Model-Based and Model-Free Learning-Based Caching for Dynamic Content

Bahman Abolhassani* Atilla Eryilmaz† Y. Thomas Hou*

* Virginia Tech, Blacksburg, VA, USA
† The Ohio State University, Columbus, OH, USA

MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM) | 979-8-3503-7423-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/MILCOM61039.2024.10773884

***Abstract*—We propose a "fresh" learning-based caching framework for content distribution networks (CDNs) with distributed front-end local caches and a dynamic back-end database. Users prefer the latest versions of dynamically updated content, while local caches lack knowledge of item popularity and refresh rates. We first explore scenarios with Poisson arrivals at the local cache and characterize the optimal policy's structure. Building on this, we introduce a model-based learning algorithm for caching dynamic content, demonstrating near-optimal costs and strong performance with limited cache sizes in simulations. For more general environments, we present a model-free Reinforcement Learning (RL) caching policy without prior statistical assumptions. Although model-free RL caching outperforms the model-based approach in high-variance arrival scenarios, it requires significantly longer training times due to its exploration phase. Model-based learning's quick adaptability to environmental changes and fast convergence rate make it a desirable approach for dynamic network environments, offering efficient fresh caching solutions for CDNs.**

***Index Terms*—Content Distribution Networks, Caching, Dynamic Content, Learning**

I. Introduction

Content distribution networks (CDNs) are essential for delivering content efficiently, improving performance, and reducing network congestion [1]. In the era of 5G, caching at the wireless edge accelerates content downloads and enhances network performance. However, new services like video on demand, augmented reality, and online gaming generate constantly changing data, making traditional caching methods less efficient [2]. The importance of content freshness has grown, as users prefer the latest data, especially in real-time applications. This necessitates the implementation of effective caching strategies to maintain the freshness of cached content [3].

Numerous works study the dynamic content delivery in caching systems such as [4]–[9] and effective strategies have been proposed. Despite the traditional caching paradigms, for content with varying generation dynamics, regular updates of cached content are crucial [10]–[13], preventing caches from becoming outdated due to aging content. The widespread deployment of edge caches necessitates a decentralized approach, with each cache independently managing its content freshness to avoid becoming outdated [14], [15]. These edge caches face the challenge of balancing update costs to ensure cost-effective operations. However, they often lack access to vital network parameters, requiring them to learn and adapt strategies for efficient cache management. Rapid changes in system parameters further demand continuous observation and dynamic adjustment of strategies based on user behavior to maintain overall cost-effectiveness [16], [17].

Machine Learning (ML) techniques have been widely utilized in network environments to achieve optimal caching performance [18]–[20]. Notably, Reinforcement Learning (RL) proves to be a viable option for edge caches, allowing them to learn user behavior through observation and interaction [20]–[22]. The rate at which users generate requests can significantly impact the time required for the RL policy to gather sufficient data and accurately learn the environment. Additionally, when the environment undergoes changes, such as shifts in popularity or refresh rates, retraining a new policy becomes necessary as the older policy may no longer be suitable for the new environment. This can demand substantial computational resources at the edge caches that exceed their capacity [23]. These challenges highlight the need for adaptive and efficient learning algorithms that can quickly adjust to changing conditions and ensure high caching performance in dynamic network environments.

In this study, we address the challenge of caching dynamic content efficiently by developing both model-based and model-free policies to manage fetching and freshness costs. We start by investigating scenarios with Poisson arrivals and characterize the structure of the optimal policy. Using insights from this optimal policy, we create a model-based algorithm that minimizes the need for resource-intensive training. To handle more diverse arrival processes, we introduce a model-free RL policy that adapts to any scenario without prior knowledge of request arrival distribution but requires longer training times compared to the model-based approach.

Our simulation results demonstrate that the model-based approach achieves near-optimal costs for unconstrained caches and significantly reduces training time compared to RL policies by eliminating the need for an exploration phase. By utilizing moving average filters, the model-based method adapts quickly to environmental changes, offering a practical and efficient alternative to model-free RL. This

This work was supported in part by ONR MURI Grant N00014-19-1-2621, NSF AI Institute (AI-EDGE) 2112471, CNS-2106679, CNS-2007231, ARO Grant W911NF-24-1-0103, Virginia Commonwealth Cyber Initiative (CCI), and Virginia Tech Institute for Critical Technology and Applied Science (ICTAS).

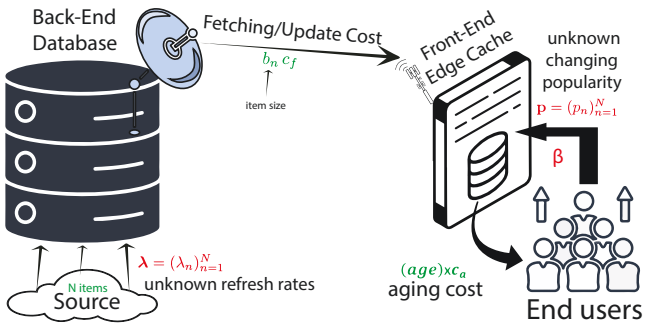

Fig. 1: Setting of *Fresh Caching for Dynamic Content*

study underscores the advantages of understanding the solution structure, enabling faster and more efficient caching strategies that match RL performance while requiring far less time and computational resources for training.

