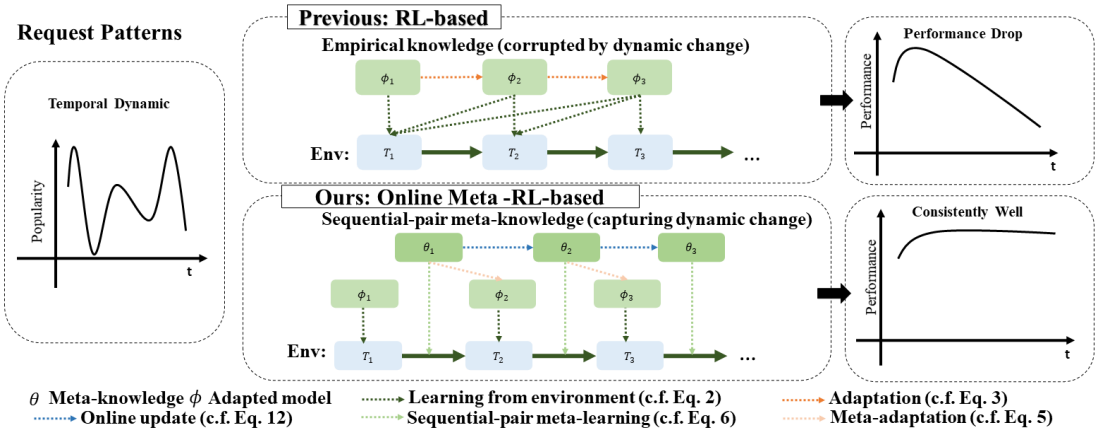

Fig. 1: A comparison of the previous RL-based framework with our online meta-RL-based framework when dealing with dynamic edge caching problems. **Left:** The request patterns in edge content delivery are temporally dynamic. **Middle and right top:** RL-based caching policies only capture empirical knowledge that is uniformly learned from historical data, and thus is gradually corrupted by the outdated data, leading to a performance drop. **Middle and right bottom:** Our online meta-RL-based method can extract the real-time sequential-pair meta-knowledge of the dynamic changes, which enables fast dynamic adaptation. The model is then adapted from real-time meta-knowledge and new data samples, and thus gets rid of the negative effect of the outdated data and adapts quickly to the changing edge environment, leading to a consistently good performance.

maintain the "freshness" of meta-knowledge.

Combining the two efforts above, we present an Online Meta Actor-Critic (OMAC) framework to solve the dynamic edge caching problem. To verify the efficacy of OMAC, we conduct experiments using real-world traces from IQIYI and KuaiShou platforms. The results show that our solution consistently performs better than baselines in all 56 randomly selected edge areas. Our ablation studies demonstrate that sequential-pair designing and online updating contribute to performance gains.

The contributions can be summarized as:

- We discover the dynamic property of edge request pattern, demonstrating that it leads to the performance drop of vanilla RL-based methods.
- We show that plain meta-learning fails to improve the performance of RL-based algorithms and develop a new framework called the *online sequential-pair meta-learning framework* capable of handling the dynamic edge caching: 1) the *sequential-pair* design provides a general knowledge for dynamic changes; 2) the *online* update helps keep the knowledge effective along the time.
- We achieve state-of-the-art performance in both short and traditional video platforms, which outperforms in the short video platform where the dynamic is more severe.

II. Related Work

A. Rule-based and RL-based Caching

Conventional caching policies cannot adapt to the dynamic edge environment since they make content replacements based on hand-designed rules that are not adaptive and not robust in edge caching situations [18]. Some recent rule-based methods [19]–[22] rely on strong prior assumptions on popularity distributions, and models with these assumptions are widely used in the video on demand systems, such as uniform distribution, Gaussian distribution, and Zipf distribution. However, the dynamic request patterns cannot be accurately modeled as a stochastic process [3], leading to limited generalization ability.

RL is a powerful framework for designing caching strategies to deal with caching problems without prior assumptions [23], [24]. It has been shown that RL-based methods achieve better hit rates than traditional strategies in practical scenarios [8]–[10], [25], [26]. Somuyiwa et al. proposed a policy-based method [10] to solve the content replacement problem under unknown data distribution. Luong et al. [25] applied value-based methods to use the value function to find the nearly optimal solution with the "ε greedy explore" in the adaptation process. Zhu and Cao et al. developed actor-critic strategies [9], [26] to leverage the on-policy strategy to optimize the agent parameter in little steps. However, all the above methods cannot quickly adapt to the edge environment because of dynamic request patterns. They use out-dated request histories to do the gradient decent and would conquer a sudden performance drop.

Other models [11]–[13] that are based on manually designed features to overcome dynamic changes have been developed. Guan et al. [11] built a featured collector of frequency and duration for further critical feature learning. Kirlin et al. [13] assumed causal knowledge of the channel quality, the content profile, and the user-access behavior to model the proactive caching problem. Zhong et al. [12] applied frequency information to face the dynamic environment. However, the hand-designed heuristic works are only for exceptional cases and not general to diverse situations of edge caching environments.

B. Meta-RL

Meta-learning [27], [28] provided a powerful framework to capture the uniform transferable knowledge for improving the model adaptation ability. Many works tried to enhance the effectiveness of the meta-framework in reinforcement learning. Duan et al. [29] proposed RL^2, which applied recurrent neural networks to serve as dynamic task embedding storage. Finn et al. proposed MAML [37], where a bilevel optimization learns the optimal initialization, accelerating the model adaptation when it meets a new task. Faktor et al. [30] proposed meta Q-learning for a more accurate function approximation. Yang et al. [31] focused on multi-task RL using soft modularization to adjust the temperature factor between each training task to better converge. However, the above works are designed to capture uniform transferable knowledge, thus not suitable for dynamic scenarios.

Al-Shedivat et al. [32] attempted to solve the continuous adaptation problem for dynamic environments and proposed to capture the dynamic changes. But, the meta-knowledge remains unchanged after meta-training, leading to the effectiveness decay. Online meta-learning [33], [35] merges ideas from both online learning and meta-learning to better capture the spirit and practice of continuous lifelong learning. However, they need to consider the sequential dependence of continuously changing environments.

In summary, no previous meta-RL design is suitable for learning under severe and continuous dynamic environments, e.g., caching environments, especially for the short video platform. Our OMAC policy addresses this by the design of online meta-learning from the sequential-pair objective.

III. Dynamic Property Observations

In this section, we introduce our discoveries: the dynamic property on request popularity (i.e., distribution of different contents) and quantity (i.e., the number of requests of all contents), which challenges the previous works that depend on stationary assumptions. We then show that such a dynamic edge request pattern would impair the vanilla RL-based method, driving the demand for a meta-RL framework.