Our contributions, along with the organization of the paper, are as follows.

- In Section II, we introduce a practical caching model focused on efficiently serving dynamic content from a back-end source. The model addresses the challenge of front-end edge caches with limited knowledge of item popularity and refresh rates, aiming to minimize long-term average costs.
- In Section III, we explore a scenario where requests follow Poisson processes. With knowledge of popularity profiles, refresh rates, and demand intensity, we identify the optimal, threshold-based caching policy and provide a benchmark for evaluating our proposed algorithms.
- In Section IV, we present a model-based learning algorithm for dynamic content caching that achieves near-optimal costs without prior knowledge of item popularity and refresh rates, demonstrating its effectiveness in simulations with constrained cache space.
- In Section V, we extend our analysis to non-exponential interarrival times, proposing a model-free RL approach. Our simulations show that the RL policy achieves near-optimal costs, particularly as content becomes more dynamic.
- In Section VI, we compare the model-based learning and RL policies, noting that the RL policy slightly outperforms the model-based approach under high interarrival time variance but requires retraining when the environment changes, whereas the model-based policy adapts automatically. Finally, we conclude the work in Section VII.

II. System Model

We consider a hierarchical caching system, as illustrated in Fig. 1, where a user population is served a set $\mathcal{N}$ of N data items with content that dynamically changes over time. Next, we provide a detailed explanation of the its underlying dynamics.

Demand Dynamics: The local cache within the system experiences incoming requests at a rate $\beta \geq 0$ requests per second, which represents the intensity of requests from the user population. We also assume that the vector $(b_n)_{n=1}^N$ represents the size of the data items. For the model-based scenario, we assume that requests follow a Poisson process[1]; however, the popularity of individual items remains unknown. On the other hand, for the model-free scenario, we make the assumption that both the request arrival distribution and the popularity distribution are unknown.

Generation Dynamics: We consider the possibility of updates for each data item within the database, where the previous content of an item is replaced. We assume that data item n undergoes updates at an average rate of $\lambda_n \geq 0$ updates per second. It is worth noting that when $\lambda_n = 0$, it represents the traditional case of *static* content that remains unchanged. To conveniently represent the collection of update rates for the entire database, we use the vector $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$.

Age Dynamics: As the data items in the local cache are subject to updates in the back-end database, it is possible for the cached items to be older versions of the content. To quantify the freshness of the local content, we use *Age-of-Version* (AoV). Specifically, we define the AoV $\Delta_n(t) \in \{0, 1, \ldots\}$ of a cached content for item n at time t as the number of updates that item n has received in the back-end database since it was most recently cached. This metric allows us to measure the difference in versions between the database and the local cache.

With the dynamics now established, we can proceed to discuss the operational and performance costs that are essential to our caching system.

Fetching and Aging Costs: On the operational side, we denote the cost of fetching an item from the back-end database to the local cache by $b_n\, c_f > 0$ where b_n is the size of the item. On the performance side, we assume that serving an item n from the local cache with age $\Delta_n(t)$ incurs a *freshness/age* cost of $c_a \times \Delta_n(t)$ for some $c_a \geq 0$, which grows linearly[2] with the AoV metric. This aging cost measures the growing discontent of the user for receiving an older version of the content she/he demands. For each data item n, we define the binary action $u_n(t) \in \{0, 1\}$, to capture the decision of fetching the most recent version of content n to the edge-cache at time t, i.e., $u_n(t) = 1$. On the other hand, $u_n(t) = 0$ captures the case when the request will be served from the edge-cache incurring the freshness cost.

Problem Statement: Our main goal is to develop efficient caching strategies that strike the right balance between the costs of frequently updating local content and providing aged content to users. Specifically, we seek a policy that minimizes the long-term average cost by making decisions on cache updates and request serving from the edge-cache.

[1]Consequently, we consider the system to evolve in continuous time.

[2]While this linearity assumption is meaningful as a first-order approximation to the aging cost and facilitates simpler expressions in the analysis, it can also be generalized to convex forms to extend this basic framework.

This objective can be formulated as follows:

$$\min_{\boldsymbol{u}} \lim_{\mathrm{T}\to\infty} \frac{1}{T}\mathbb{E}_s^u \left[\int_o^{\mathrm{T}} \sum_{n=1}^{N} C_n \left(s_n(t), r_n(t), u_n(t)\right) dt \right] \quad (1)$$
$$s.t. \quad u_n(t) \in \{0,1\}, \quad \forall n \in \mathcal{N}, t \geq 0,$$

where $u_n(t) \in \{0,1\}$ and $s_n(t)$ represent the action and state of the system at time t for data item n, respectively. The state is defined as $s_n(t) = \{t - l : l = \max(t' \leq t : u_n(t') = 1)\} \geq 0, \forall n \in \mathcal{N}$ and it captures the elapsed time since the last time item n was updated in the cache. Moreover, $r_n(t) \in \{0,1\}$ is an indicator variable for the presence of a request for item n at time t. The cost of the system for data item n at time t, given the state $s_n(t)$ and action $u_n(t)$ is denoted as C_n and given by:

$$\mathbb{E}_s^u \left[C_n(s_n(t), r_n(t), u_n(t))\right] = u_n(t) b_n c_f + r_n(t) s_n(t) \lambda_n c_a (1 - u_n(t)),$$

where the first term captures the cost of fetching the freshest version from the database and the second term captures the freshness cost due to serving a potentially aged item from the cache. The operator $\mathbb{E}_s^u$ captures the expected value over the system state s for a given action u. Our aim is to find the optimal policy $\boldsymbol{u}$ that minimizes this long-term average cost.

Here, the local cache is responsible for deciding when to request the most recent version of locally cached item n from the back-end database represented by $u_n(t) \in \{0,1\}$, at a fetching cost of c_f. Additionally, while the local cache has complete knowledge of the request arrival times, it does not have information regarding the exact age of the cached content. Therefore, the optimal policy aims to determine action vector $\boldsymbol{u}$, which dictates when to pull the item from the database, based on the known request arrival times.

The general problem expressed in 1 falls under the scope of Partially Observable Markov Decision Processes (POMDP), and quickly becomes intractable. Even formulating the problem explicitly, let alone solving it, becomes practically impossible. Therefore, a more productive approach is needed to attack this problem in order to develop algorithms and principles with performance guarantees. In this work, we first focus on a specific scenario where requests arrive at the local cache following a Poisson Process. Additionally, we assume that the local cache has knowledge of item popularity, refresh rates, and demand intensity. These assumptions enable us to analyze the optimal policy's structure, which will be discussed in the next section. Leveraging the insights gained from the optimal policy's structure, we will introduce a model-based learning approach for caching dynamic content.

III. The Optimal Policy

In this section, we direct our attention to the problem introduced in the preceding section. Our objective is to analyze the optimal policy's structure in a scenario where requests arrive at the local cache according to a Poisson process with a known rate β. Furthermore, the local cache, which has a limited storage capacity, possesses knowledge of the popularity profile $\mathbf{p}$ and refresh rate $\boldsymbol{\lambda}$. Given these assumptions, we provide an explicit characterization of the optimal policy through the following theorem.

Theorem 1: Consider a system comprising a dataset $\mathcal{N}$ of N dynamic items, where each item has a corresponding size denoted as $(\boldsymbol{b})_{n=1}^N$. In this system, a local cache with a constraint on its average cache occupancy $\tilde{B}$ serves a group of users that generate demand according to a Poisson Process with a known rate β. The local cache possesses knowledge of the popularity profile $\mathbf{p} = (p_n)_{n=1}^N$ and refresh rates $\boldsymbol{\lambda} = (\lambda_n)_{n=1}^N$. Under these conditions, the optimal caching strategy is an eviction-based policy. Specifically, the local cache assigns a constant timer to each item upon its storage, and the item is evicted from the cache when the timer expires. The timers for each item are determined as follows:

$$\tau_n^*(\beta, \mathbf{p}, \boldsymbol{\lambda}) = \frac{1}{\beta p_n} \left[\sqrt{1 + 2\, b_n \frac{\beta p_n c_f - \tilde{\alpha}^*}{c_a \lambda_n}} - 1 \right]^+, \quad (2)$$

where $\tilde{\alpha}^* \geq 0$ is selected such that the following conditions are satisfied.

$$\tilde{\alpha}^* \left(\sum_{n=1}^{N} \frac{\beta p_n \tau_n^*}{\beta p_n \tau_n^* + 1} b_n - \tilde{B} \right) = 0, \sum_{n=1}^{N} \frac{\beta p_n \tau_n^*}{\beta p_n \tau_n^* + 1} b_n \leq \tilde{B} \quad (3)$$

Proof. We showed in [24] that for the general optimization problem described in Equation 1, the optimal policy follows a threshold-based structure. In particular, in order for an update or fetch event to occur, two conditions must be met. Firstly, there must be a request for the item at that particular time. Secondly, the time elapsed since the last update of the item must exceed a certain threshold value. Thus, the optimal policy can be expressed as:

$$u_n^*(s_n, r_n) = \begin{cases} r_n & s_n > \tau_n^*, \\ 0 & s_n \leq \tau_n^*, \end{cases} \quad (4)$$

Under such an eviction-based policy, the average long-term cost and the average cache occupancy can be given as:

$$C(\boldsymbol{\tau}, \beta, \mathbf{p}) = \min_{\boldsymbol{\tau} \in \mathcal{T}} \beta \sum_{n=1}^{N} p_n \frac{\frac{1}{2} c_a \lambda_n p_n \beta\, \tau_n^2 + b_n\, c_f}{1 + \beta p_n\, \tau_n}, \quad (5)$$

$$B(\boldsymbol{\tau}, \beta, \mathbf{p}) = \sum_{n=1}^{N} \frac{\beta p_n \tau_n}{1 + \beta p_n \tau_n} b_n. \quad (6)$$