A. Observations from Real Traces

In this trace-driven study, we examine two real-world datasets that contain user request traces collected from two of the largest Chinese video platforms, i.e., IQIYI, featuring long-form video content, and KuaiShou, featuring short-form video content. Each contains video-watching request traces over 13 days from randomly sampled 56 edge areas with a service range of $2.05km \times 2.31km$ in Beijing. Each valid user-item trace contains information, including timestamps, video content ID, and the location of each request. We focus on two perspectives in our trace-driven study, especially targeting the dynamic features of edge caching:

1) ***Temporal dynamic on request popularity****:* To visualize the request pattern, we first study the heterogeneity of the temporal dynamic of video requests from edges. We calculate the request distributions of each day and plot the

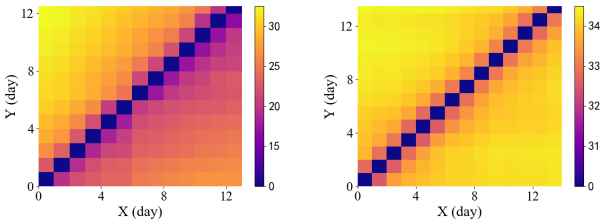

Fig. 2: The KL divergence of temporal content distribution among platform IQIYI (a) and KuaiShou (b).

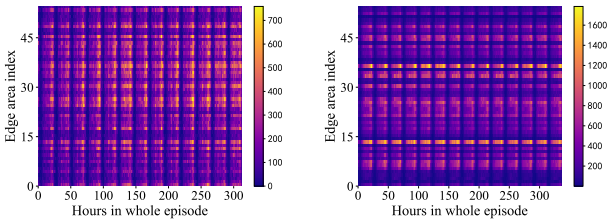

Fig. 3: Request frequency of each area during the whole time among platform IQIYI (a) and KuaiShou (b).

KL divergences between them in Fig. 2a and Fig. 2b. This measurement is conducted in 56 different areas to eschew the exceptional cases, and we present the average results. The fact that KL divergence values are generally higher than 15.0 indicates that the edge caching environment is highly dynamic over time. A significant KL divergence value indicates that the respective content popularities are significantly different and impossible for a static distribution to fit along the time. The dynamic is more severe on the short video platform, as shown in the KuaiShou dataset, where most KL divergence values exceed 31.47. This dynamic property requires cache strategies to have higher adaptability in the severely changing environment. In particular, the caching strategy has to adapt to the current environment using a few real-time samples since the historical data is outdated under edge caching situations. Furthermore, as shown in Fig. 2a and Fig. 2b, the popularity divergence is positively correlated to the time span, suggesting that the content popularity is continually changing instead of just verifying the non-stationarity.

2) ***Highly variable request frequency****:* We next observe the dynamic of request frequency. Fig. 3a and Fig. 3b visualize the request frequency of each area during the whole time, where the x axis represents the time slot, the y axis represents edge ID (56 selected areas), and the value represents the requested quantity during an hour. We observe that the request frequency is highly variable, e.g., the highest number is more than 1500, and the lowest one is below 200 in the KuaiShou dataset, indicating the change of edge request pattern. In Fig. 4a and Fig. 4b, the curves represent the request frequency of each content over time. We observe that the frequency change is also highly variable among different contents, indicating that they share no uniform pattern. This observed result suggests that the cache policy needs to conclude a general knowledge to deal with the dynamic of request pattern.

In summary, 1) and 2) show that the request pattern con-

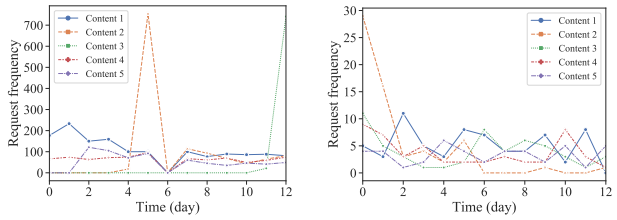

Fig. 4: Request frequency of contents among platform IQIYI (a) and KuaiShou (b).

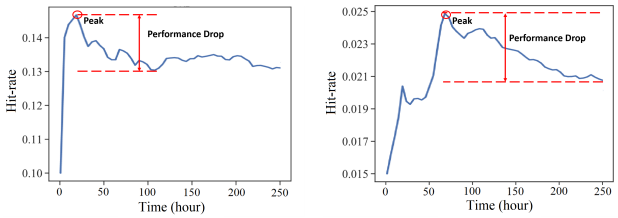

Fig. 5: The performance drop of vanilla RL-based caching strategy among platform IQIYI (a) and KuaiShou (b).

tinually changes in popularity and quantity, and each kind of content has its pattern. This demands caching strategies to become adaptive to deal with dynamic environmental changes and to become model-free to face different patterns.

B. Challenges to Previous RL-based Methods

Fig. 5a and Fig. 5b illustrate how the performance of the RL-based method [9] changes over time. There are two main observations here:

i) The performance increases in the beginning: To begin with, the hit rate gradually increases and attains the highest (the peak). Since the request popularity is continually changing, the environmental change is not severe and can still be considered relatively stable. Therefore, the RL model could perform to increase the hit rate. We notice that it takes a long time of 23 hours and 70 hours for the RL-based method to attain the peak in IQIYI and Kuaishou traces.

ii) The performance drops a lot in the long term: The hit rate decays obviously after the peak. Along with the request popularity changes over time significantly, the RL-based method would gradually fail after the peak because most traditional RL models have the assumption of a stationary and static environment [14], [15]. The reason is that in edge caching, the dynamic request pattern would continually impair the stationary assumption, i.e., the outdated historical data degrades the real-time performance of the RL model. Therefore, the RL-based method could not achieve high performance under the edge's dynamic environment, especially for today's short-term video-sharing platforms.

These two phenomena raise a question: *how to eschew the performance drop and maintain the highest performance during the whole episode?*

C. Motivation towards Meta-Learning

To counter the performance decline, a natural idea is to *segment* the dynamic episode into multiple short intervals. A re-initialized RL model may appropriately handle each interval. The initial motivation is that the continually changing distribution impairs vanilla RL-based methods' "long-time" performance (though it is continually updating with new data), as shown in Fig. 3. It is expected that, after running initialized RL model on each short interval, the model can escape from the historical data disturbance and reach a "short-time" good performance, as RL model can reach the "peak" in Fig. 5a and Fig. 5b. However, simply restarting the model could make it more difficult to converge, as it takes a relatively long time to collect enough samples (23 hours in IQIYI and 70 hours in Kuaishou for adaptation), and we observe that purely restarting the RL model degrades the performance in Table II.

Since it takes a long time to attain the peak performance, to shorten the adaptation time, we consider offering transferable knowledge (i.e., the *meta*-knowledge) to enable a faster adaption to the re-initialized RL model when dealing with a new time interval. We adopt meta-learning [37] to enhance the initialization. Considering the severe dynamic of the edge caching environment, we also refresh the transferable knowledge to better fit the changing environment's general features.

Therefore, we propose a new meta-RL-based caching solution for dynamic edge caching.

IV. PRELIMINARY

In this section, we present some preliminary concepts of RL-based caching and meta-learning. First, we introduce the system model for RL-based edge caching, where we give necessary notations and definitions. Second, we introduce the basic meta-learning captions.

A. RL-based Edge Caching

We study a system of one edge cache with a maximum storage capacity size S. We denote the index set of library for equal size content files by $\mathbf{F} = \{1, 2, \cdots, C\}$, and the request is made for $f \in \mathbf{F}$ in each time slot t. We further denote the numbers of requests to content f in time slot t as Num_f^t. A binary variable x_f^t is used to represent whether content f is requested ($x_f^t = 1$) or not ($x_f^t = 0$) at time slot t, as a flag variable. We also use a binary indicator y_f^t to represent whether video content f is cached ($y_f^t = 1$) or not ($y_f^t = 0$). When content f is requested at slot t and found in the cache $c^t = \{c_1, \cdots, c_S\}$, it is called a hit, otherwise a miss.

Our goal is to maximize the hit rate, i.e., minimizing the content transmission cost. The content transmission cost within a piece of time period of the edge e could be calculated as:

$$\min : L_e^t(transmit) = \sum_f Num_f^t \cdot x_f^t \cdot (1 - y_f^t)$$

With the restrictive conditions:

$$\sum_f y_f^t \leq S, \forall f \in F \tag{1}$$

Eq. 1 indicates that contents are cached in an edge server which has a finite memory.

In traditional RL, the agent tries to make proper decisions towards optimization goals based on past experience and given rewards. In particular, the agent observes the state s_t of the task environment T in each time slot t and takes an action a_t according to the policy π with parameter ϕ. Then the agent gets the reward r_t and transfers to state s_{t+1}. We then denote the trajectory as a sequence $\tau = (s_t, a_t, r_t, s_{t+1}, \cdots, s_W, a_W, r_W)$, and the agent samples K trajectories $\tau^{1:K}$ based on the previous Markov decision process, with a time window W. The agent then updates to a new policy. The goal of the learning process is to find an optimal policy π^* that can maximize the expected discounted cumulative reward $R = \sum_1^W \gamma^t r_t$, where γ is a discount factor that reduces the importance of future rewards. The optimization goal of the RL-based method is to:

$$\min_{\phi} l_T(\phi) = \mathbb{E}_{\tau_T, \phi \sim P_T(\tau|\phi)} \left[-R^{1:K} \mid \tau_{T,\phi}^{1:K}, \phi \right] \tag{2}$$

where $R^{1:K} = \sum_{i=1}^K R^i$ and R^i is the discounted cumulative reward for trajectories τ^i. Then, we use Eq. 3 to update the parameter ϕ.

$$\phi = \phi - \alpha \nabla_{\phi} l_T(\phi) \tag{3}$$

We next present the details for edge caching policy, including the state, the action, and the reward. The state of the agent includes the current content request and the caching situations ($s_t = \{x_1^t, \cdots, x_C^t, c^t\}$). After being fed the state, the agent gives the action $a_t \in [1, S+1]$ based on its policy π. If $a_t = S+1$, the agent will not do the content replacement. Otherwise, the caching content in position i will be replaced, when $a_t = i$. New arrived content IDs are also applicable to the adapatation process, since it concerned only the cached and incoming requested status instead of specific IDs [10]. After this action, the agent receives a reward $r_t = \sum_{f=1}^N (Num_f^t * x_f^t * y_f^t + \max\{y_f^t - y_f^{t-1}, 0\})$ and the state changes to s_{t+1}.

B. Meta-learning

In meta-learning settings, the goal is to enable an agent to quickly acquire an effective policy for a new coming task environment, using a small amount of experience or samples in adaptation iterations [37].

To achieve this, it learns a meta initialization θ from a set of tasks $\{T_1, T_2, \ldots, T_N\}$, such that at meta-test time, performing a few steps of gradient descent from θ using new trajectories minimizes the loss on each T_i. To get such an initialization when meta-training, it solves the following optimization problem:

$$\begin{aligned} \min_{\theta,\alpha} \sum_{i=1}^{N-1} L_{T_i}(\theta, \alpha) = \sum_{i=1}^{N-1} \mathbb{E}_{\substack{\tau_{T_i,\theta} \sim P_{T_i}(\tau|\theta) \\ \tau_{T_i,\phi_i} \sim P_{T_i}(\tau|\phi_i)}} \left[-R_{T_i}^{1:K} \mid \tau_{T_i,\phi_i}^{1:K}, \phi_i \right] \\ \text{where } \phi_i^M = h(\tau_{T_i,\theta}^{1:K}, \theta, \alpha), i = 1, \cdots, N-1 \end{aligned} \tag{4}$$

Where we denote the policy parameter in the *meta-learning* part (the outer loop, i.e., the optimized objective) as θ and the policy in the *meta-adaptation* part (the inner loop, i.e., the constrain) as ϕ. To obtain policy parameter ϕ adapted from θ, h denotes an adaptation rule that maps the initialization (i.e. meta-knowledge) to an adapted agent ϕ. Specifically, from initialization θ with adaptive stepsizes $\alpha = \{\alpha^1, \alpha^2, \cdots, \alpha^M\}$ on trajectories $\tau_{T_i,\theta}^{1:K}$, h is subsequently carried out by $M-$steps gradient decent:

$$\begin{aligned} \phi_i^m = \begin{cases} \theta, \tau_{T_i,\theta}^{1:K} \sim P_{T_i}(\tau \mid \theta), m = 0 \\ \phi_i^{m-1} + \alpha^m \nabla_{\phi_i^{m-1}} \left[R_{T_i}^{1:K} \mid \tau_{T_i,\phi_i^{m-1}}^{1:K}, \phi_i^{m-1} \right] \end{cases} \\ \text{where } M \geq m > 0 \end{aligned} \tag{5}$$

V. Online Sequential-pair Meta-RL Framework

In this section, we present our online sequential-pair meta-learning framework for edge caching. First, we restruct the meta-RL with the proposed sequential-pair meta loss for meta-knowledge of dynamic caching tasks. Second, we use an online manner to update the meta-knowledge, and maintain its long-term effects. Finally, we present our Online Meta Actor-Critic (OMAC) caching algorithm for practical usages.