Where $\boldsymbol{\tau} = (\tau_n)_{n=1}^N$ is the time each item will remain in the cache before eviction. Therefore, the cost-minimization problem with a constraint on the average cache occupancy would be as follows:

$$\min_{\boldsymbol{\tau} \geq 0} \quad C(\boldsymbol{\tau}) \quad (7)$$
$$s.t. \quad B(\boldsymbol{\tau}) \leq \tilde{B}.$$

This is not a convex optimization problem. However, we take the following approach to solve it. Define the feasible set $\mathcal{F}_{\tilde{B}}$ as:

$$\mathcal{F}_{\tilde{B}} = \left\{ \boldsymbol{\tau} | \quad \tau_n \geq 0, g(\tau) = \sum_{n=1}^{N} \frac{\beta p_n \tau_n}{1+\beta p_n \tau_n} b_n \leq \tilde{B} \right\}$$

which is a non-convex set. Then the cost optimization problem (7) can be expressed as:

$$\min_{\boldsymbol{\tau} \in \mathcal{F}_B} C(\boldsymbol{\tau}). \tag{8}$$

The authors in [25] demonstrated that for any optimization problem $\min_{\boldsymbol{\tau} \in \mathcal{F}} C(\boldsymbol{\tau})$, if all the following hold:
1) Slater condition,
2) non degeneracy assumption for $\forall \boldsymbol{\tau} \in \mathcal{F}$,
3) $\exists \boldsymbol{\tau}' \in \mathcal{F} : \forall \boldsymbol{\tau} \in \mathcal{F}, \exists \boldsymbol{t}_n \downarrow 0$ with $\boldsymbol{\tau}' + \mathbf{t_n}(\boldsymbol{\tau} - \boldsymbol{\tau}') \in \mathcal{F}$,
4) $L_C(\boldsymbol{\tau}) = \left\{ \boldsymbol{\tau}' \in R^N : C(\boldsymbol{\tau}') < \mathrm{C}(\boldsymbol{\tau}) \right\}$ is a convex set,
then if $\boldsymbol{\tau}$ is a non trivial KKT point, it is a global minimizer.

To check that Slater condition holds for any $0 < \tilde{B} < N$, assume $\tau_n = \frac{1}{2\beta p_n} \frac{\tilde{B}}{N-\tilde{B}} > 0, \forall n \in \mathcal{N}$ which gives $g(\boldsymbol{\tau}) < \tilde{B}$. So choose $\boldsymbol{\tau} = \frac{1}{2\beta} \frac{\tilde{B}}{N-\tilde{B}} (\frac{1}{p_1}, ..., \frac{1}{p_N}) \in \mathcal{F}_{\tilde{B}}$ which is a feasible point and all the inequalities are inactive.

To check the non-degeneracy assumption, we need to show that every where that a constraint is active, it's gradient is nonzero. Since constraints $\tau_n \geq 0, \forall n \in \mathcal{N}$ have always nonzero gradient, so we only need to check this for $g(\boldsymbol{\tau}) = \sum_{n=1}^{N} \frac{\beta p_n \tau_n}{\beta p_n \tau_n + 1} b_n - \tilde{B}$. We have $\nabla g(\boldsymbol{\tau}) \neq \mathbf{0}$. To check the third condition, consider $\boldsymbol{\tau}' = (0, ..., 0) \in \mathcal{F}_B$ and choose $t_n = \frac{c}{n}$ such that $c\boldsymbol{\tau} \in \mathcal{F}_{\tilde{B}}$ for a given $\boldsymbol{\tau}$. Then for this choice of $\boldsymbol{\tau}'$ and t_n we can show that condition 3 holds for all $\tau \in \mathcal{F}_{\tilde{B}}$. To check the last condition, notice that $L_C(\boldsymbol{\tau}) = \left\{ \boldsymbol{\tau}' \in R^N : C(\boldsymbol{\tau}') < \mathrm{C}(\boldsymbol{\tau}) \right\}$ is sub level set of the convex function $C(\boldsymbol{\tau})$ and therefore is also itself a convex set.

The non-trivial KKT solution to the problem 7 can be expressed as:

$$\tau_n^* = \frac{1}{\beta p_n} \left[\sqrt{1 + 2\, b_n \frac{\beta p_n c_f - \tilde{\alpha}^*}{c_a \lambda_n}} - 1 \right]^+,$$

where $\tilde{\alpha}^* \geq 0$ is the Lagrange multiplier, chosen to satisfy the conditions in Equation 3. As demonstrated in [25], such a non-trivial KKT point is a global minimizer for the average cost minimization, completing the proof. ■

As evident from Equation 2, the timers can be explicitly determined as a function of the system parameters, including the popularity, refresh rate, and size of the items. The parameter $\tilde{\alpha}$ is selected such that the constraint $\tilde{B}$ on the average cache occupancy is satisfied. The optimal value $\tilde{\alpha}^*$ exhibits an inverse relationship with $\tilde{B}$. As the cache capacity increases, $\tilde{\alpha}^*$ tends to approach zero. In the specific scenario of an unlimited cache capacity, this corresponds to setting $\tilde{\alpha}^* = 0$.

The following proposition provides a characterization of the optimal cost when the cache capacity is unlimited. We will utilize this cost as a benchmark for comparing the performance of our proposed algorithms.

Proposition 1: Under the conditions outlined in Theorem 1, in the scenario where the cache storage capacity is unlimited, the optimal cost can be expressed as follows:

$$C^*(\beta, \mathbf{p}, \boldsymbol{\lambda}) = \sum_{n=1}^{N} c_0 \lambda_n \left(\sqrt{1 + 2\, b_n \frac{\beta p_n}{\lambda_n} \frac{c_f}{c_0}} - 1 \right). \tag{9}$$

Proof. In the case of unlimited cache size, the Lagrange multiplier reduces to $\tilde{\alpha}^* = 0$. Substituting this value into Equation 2, we can express the optimal holding times as:

$$\tau_n^* = \frac{1}{\beta p_n} \left(\sqrt{1 + 2\, b_n \frac{\beta p_n c_f}{c_a \lambda_n}} - 1 \right).$$

By substituting these optimal holding times into the average cost given in Equation 5, the optimal cost under unlimited cache capacity is obtained as presented in Equation 9. ■

While Proposition 1 provides the optimal cost for unlimited cache capacity, real-world scenarios always have limited average cache occupancy due to the dynamic nature of content. Cached items must be periodically updated to maintain freshness, leading to eviction. The optimal policy's eviction-based structure keeps items in the cache briefly before eviction, ensuring that average cache occupancy remains bounded even with unlimited storage. Additionally, as content becomes more dynamic, cache occupancy further decreases.

In the next section, we build on the insights from Theorem 1 to propose a learning-based algorithm. This algorithm avoids the exploration phase typical of RL training, saving time and computational resources. Additionally, its use of moving average filters ensures adaptability and robustness to changing system parameters, making it both practical and efficient.

IV. Model-Based Learning

Our main goal is to design a robust caching policy that efficiently handles dynamic content without prior knowledge of its popularity, refresh rates, or demand intensity. The caching policy must continuously learn and adapt to changing popularity and refresh rates, ensuring the policy effectively minimizes long-term average costs.

We introduce Algorithm 1, a model-based caching policy for dynamic content based on Theorem 1. This algorithm uses the theoretical framework to guide its decisions effectively. Inspired by Theorem 1, the policy features an eviction-based structure, dynamically adjusting timers to maintain average cache occupancy below the specified threshold while minimizing average costs.

The proposed algorithm utilizes three parameters to effectively manage the caching process. These parameters include the last request time of items ($\bar{\boldsymbol{s}}$), the last fetch time of items ($\bar{\boldsymbol{t}}$), and the corresponding age of items during the last fetch ($\boldsymbol{\Delta}$). Based on these observed parameters, the algorithm uses two moving average filters with time averaging parameter θ to estimate refresh rates ($\hat{\boldsymbol{\lambda}}$) and interarrival times ($\boldsymbol{eit}$). These estimated values are then utilized to calculate the

Algorithm 1 Model Based Caching for Dynamic Content

Require: $c_f, c_a, N, \boldsymbol{b}, B, \theta$

Initialization: $\alpha = 0, \boldsymbol{eit} = [0]^N, \hat{\boldsymbol{\lambda}} = [0]^N, \boldsymbol{\tau} = [0]^N, \Delta = [0]^N, \bar{\boldsymbol{t}} = [0]^N, \bar{\boldsymbol{s}} = [0]^N, \bar{B} = 0$
while there is a request for an item **do**
  $n =$ index of the requested item
  $t =$ time $- \bar{\boldsymbol{t}}_n$
  $s =$ time $- \bar{\boldsymbol{s}}_n$
  $\bar{\boldsymbol{s}}_n =$ time
  if $t \geq \tau_n$ **then**
    Fetch item n and read its age, δ
    Calculate current cache occupancy $\hat{B}$
    $\bar{\boldsymbol{t}}_n =$ time
    $\hat{\lambda}_n = (1-\theta)\hat{\lambda}_n + \theta \frac{\delta - \Delta_n}{t}$
    $\Delta_n = \delta$
    $\tau_n = eit_n \left[\sqrt{1 + 2b_n \frac{c_f - eit_n \alpha}{c_a \hat{\lambda}_n eit_n}} - 1 \right]^+$
  end if
  $eit_n = (1-\theta)\, eit_n + \theta\, s$
  $\bar{B} = (1-\theta)\, \bar{B} + \theta\, \hat{B}$
  $\alpha = \max\left(0,\ (\bar{B} - B)/(\max_n(\boldsymbol{eit}))_{n=1}^N\right)$.
end while
return $\boldsymbol{\tau}$ *end*

timers ($\boldsymbol{\tau}$) for each item, determining how long they will remain in the cache before eviction:

$$\tau_n = eit_n * \left[\sqrt{1 + 2b_n \frac{c_f - eit_n * \alpha}{c_a \hat{\lambda}_n eit_n}} - 1 \right]^+, \qquad (10)$$

The algorithm begins with $\alpha = 0$ to calculate the timers. It then monitors the difference between the current cache occupancy and the target cache occupancy B, continuously adjusting α to ensure the average cache occupancy stays below the specified threshold.