A. Sequential-pair meta-RL design

We use a sequential-pair meta-RL design to reformulate the meta-RL framework, which is specially designed for continually changing environments. As described in Sec. III, we segment the dynamic episode of the edge environment into a series of intervals as the sub-tasks. The RL agent is reinitialized in each sub-tasks. For the overall goal is to realize a fast adaptation when the sub-task switches, we use meta-RL [37] to re-initialize the RL agent with a learned initialization, i.e., the meta-knowledge.

To measure the quality of this meta-knowledge, specifically, we apply a *sequential-pair (meta)* loss $L_{T_i,T_{i+1}}$ to evaluate the adaptation (from the i-th environment T_i to the consecutive environment T_{i+1}). Derived from Eq. 2 and Eq. 4, the objective function of our *sequential-pair* meta-RL is:

$$\begin{aligned} \min_{\theta,\alpha} \sum_{i=1}^{N-1} L_{T_i,T_{i+1}}(\theta, \alpha) = \sum_{i=1}^{N-1} \mathbb{E}_{\substack{\tau_{T_i,\theta} \sim P_{T_i}(\tau|\theta) \\ \tau_{T_{i+1},\phi_i} \sim P_{T_{i+1}}(\tau|\phi_i)}} \left[-R_{T_{i+1}}^{1:K} \mid \tau_{T_{i+1},\phi_i}^{1:K}, \phi_i \right] \\ \text{where } \phi_i = h(\tau_{T_i,\theta}^{1:K}, \theta, \alpha), i = 1, \cdots, N-1 \end{aligned} \tag{6}$$

The function h denotes the same as Eq. 5. We notice that the sequential-pair meta loss is not much different from the original one in Eq. 4, by changing the inner expectation on T_i to T_{i+1}. This is specialized to deal with dynamic environments. In more detail, it firstly samples trajectories $\tau_{T_i,\theta}$ from task T_i using policy π_θ parameterized by meta-knowledge θ. Through the adaptation process (adaptation rule h) from trajectories $\tau_{T_i,\theta}$ with initialization θ, we get the adapted policy $\pi_{\phi_i^M}$. The process then evaluates the adapted policy $\pi_{\phi_i^M}$ by sampling trajectories $\tau_{T_{i+1},\phi}$ from task T_{i+1} with adapted policy $\pi_{\phi_i^M}$. The objective function measures the adaptive performance from a sequential-pair view, where the initialized policy θ first learns from T_i and then evaluates from

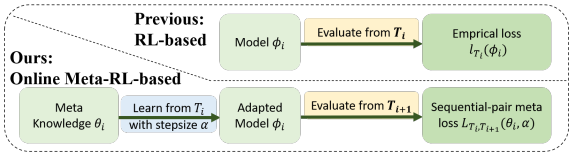

Fig. 6: An illustration of the sequential-pair meta loss. Compared with the loss used in previous RL-based frameworks, our sequential-pair meta loss measures the dynamic adaptation of the meta-knowledge θ_i by evaluating the performance of the adapted model ϕ_i on a consecutive task.

consecutive task T_{i+1}. It measures the dynamic transferable efficacy of meta-knowledge θ, and is suitable from dynamic edge caching.

Intuitively, this optimization aims to search for a good initialization θ and stepsizes α, such that the adapted policy π_ϕ (computed on the trajectories sampled from T_i) is good for solving T_{i+1}. This process imitates the dynamic adaptation from an old environment to a new one for the RL models. The sequential-pair loss measures the dynamic changes between two environments and how the adapted policy π_ϕ performs in the new environment to learn general dynamic information. Therefore, the RL agent initialized by the optimal meta-knowledge θ^* can fast adapt to the new environment through multiple steps adaptation with the optimal stepsizes α^*. The diagrammatic sketch of sequential-pair meta-RL is displayed in Fig. 6, where we also demonstrate its difference with empirical loss used in vanilla RL.

B. Online manner of meta-RL

Since the RL model collects the request trajectories from the edge cache in an online manner, the global optimal θ, α (in Eq. 6) are impossible to be determined since the agent does not know the future environments. Specifically, when the agent makes the decision in the environment indexed by i, it is impossible to know the future environments indexed by $i' > i$.

Thus, we propose to approximate the optimal in an *online manner* by minimizing the regret:

$$\text{Regret} = \sum_{i=1}^{N-1} \mathbb{E}[L_{T_i,T_{i+1}}(\theta_i, \alpha_i)] - \min_{\theta} \sum_{i=1}^{N-1} \mathbb{E}[L_{T_i,T_{i+1}}(\theta, \alpha)] \tag{7}$$

where θ_i is the meta-knowledge of task i, and the regret is a matrix for online decisions.

The validity of our approximation can be guaranteed to be valid when a sublinear term of N bounds the regret, which implies near optimal online solutions [38]. This bound is achieved by Eq. 12, which we will discuss later in Sec. V-C2. By those operations, the meta-knowledge θ could stay fresh and effective along with the task T changes. Different from traditional meta-RL, the meta-knowledge keeps updating and maintains the good re-initialization performance of the agent.

C. The OMAC Caching Algorithm

In this subsection, we instantiate our online sequential-pair meta-RL framework by using Actor-Critic. We present the sequential-pair meta-RL loss with Actor-Critic in detail. Then we give an overview of our OMAC (Algorithm 1).

1) Instantiation of OMAC: Since many previous *value-based* RL methods maintain a reply buffer [24] that may suffer from high memory cost, occupying too much cache memory. Therefore, we instantiate our Meta-RL as *Meta actor-critic*, alleviating the potential memory issue in edge caching. Additionally, a soft-policy entropy bonus is introduced, inspired by [28], together with the temperature factor μ_t, to encourage the exploration.

The agent with parameter θ is updated by optimizing the loss function with three parts: actor, critic, and entropy. Thus, the total loss function is:

$$L^{total}_{T_i,T_{i+1}}(\theta,\alpha) = L^{actor}_{T_i,T_{i+1}}(\theta,\alpha) + \mu_t \mathcal{H}_{T_i,T_{i+1}}(\theta,\alpha) + L^{critic}_{T_i,T_{i+1}}(\theta) \tag{8}$$

Notice that the *sequential-pair* design is utilized to help the agent adapt better with the continued changing request pattern, which is further assessed in later Sec. VI-C. The specific actor loss function (for training the Actor-network π) can be described as:

$$\begin{aligned} L^{actor}_{T_i,T_{i+1}}(\theta,\alpha) = \mathbb{E}_{\substack{\tau^{1:K}_{T_i,\theta}\sim P_{T_i}(\tau|\theta) \\ \tau^{1}_{T_{i+1},\phi}\sim P_{T_{i+1}}(\tau|\phi)}} & \frac{\hat{A}^{GAE}\cdot \pi_\phi\left(\tau_{T_{i+1},\phi}\right)\prod_{k=1}^{K}\pi_\theta\left(\tau^k_{T_i,\theta}\right)}{\pi^{old}_\phi\left(\tau_{T_{i+1},\phi}\right)\prod_{k=1}^{K}\pi^{old}_\theta\left(\tau^k_{T_i,\theta}\right)} \\ R_t = \hat{A}^{GAE}_t &= (r_t + \gamma V_{t+1} - V_t) + \gamma\lambda A_{t+1} \end{aligned} \tag{9}$$

$$\mathcal{H}_{T_i,T_{i+1}}(\theta,\alpha) = \mathbb{E}_{\substack{\tau^{1:K}_{T_i,\theta}\sim P_{T_i}(\tau|\theta) \\ \tau^{1}_{T_{i+1},\phi}\sim P_{T_{i+1}}(\tau|\phi)}} \left[\mathcal{H}\left(\pi_{\phi^M_i}\right) \mid \tau^{1:K}_{T_i,\theta}, \theta\right] \tag{10}$$

Here, the A^{GAE} with multi-step estimation is used to reduce the gradient variance [23], reaching more precise estimation results in dynamic edge scenarios.

Similarly to the actor loss, we also construct a cross sequential-pair critic loss (for training the Critic network V) aiming to better cache proper content in a dynamic scenario (Temporal Difference update is utilized):

$$L^{critic}_{T_i,T_{i+1}}(\theta) = \mathbb{E}_{\substack{\tau^{1:K}_{T_i,\theta}\sim P_{T_i}(\tau|\theta) \\ \tau^{1}_{T_{i+1},\phi}\sim P_{T_{i+1}}(\tau|\phi)}} \begin{bmatrix} \|V_{t,\phi} - \gamma V_{t+1,\phi} - r_{t,\phi}\|^2_2 \\ + \|V_{t,\theta} - \gamma V_{t+1,\theta} - r_{t,\theta}\|^2_2 \end{bmatrix} \tag{11}$$

2) OMAC Pipeline: Based on our online sequential-pair meta framework, we present our practical OMAC algorithm (Alg. 1) which consists of two stages: *Meta Pretraining Process* and *Online Meta Adaptation*.

i) Meta Pretraining Process (obtaining the meta-knowledge with sequential-pair tasks): In the meta pretraining part of our framework, we use a small number of historical request sequences to jointly update the three parameters θ, μ_t, and α, to get the initial meta-knowledge. We use π_ϕ adapted from π_θ of T_i to collect the trajectories of T_{i+1}. The *main intuition* is that the trajectories of T_i may contain some transferable information suitable for T_{i+1}, since the sequential-pair tasks are dependent to each other.

ii) Online Meta Adaptation (refreshing the meta knowledge): OMAC updates the meta-knowledge by multiple-step

Online Gradient Descent (OGD) [38], i.e., for each day $N+i$. OMAC iterates the meta-knowledge by:

$$\theta^1_{\mathcal{N}+i+1} = \theta^*_{\mathcal{N}+i},$$
$$\theta^k_{\mathcal{N}+i+1} = \theta^{k-1}_{\mathcal{N}+i+1} + \beta \nabla_{\theta_{\mathcal{N}+i+1}} L^{\text{total}}_{T_{\mathcal{N}+i}, T_{\mathcal{N}+i+1}}(\theta^{k-1}_{\mathcal{N}+i+1}, \alpha), \tag{12}$$

where $k = 1, \cdots, M$, and the meta-knowledge for next round is $\theta^*_{\mathcal{N}+i+1} = \theta^M_{\mathcal{N}+i+1}$.

Intuitively, after the meta pretraining process providing an initial meta-knowledge of $\mathcal{N}$ days of tasks, the online meta adaptation part of our framework searches for fresh meta-knowledge $\theta^*_{\mathcal{N}+i}$ to reach a better-adapted policy within only a few samples from $T_{\mathcal{N}+i+1}$. Theoretically, this OGD-alike [38] iteration is guaranteed to achieve a sublinear regret, which ensures that the online meta-knowledge is nearly optimal [33]. Note that there exist works on caching with online learning [34]–[36], [39], however is not RL-based. The meta-learning phase aims to find some universal features as meta-knowledge and obtains a transferable meta-knowledge $\theta^*_{\mathcal{N}+i}$ to help the agent acquire a better adaptation rule.

D. Discussion

*1) **The advantage of sequential-pair reformulation**:* Our sequential-pair meta-RL samples trajectories from environment T_i and T_{i+1} in a sequential pair way, aiming to capture dynamic changes that help the dynamic adaptation. The equations in Sec. V-A display their focus on the temporal feature in the edge caching scenario. Note that the simple meta-RL is design for multi-task learning, and ignores the information from the time sequence.

*2) **The advantage of online update manner**:* When it comes to the meta-adaptation stage, the offline version is simply initialized by the meta pretraining parameters at the beginning. Then, the model maintains the same meta-knowledge during the whole process. However, since the content popularity is continuously changing, the meta-knowledge should also track this dynamic environment. Thus, the traditional meta-RL methods may not guarantee the freshness for meta-knowledge owing to the lack of online updating. Besides, without the online update, it will lead to *an attenuation of the generality* acquired by meta-pretraining, which implies a negative effect when searching for the adaptation rule.

Oppositely, our online manner effectively uses the latest caching information in the adaptation stage for fresh meta-knowledge. Thus, both the past experience and current information will be enriched to the OMAC caching policy.

VI. Performance Evaluation

In this section, we conduct comprehensive experiments to evaluate the performance of the OMAC edge caching policy on real-world traces from IQIYI and KuaiShou platforms. First, we introduce the datasets, baselines, and implementation details. Second, we present the experiment results and show the superiority of the OMAC on both dynamic adaptability and robustness. Finally, our ablation study shows that both the design of sequential-pair and the online manner for meta-RL updates are indispensable to performance gains.