To evaluate the proposed algorithm against the benchmark cost in Proposition 1, we consider a system with $N = 1000$ items, each with a size of $b = 10$, and refresh rates $\lambda = 20$. Additionally, we assume that user requests for items follow a Zipf popularity distribution with a parameter of $z = 1$, where p_n is proportional to $\frac{1}{n}$. Simulations are conducted under various average cache occupancy constraints, denoted as B. The fetching cost per item size is $c_f = 1$, the aging cost per single age is $c_a = 0.1$, and the time averaging parameter is $\theta = 0.005$. These parameters will be used throughout unless specified otherwise.

We use the percentage cost increase as our comparison metric, defined as follows:

$$\text{Percentage Cost Increase}(\%) = 100 \times \frac{C^{MB} - C^*(\beta, \mathbf{p}, \boldsymbol{\lambda})}{C^{MB}}, \qquad (11)$$

where C^{MB} represents the average long-term cost resulting from the proposed model-based learning policy, and $C^*(\beta, \mathbf{p}, \boldsymbol{\lambda})$ denotes the optimal average cost given in Equation 9, considering unlimited cache capacity and perfect knowledge of the system parameters.

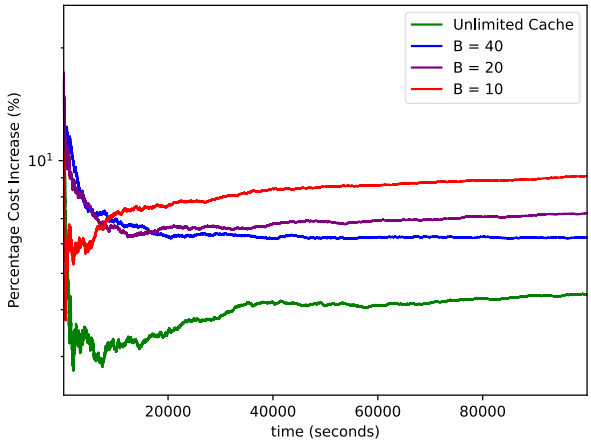

Fig. 2: Cost Comparison: Model-based vs. Optimal Policy

Fig. 2 shows the percentage cost increase of the proposed caching policy compared to the optimal strategy. Under the unlimited cache capacity, the proposed policy results in only a 4% cost increase, approaching the optimal cost. This performance is remarkable given that the proposed policy lacks knowledge of the popularity profile, refresh rates, and demand intensity, unlike the optimal policy. As the cache size decreases, the performance gap between the proposed policy and the optimal strategy widens slightly. However, even in these constrained scenarios, the cost increase remains relatively low, below 10%. This underscores the effectiveness of the proposed policy in leveraging estimated popularity and refresh rates to achieve a lower average cost under limited cache constraints.

Up to now, our analysis has assumed requests arrive at the local cache following a Poisson process, allowing us to derive the optimal policy structure and propose Algorithm 1. However, real-world scenarios may not conform to exponential interarrival times. To address this, we propose a model-free RL approach in the next section, enabling the system to learn an optimal policy without assuming specific request arrival distributions.

V. Model-Free Learning

In this section, we extend our analysis beyond the Poisson Process assumption and consider the scenario where arrival requests do not follow a known distribution. We propose a learning approach for caching dynamic content based on RL, aiming to enhance the efficiency of content delivery in such dynamic environments.

More specifically, we take the Q-Learning approach where the evolution of Q's for each data item n can be expressed as:

$$\begin{aligned} &Q(S_t, u_t) \leftarrow Q(S_t, u_t) \\ &+ \alpha \left[R_{t+1} + \gamma \max_u Q(S_{t+1}, u) - Q(S_t, u_t) \right], \end{aligned} \qquad (12)$$

where S_t, S_{t+1}, and u_t represent the current state, next state, and the action taken at time t, respectively. R_{t+1} is the