Algorithm 1 Online Meta Actor-Critic (OMAC) Caching

Input: Number of Meta Pretraining days $\mathcal{N}$, a dynamic sequence of tasks $T_1, T_2, \cdots, T_{\mathcal{N}}, \cdots T_{\mathcal{N}+i}, T_{\mathcal{N}+i+1}$.
Stage 1: Meta Pretraining Process
Randomly initialize θ_1 and α_1
repeat
  Gain all sequential-pairs $(T_i, T_{i+1})_{i=1:\mathcal{N}-1}$ from the sequence.
  for all Sequential-pair data **do**
    Sample traj.$\tau^{1:K}_{T_i,\theta_i}$ using π_{θ_i}
    Compute $\phi^M_i = h(\tau^{1:K}_{T_i,\theta_i}, \theta_i, \alpha_i)$ using Eq. 5
    Sample traj.$\tau^{1}_{T^{1:K}_{i+1},\phi^M_i}$ using $\pi_{\phi^M_i}$
  end for
  Compute $\nabla_{\theta_i} L^{\text{total}}_{T_i,T_{i+1}}(\theta, \alpha)$ and $\nabla_{\alpha_i} L_{T_i,T_{i+1}}(\theta, \alpha)$ using Eq. 8.
  Update $\theta_{i+1} = \theta_i + \beta \nabla_\theta L(\theta, \alpha)$, $\alpha_{i+1} = \alpha_i + \beta \nabla_\alpha L(\theta, \alpha)$
until Convergence to obtain a meta-knowledge θ^*_N and α^*_N
Stage 2: Online Meta Adaptation:
while New task $T_{\mathcal{N}+i+1}$ from the dynamic tasks' sequence **do**
  Initialize $\Phi_i = \theta^*_{\mathcal{N}+i}$ and $\alpha_i = \alpha^*_{\mathcal{N}+i}$
  Deal with $T_{\mathcal{N}+i+1}$ using policy π_{Φ_i}
  Sample traj.$\tau^{1:K}_{T_{\mathcal{N}+i+1},\Phi_i}$ while solving $T_{\mathcal{N}+i+1}$
  Update $\Phi_{i+1} = h(\tau^{1:K}_{T_{\mathcal{N}+i+1},\Phi_i}, \Phi_i, \alpha^*_{\mathcal{N}+i})$ using PPO with *Importance Sampling*.
  for $(T_{\mathcal{N}+i}, T_{\mathcal{N}+i+1})$ sequential-pair **do**
    Sample traj.$\tau^{1:K}_{T_{\mathcal{N}+i},\Phi i+1}$ using $\pi_{\Phi_{i+1}}$
    Compute $\phi = h(\tau^{1:K}_{T_{\mathcal{N}+i},\Phi_{i+1}}, \Phi_{i+1}, \alpha^*_{\mathcal{N}+i})$ using Eq. 5
    Sample traj.$\tau^{1:K}_{T_{\mathcal{N}+i},\phi}$ using π_ϕ
    Compute $\nabla_{\theta_{\mathcal{N}+i},\alpha_{\mathcal{N}+i}} L^{\text{total}}_{T_{\mathcal{N}+i},T_{\mathcal{N}+i+1}}(\theta, \alpha)$ using Eq. 8.
    Online update $\theta_{\mathcal{N}+i+1} = \theta^*_{\mathcal{N}+i} + \beta \nabla_\theta L(\theta, \alpha)$ and $\alpha_{\mathcal{N}+i+1} = \alpha^*_{\mathcal{N}+i} + \beta \nabla_\alpha L(\theta, \alpha)$ for M times to obtain fresh meta-knowledge $\theta^*_{\mathcal{N}+i+1}$ and $\alpha^*_{\mathcal{N}+i+1}$
  end for
end while

A. Experiment Setup

Datasets: We collect content request records from platforms IQIYI and KuaiShou in 13 days and extract the timestamps and locations of each request. The IQIYI traces contains 53,954,230 requests of 417,077 contents and the KuaiShou traces contains 52,852,160 requests of 1,746,227 contents. Since the trace analysis shows that the request pattern is different among areas, we randomly select 56 representative edge areas ($2.05km \times 2.31km$) as our edge regions to verify the robustness. In addition, we vary the content delivery range from $0.59km^2$ to $303.07km^2$ of an edge cache device to evaluate the impact of the service range. In our experiments, we assume that a caching agent located at the center of the edge area will serve requests in this area.

Baseline methods: 1) *Least Recently Used (LRU)*: the edge cache device caches the most recently used content. 2) *Least Frequently Used (LFU)*: LFU counts the frequency of content requests and evicts the least frequently used ones. 3) *Deep reinforcement learning (DRL)* [9]: an RL-based policy with A2C architecture makes a decision to only replace one cached content with the requested content or not. 4) *Reinforcement learning with dynamic features (Adaptive-RL)* [12]: an RL-based caching policy uses manual features, i.e., the frequency of each content among last 10, 100, 1000 requests. We compare with LRU and LFU to demonstrate the advantage of the RL-based methods than traditional approaches, with DRL to demonstrate the efficacy of the designed online meta mechanism for RL-based methods, and with Adaptive-RL to show that the online meta-RL framework is more effective than manually designed time-series features.

Implementation details: The default cache capacity for every edge is set as 0.02% of all requested video contents. We implement the OMAC learning model using TensorFlow, which runs on a server with dual RTX 2070 GPU cards and 6GB memory. The learning rate for meta-learning and meta updates are set as 1e-3 and 2e-4. We set the discount factor γ as 0.99 by default. We consider the caching states of the past 7 time steps as the retained number for LSTM. Besides, we set the default meta pretraining days as 5, do MAML for 3 times per adaptation round and update the meta-knowledge every day.

B. Evaluation Results

Verification of dynamic adaptability: We first verify the dynamic adaptability of the OMAC strategy and display the average hit rate (y axis) of the selected area within the whole episode (x axis represents 13 days in total). The cache size used for experiments is set as 0.02% of all requested video contents. As illustrated in Fig. 7, we make the following main observations:

1) *OMAC has great dynamic adaptability.* OMAC works consistently well during the whole episode, while other baselines suffer from severe performance drop. Specifically, the curves of baselines have the trend to decrease over time after the peak, while OMAC maintains a high hit rate during the whole episode. Though LFU and adaptive-RL perform better in the initial stage, OMAC later keeps a hit rate of 19.3% in IQIYI and 2.72% in KuaiShou, while making an improvement over other baselines for 16.3% and 37.4%, respectively. This demonstrates that OMAC can continuously adapt to the changing environment and keep a good performance. The reason is that the initial environment can be viewed as relatively stable when the time interval is short. Other baselines depending on prior knowledge or stationarity may perform well at the beginning. However, when the time interval becomes larger, the inconsistency between the assumption and environment is more significant, leading to a severe performance drop. In contrast, in OMAC, the restart mechanism discards the outdated historical data, thus gets rid of the assumption-practice inconsistency. Besides, the designed sequential-pair meta objective captures the meta-knowledge of the environmental change, which helps the re-initialized model to adapt quickly. The online update of the meta-knowledge maintains its freshness and continually provides real-time adaptation rules. Therefore, OMAC does not suffer from the performance drop caused by the change of contents popularity.