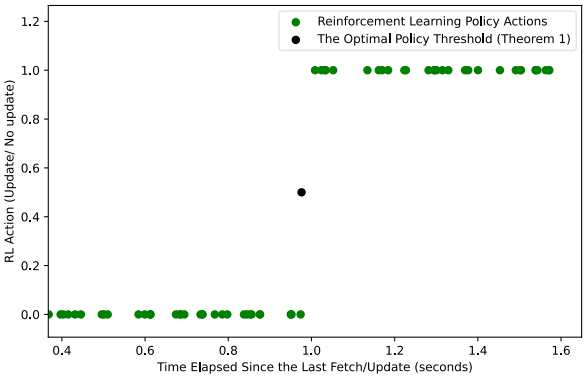

Fig. 3: Actions Under the RL Caching Policy

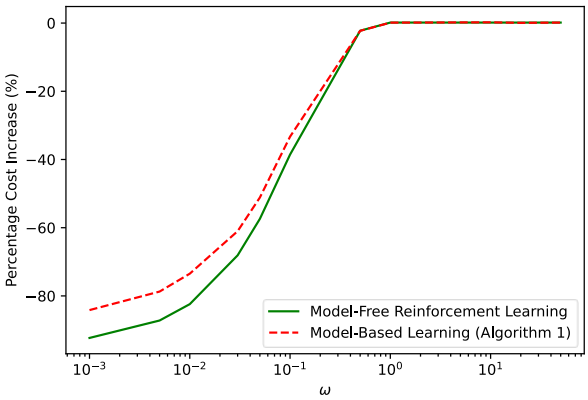

Fig. 4: Cost Comparison: Model-based vs. RL Policy

reward observed at the next state. Due to the unavailability of the probability distribution $P(S_{t+1}|S_y, u_t)$, the learning approach we employ is considered model-free. At each time step, there are two potential actions: cache update/fetch, denoted as $u_t = 1$, and no cache update, represented by $u_t = 0$. According to [24], actions should be taken only at the time of request arrivals and using the estimated value for Q's, the optimal action at each state S_t is given by:

$$u_t^* = \underset{u\in\{0,1\}}{argmin}\ Q(S_t, u).$$

Given that the state S_t evolves continuously over time $[0, \infty)$, we discretize time into steps of size 0.1 seconds before applying the Q-Learning algorithm.

Fig. 3 illustrates how the model-free RL policy decides whether to perform an "update/fetch" (i.e., $u_t = 1$) or take no action (i.e., $u_t = 0$) based on the time that has passed since the last fetch (i.e., $S_t > 0$). We can see that this policy has a threshold based structure where updates/fetches are done only when a request comes in and the elapsed time reaches a certain threshold value. Interestingly, the time threshold used by the model-free RL policy is exactly the same as the one described in Theorem 1 for the optimal policy under Poisson process arrivals (black dot on the figure). This similarity demonstrates that the RL policy accurately replicates the structure of the best possible policy under the Poisson process arrivals.

Next, we want to compare the performance of the model-free RL policy with the proposed model-based algorithm in more general scenarios.

VI. Comparison: Model-based vs Model-free Learning

In this section, we compare the performance of model-based and model-free learning approaches under varying conditions. We consider the scenario where interarrival times of each data item n follow a Gamma distribution with parameters $(\omega, \frac{1}{\omega\beta p_n})$, ensuring that the average interarrival time is $\frac{1}{\beta p_n}$, regardless of ω. The parameter $\omega > 0$ determines the distribution's variance, which is $\frac{1}{\omega(\beta p_n)^2}$. Notably, when $\omega = 1$, this distribution becomes exponential, allowing a meaningful performance comparison of the algorithms.

For our simulations, we assume a system with $N = 1000$ items, each of size $b = 1$, and a uniform popularity distribution ($p_n = 0.001$ for all n). Each item has a refresh rate of $\lambda = 20$, and the interarrival times follow a Gamma distribution with demand intensity $\beta = 5$. By varying ω, we analyze how different arrival distributions impact the caching algorithms' performance.

Fig. 4 shows the percentage cost increase of model-based and model-free policies compared to the benchmark cost given in Equation 9, as functions of the environment parameter ω over a 10^6 seconds duration. Surprisingly, model-based learning, designed for Poisson arrivals ($\omega = 1$), performs nearly as well as model-free RL in more general environments, maintaining high gains as the variance of interarrival times decreases ($\omega > 1$). However, when the variance increases ($\omega < 1$), model-free RL outperforms model-based learning.

To better demonstrate the gains and the convergence rate of the policies, we use the following percentage cost increase as our metric to depict Fig. 5:

$$\text{Percentage Cost Increase}(\%) = 100 \times \frac{C^{MF} - C^{MB}}{C^{MB}},$$

where C^{MB} and C^{MF} represent the average long-term costs of the proposed model-based learning policy and the model-free RL policy, respectively.

Fig. 5 also confirms the high gains of model-free RL compared to model-based learning in high-variance scenarios. However, for $\omega = 0.001$, corresponding to high variance in interarrival times, even though, the RL caching outperforms the model-based approach by nearly 50%, this significant gain is achieved after more than 10^6 seconds due to the extensive training time required for the RL policy to explore and learn the environment. Model-based learning performs almost as well as model-free RL in many scenarios and does not require an exploration phase. Utilizing a moving average filter with a time averaging parameter of $\theta = 0.005$, the model-based approach takes only $\frac{1}{\theta} = 200$ samples to adapt

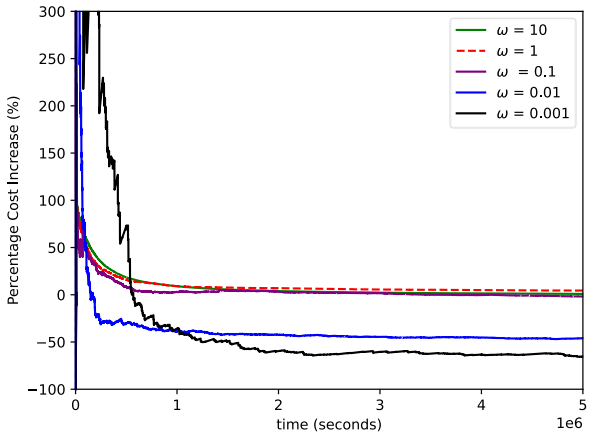

Fig. 5: Cost Comparison: Model-based vs. RL Policy

to new environments. In contrast, model-free RL must re-explore the new environment and construct a new policy, resulting in longer adaptation times.