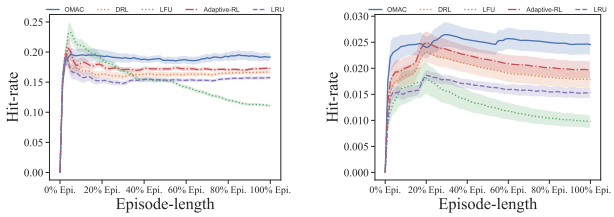

Fig. 7: Average hit rate versus episode time among platform IQIYI (a) and KuaiShou (b).

TABLE I: The average hit rate rank of all selected areas, with 90% confidence intervals for the mean over 56 areas.

	OMAC	DRL	LFU	Adaptive-RL	LRU
IQIYI	1.0 ± 0.0	2.9 ± 0.8	5.0 ± 0.0	2.3 ± 1.7	3.8 ± 0.9
KuaiShou	1.0 ± 0.0	3.2 ± 0.9	5.0 ± 0.0	2.1 ± 0.8	3.7 ± 1.6

2) *OMAC shows more advantages when dealing with more severe dynamic environments.* The advantage of OMAC is more significant on the short video platform, which shows a more significant temporal dynamic as discussed in Sec. III. Notably, the gap between the curves of OMAC and other baselines in KuaiShou (Fig. 7b) is obviously larger than in IQIYI (Fig. 7a), which demonstrates more superiority of OMAC in such more severe dynamic scenarios. One possible reason is that previous methods are based on either manual rules (LRU, LFU), manual features (Adaptive-RL), or RL algorithm for static environments (DRL), which are not so adaptive to the dynamic caching environment. Therefore, the performances of baselines drop in dynamic scenarios and get worse when the dynamic becomes more severe.

Verification of robustness: As shown in our trace analysis (Sec. III), the edge request pattern shows high spatial diversity, which implies that these regions have different patterns. From Fig. 8 of the average hit rate of 56 areas, the OMAC achieves a hit rate improvement of 16.3% in IQIYI and 37.4% in KuaiShou, demonstrating the effectiveness of our designs, especially for the short video platform. As shown in the Tab. I, the OMAC maintains the best performance among *all* 56 *areas* in IQIYI and KuaiShou, showing that the outperformance is robust to different request patterns. Adaptive-RL, LRU, and LFU are designed with hand-designed heuristics, which are not robust to diverse scenarios. DRL policy does not consider the dynamic changes, and therefore suffers from the performance drop for all dynamic situations.

Verification of generality for different cache sizes: We evaluate the performance of the proposed method under the impact of different edge servers' cache sizes. We conduct the experiment on five video content cache sizes, ranging from 0.02% to 0.10% with 0.02% size increase, respectively. As illustrated in Fig. 9, we observe that OMAC outperforms other baselines in all cache sizes. Our OMAC gains for a

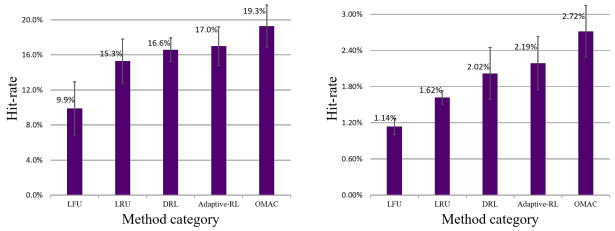

Fig. 8: The average hit rate of OMAC and the baselines, with 90% confidence intervals for the mean over 56 areas among platform IQIYI (a) and KuaiShou (b).

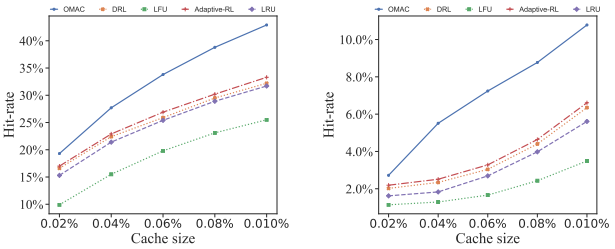

Fig. 9: Average hit rate versus the edge cache size among platform IQIYI (a) and KuaiShou (b).

16.3% to 33.2% improvement in IQIYI and 37.4% to 138.2% improvement in Kuaishout than DRL method, on five different cache sizes. The possible reason is that the OMAC learns a better ranking of contents than other baselines, and therefore the performance improvement is more significant when the cache size becomes larger.

Verification of generality for different edge area sizes: To evaluate the performance of the proposed method under the impact of edge service range, we experiment on 10 edge areas, ranging from $303.07km^2$ to $0.59km^2$ with 1/2 area decay respectively. As illustrated in Fig. 10, we observe that OMAC consistently outperforms other baselines in all service ranges, which shows that OMAC is more suitable for different edge service sizes.

In summary, OMAC shows a superior adaptation for dynamic environments, and performs consistently well in different areas with various locations and sizes, supporting the general effectiveness of the online meta-learning framework.

C. Ablation Tests of OMAC

To assess the efficacy of the two main components, we remove the meta-learning and online components of the framework in turn and look at the changes in hit rate. The results are reported in Tab. II, demonstrating that our proposed OMAC framework achieves evident performance improvement with all following components.

The effectiveness of sequential-pair meta-learning: Comparing with the vanilla DRL, DRL with our online sequential-pair meta-learning achieves a hit rate increase of 2.3% (13.9% improvement) in IQIYI and 0.49% (24.3% improvement) in KuaiShou. Purely restarting DRL model degrades the performance, because DRL model requires relative long exploration

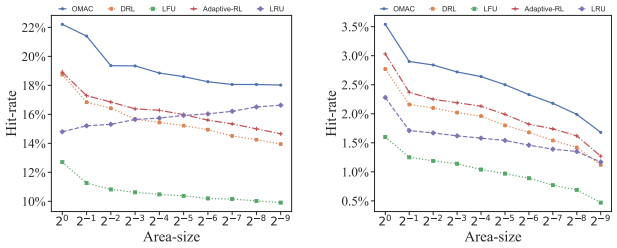

Fig. 10: Average hit rate versus the edge service range among platform IQIYI (a) and KuaiShou (b).