Thus, model-based learning converges faster, requiring less training time, and performs competitively in environments with manageable variance in interarrival times. Additionally, since model-based learning utilizes a moving average filter, it quickly adapts to changes in popularity, refresh rates, and demand intensity without retraining, unlike the model-free RL caching, which necessitates longer training times and frequent retraining for each environmental change.

VII. Conclusion

In this study, we propose a model-based framework for caching dynamic content that leverages prior assumptions to derive a closed-form solution. We first characterize the optimal policy's structure under Poisson arrivals and use these insights to design a model-based learning algorithm. This algorithm learns faster than RL policies, avoids resource-intensive training, and remains robust in changing environments. It achieves near-optimal costs and manages limited cache capacity effectively. Compared to RL policies, our model-based approach performs equally well in most scenarios, offering resilience to environmental changes while eliminating the exploration phase.

References

[1] N. Bartolini, F. L. Presti, and C. Petrioli, "Optimal dynamic replica placement in content delivery networks," in *The 11th IEEE International Conference on Networks, 2003. ICON2003.* IEEE, 2003, pp. 125–130.

[2] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Single vs distributed edge caching for dynamic content," *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 669–682, 2021.

[3] S. Zhang, J. Li, H. Luo, J. Gao, L. Zhao, and X. S. Shen, "Towards fresh and low-latency content delivery in vehicular networks: An edge caching aspect," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP).* IEEE, 2018, pp. 1–6.

[4] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh, "Fresh caching for dynamic content," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[5] ——, "Fresh caching of dynamic content over the wireless edge," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2315–2327, 2022.

[6] E. Najm and R. Nasser, "Age of information: The gamma awakening," in *2016 IEEE International Symposium on Information Theory (ISIT)*. Ieee, 2016, pp. 2574–2578.

[7] K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal, "Enabling dynamic content caching for database-driven web sites," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 532–543.

[8] W.-S. Li, O. Po, W.-P. Hsiung, K. S. Candan, and D. Agrawal, "Freshness-driven adaptive caching for dynamic content web sites," *Data & Knowledge Engineering*, vol. 47, no. 2, pp. 269–296, 2003.

[9] P. Wu, J. Li, L. Shi, M. Ding, K. Cai, and F. Yang, "Dynamic content update for wireless edge caching via deep reinforcement learning," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1773–1777, 2019.

[10] M. Bastopcu and S. Ulukus, "Maximizing information freshness in caching systems with limited cache storage capacity," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 423–427.

[11] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Optimal load-splitting and distributed-caching for dynamic content," in *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt).* IEEE, 2021, pp. 1–8.

[12] ——, "Optimal load-splitting and distributed-caching for dynamic content over the wireless edge," *IEEE/ACM Transactions on Networking*, 2023.

[13] X. Zhou, W. Wang, N. U. Hassan, C. Yuen, and D. Niyato, "Age of information aware content resale mechanism with edge caching," *IEEE Transactions on Communications*, vol. 69, no. 8, pp. 5269–5282, 2021.

[14] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Achieving freshness in single/multi-user caching of dynamic content over the wireless edge," in *2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT).* IEEE, 2020, pp. 1–8.

[15] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *2012 Proceedings IEEE INFOCOM Workshops*. IEEE, 2012, pp. 268–273.

[16] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Wireless multicasting for content distribution: Stability and delay gain analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1–9.

[17] ——, "Delay gain analysis of wireless multicasting for content distribution," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 529–542, 2020.

[18] S. Sethumurugan, J. Yin, and J. Sartori, "Designing a cost-effective cache replacement policy using machine learning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA).* IEEE, 2021, pp. 291–303.

[19] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 28–35, 2018.

[20] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 181, p. 103005, 2021.

[21] 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.

[22] 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.

[23] E. Ghoreishi, B. Abolhassani, Y. Huang, S. Acharya, W. Lou, and Y. T. Hou, "Cyrus: A drl-based puncturing solution to urllc/embb multiplexing in o-ran," in *2024 33rd International Conference on Computer Communications and Networks (ICCCN).* IEEE, 2024, pp. 1–9.

[24] B. Abolhassani, J. Tadrous, A. Eryilmaz, and S. Yüksel, "Optimal push and pull-based edge caching for dynamic content," *IEEE/ACM Transactions on Networking*, 2024.

[25] Q. Ho, "Necessary and sufficient kkt optimality conditions in non-convex optimization," *Optimization Letters*, vol. 11, no. 1, pp. 41–46, 2017.