TABLE II: Ablation experiment of different modules, hit rates are listed (higher is better):

Vanilla DRL	+ Meta	+ Online	IQIYI	KuaiShou
●	×	×	16.6%	2.02%
●(restart)	×	×	15.2%(↓1.4%)	1.62%(↓0.40%)
●	● (traditional)	×	16.1%(↓0.5%)	1.79%(↓0.23%)
●	● (sequential-pair)	×	18.9%(↑2.3%)	2.51%(↑0.49%)
●	●	●	19.8%(↑3.2%)	2.72%(↑0.70%)

to adapt to a new environment without prior knowledge. The traditional meta-learning, which is used for multi-task learning, does not fit the dynamic sequence of the temporal domain and performs even worse than vanilla DRL (a hit rate of 0.5% and 0.23% worse than DRL in IQIYI and Kuaishou). This demonstrates that our *sequential-pair meta-learning* is suitable for the edge caching problem and could significantly boost the performance when facing continuous dynamic request patterns. The reason is that the sequential-pair meta-learning can capture the dynamic meta feature, which enables fast adaptation to the dynamic environment. While meta-learning only captures the shared meta-knowledge among all tasks and ignores the time-sequential order, it thus fails to enhance the RL performance.

The effectiveness of online components: Comparing with offline meta-learning, DRL with online meta-learning achieves a hit rate increase of 0.9% (5.4% improvement) in IQIYI and 0.21% (10.3% improvement) in KuaiShou. This shows the importance of online updating the meta-knowledge under the continuous dynamic since the previous meta-knowledge would gradually lose effectiveness over time, and it is reasonable to update the meta-knowledge to keep it fresh regularly.

VII. Conclusion

In the paper, we study an open and difficult problem: edge caching under continuously changing video popularity. We first demonstrate such dynamic request pattern leads to the performance drop of RL-based caching approaches. To get rid of this performance drop, we then develop an online sequential-pair meta-RL framework that is used for fast adaptation in dynamic environments. The key design consists of sequential-pair meta-knowledge that captures the dynamic changes, and an online manner that keeps this knowledge fresh. The experimental results support our design by the significant outperformance compared to baselines, and demonstrate the efficacy of each component.

Our results take the first step towards meta-learning to solve the dynamic caching problem. Our framework can be easily integrated into various scenarios with temporal dynamic, e.g., multi-agent caching, cooperative caching.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 1, no. 99, pp. 1–1, 2017.

[2] W. A. Aziz, H. K. Qureshi, A. Iqbal, and M. Lestas, "Accurate prediction of streaming video traffic in tcp/ip networks using dpi and deep learning," in *2020 International Wireless Communications and Mobile Computing*, pp. 310–315, 2020.

[3] C. Li, L. Toni, J. Zou, H. Xiong and P. Frossard, "QoE-Driven Mobile Edge Caching Placement for Adaptive Video Streaming," in *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965-984, April 2018.

[4] Y. Zhou, L. Chen, C. Yang and D. M. Chiu, "Video Popularity Dynamics and Its Implication for Replication," in *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1273-1285, Aug. 2015.

[5] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, 2003.

[6] Q. Huang, K. Birman, R. Van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of facebook photo caching," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pp. 167–181, 2013.

[7] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Computer Architecture Letters*, vol. 50, no. 12, pp. 1352–1361, 2001.

[8] K. Guo, C. Yang, and T. Liu, "Caching in base station with recommendation via q-learning," in *2017 IEEE Wireless Communications and Networking Conference*, pp. 1–6, 2017.

[9] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, "Caching transient data for internet of things: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2074–2083, 2018.

[10] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1331–1344, 2018.

[11] Y. Guan, X. Zhang, and Z. Guo, "Caca: Learning-based content-aware cache admission for video content in edge caching," in *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 456–464, 2019.

[12] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *2018 52nd Annual Conference on Information Sciences and Systems*, pp. 1–6, 2018.

[13] V. Kirilin, A. Sundarrajan, S. Gorinsky, and R. K. Sitaraman, "Rl-cache: Learning-based cache admission for content delivery," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2372–2385, 2020.

[14] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *2020 IEEE Conference on Computer Communications*, pp. 2499–2508, 2020.

[15] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile d2d networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1610–1622, 2019.

[16] E. Z. Liu, A. Raghunathan, P. Liang, and C. Finn, "Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices," in *International Conference on Machine Learning*, vol. 139, pp. 1856–1865, 2021.

[17] J.-C. Shi, Y. Yu, Q. Da, S.-Y. Chen, and A.-X. Zeng, "Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning," in *2019 AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4902–4909, 2019.

[18] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *2015 IEEE Conference on Computer Communications*, pp. 756–764, 2015.

[19] H. Zhao, Q. Zheng, W. Zhang, B. Du and H. Li, "A Segment-Based Storage and Transcoding Trade-off Strategy for Multi-version VoD Systems in the Cloud," in *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 149-159, Jan. 2017.

[20] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3212–3229, 2016.

[21] T. X. Tran and D. Pompili, "Octopus: A cooperative hierarchical caching strategy for cloud radio access networks," in *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 154–162, 2016.

[22] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *Special Interest Group on Data Communication Computer Communication Review*, vol. 43, no. 5, pp. 5–12, Nov. 2013.

[23] L. Engstrom, A. Ilyas, S. Santurkar, *et al.*, "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO," in *International Conference on Learning Representations*, pp. 1–14, 2020.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, pp. 1–14, 2016.

[25] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[26] 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.

[27] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," in *International Conference on Learning Representations*, pp. 1–20, 2019.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *International Conference on Machine Learning*, vol. 80, pp. 1856–1865, 2018.

[29] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," in *International Conference on Learning Representations*, pp. 1–14, 2017.

[30] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-Q-learning," in *International Conference on Learning Representations*, pp. 1–17, 2019.

[31] R. Yang, H. Xu, Y. Wu, and X. Wang, "Multi-task reinforcement learning with soft modularization," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 4767–4777, 2018.

[32] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," in *International Conference on Learning Representations*, pp. 1–21, 2018.

[33] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *International Conference on Machine Learning*, pp. 1920–1930, 2019.

[34] S. Zhou et al., "Caching in Dynamic Environments: a Near-optimal Online Learning Approach," in *IEEE Transactions on Multimedia*, early access, doi: 10.1109/TMM.2021.3132156.

[35] S. Li, J. Xu, M. van der Schaar and W. Li, "Trend-Aware Video Caching Through Online Learning," in *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503-2516, Dec. 2016.

[36] Z. Zhang, C. H. Lung, M. St-Hilaire and I. Lambadaris, "An SDN-Based Caching Decision Policy for Video Caching in Information-Centric Networking," in *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 1069-1083, April 2020.

[37] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, pp. 1126–1135, 2017.

[38] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[39] C. Tekin and M. van der Schaar, "Contextual Online Learning for Multimedia Content Aggregation," in *IEEE Transactions on Multimedia*, vol. 17, no. 4, pp. 549-561, April 2015.