Optimizing Freshness of Machine Learning Model: A Timing (Multi-Armed) Bandit Approach

Qiulin Lin, Liyuan Wang, Junyan Su, and Minghua Chen

Abstract

Maintaining the freshness of machine learning (ML) models, i.e., the time elapsed since their last update and thus representing their ability to adapt to data drifting, is crucial for AI systems to deliver optimal performance. This is particularly relevant for resource-intensive models like Large Language Models, where frequent retraining or fine-tuning incurs significant costs. This paper addresses the challenge of jointly optimizing model freshness and update expense by carefully timing and tailoring ML model updates. We formulate this puzzle as a novel *consecutive* multi-armed bandit problem, where each "arm" represents a specific ML model update strategy with associated update costs and unknown, stochastic, model staleness cost. The consecutive nature of the problem comes from the realistic constraint that the feedback of the arms arrives from a consecutively ordered set in each inter-update interval. We develop an efficient online algorithm for the problem and prove it achieves sublinear regret. Our approach enables the dynamic allocation of resources towards retraining or fine-tuning, striking a balance between maintaining ML model freshness and minimizing update expenses. Simulations based on real-world traces demonstrate that our approach significantly outperforms state-of-the-art alternatives, reducing the total cost of ML model performance degradation and update expense by up to 48.7% under typical settings.

I. Introduction

Machine learning models have become increasingly prevalent across a wide range of systems, from recommendation engines [1] and image recognition [2] to autonomous vehicles [3] and networking [4]. Usually, learning models are trained and validated using collections of data and then applied to provide accurate inferences or predictions, make data-driven decisions, and finally enhance system performance. However, similar to product aging, the efficacy of learning models tends to degrade over time [5], [6], due to changes in the underlying data distributions as compared with the training data set, which is commonly known as concept drift or data drift.

Maintaining model performance necessitates updates or adaptations to new circumstances [6]. However, the update or adaption process of a learning model can be costly. For instance, large-scale models used in natural language processing, content generation, or image recognition often require substantial computational resources, such as high-performance GPUs or distributed computing clusters, and incur a significant amount of energy consumption. Moreover, the process of collecting, pre-processing, and labeling high-quality training data is often labor-intensive and costly. Consequently, striking a balance between maintaining high model performance and managing the associated update costs is crucial.

A viable approach to this challenge is to strategically determine update timings, optimizing the trade-off between model staleness cost and update cost. While there have been previous studies along this line, they typically either focus on pure offline analysis [7] or rely heavily on using historical data to represent future inputs for algorithm parameterization [8]. However, these assumptions may not hold in practice and render the results inapplicable. Furthermore, the highly dynamic and uncertain nature of model degradation behavior complicates system design and analysis. Thus, optimizing learning model update timing in an *online* manner, without prior information about uncertain model degradation, remains a largely unaddressed problem. We defer a detailed discussion of related work to Sec. II.

In this paper, we study the problem of strategically determining the learning model update timings under realistic conditions where future model performance degradation cost or its distribution over time is unknown or unavailable. Our goal is to jointly minimize model performance degradation costs and update costs. The core challenge is to learn the uncertain staleness cost while making real-time update decisions, without prior access to degradation costs over time or their distributions. We frame this issue as a novel variant of the classical multi-armed bandit problem and propose online strategies that achieve low regret compared to the optimal offline strategy with known distribution. We make the following **contributions**.

▷ In Sec. III, we introduce the concept of model age to represent the freshness of a learning model and define the model degradation cost as a random variable associated with model age. We then formulate the problem of minimizing the total expected model degradation cost and update cost by optimizing the model update decisions at each time.

▷ In Sec. IV, we interpret the updating decisions at each slot as an equivalent form of determining the inter-update times and show that our problem is essentially a new variant of the multi-armed bandit problem. We call it a consecutive multi-armed bandit problem where it returns feedbacks of arm 1 to arm k consecutively in the following k slots when arm k is pulled. We develop a model updating policy and show it achieves sublinear regret compared with the optimal solution with full knowledge of the random model degradation cost.

Q. Lin, L. Wang, J. Su and M. Chen are with Department of Data Science, City University of Hong Kong, Hong Kong. E-mail: {qiulin.lin@, liyuawang9-c@my., junyan.su@my., minghua.chen@}cityu.edu.hk.

TABLE I: Comparison of variants of stochastic bandit problems. The *regret* column shows the achieved regret bound of the corresponding bandit problem. The $\tilde{O}(\cdot)$ drops the logarithmic factor in the big-O notation. We only demonstrate the problem-independent bound for fair comparison.

Bandit Type	**Decision Making**	**Feedback Timing**	**Cost/Reward**	**Regret**
Standard [9]	Sequential	Immediate	Chosen Arm	$\tilde{O}(\sqrt{KT})$
Batched [10]	Batched	Delayed	Chosen Arms	$\tilde{O}(\sqrt{KT})$
Combinatorial [11], [12]	Sequential	Immediate	Arm Combination	$\tilde{O}(\sqrt{KT})$
Renewal Process [13]	Intermittent	Delayed	Chosen Arm	Problem Dependent Only
Timing **(this work)**	Intermittent	Consecutive	Chosen Arm $\times$ Arm Index	$\tilde{O}(\sqrt{KT})$ (cf. Thm. 4)

$\triangleright$ In Sec. V, we conduct simulations to evaluate the empirical performance of our proposed model updating algorithm. We show that our algorithms significantly outperform state-of-the-art alternatives under various settings based on real-world traces, reducing the total cost of ML model performance degradation and update expense by up to 48.7% under typical settings.

II. Related Work

Maintaining the efficacy of learning models in dynamic real-world scenarios is one of the crucial steps in machine learning operations (MLOps) [14]. Different approaches have been proposed for such a purpose. Among them, learning under concept drift involves detecting the concept drift and making model updates or retrains to adapt to new data distributions, such as [15], [16] and more references in a survey [17]. Continual learning develops models that can learn from a continuous stream of data, retaining knowledge from previous tasks while learning new ones [18]. Another line of work considers developing rapid model retraining algorithms [19], [20]. These studies focus on the model design or adaption approaches for specific learning tasks and maintaining performance but do not consider the trade-off between the freshness of the model and the cost of updating the model.

In this work, we consider general learning models and focus on the optimal trade-off between updating the model to avoid staleness and keeping the model to reduce the updating cost, where there have been only a few studies. In [7], the authors propose a framework for assessing the costs and performance gains of model updates and making update decisions based on a return on investment (ROI) metric. In [8], the authors propose threshold-based model updating policies using various indicators, such as instant and cumulative model staleness costs. These approaches are either pure offline analysis or rely on historical data to determine the key parameter (e.g., threshold) in their design. In contrast, our work considers an online method where we do not have historical traces or prior information on the uncertain performance of the learning model. Instead, we need to learn how the model decays over time while making updating decisions.

Our problem is closely related to the stochastic multi-armed bandit problem [21], but we introduce a novel variant called the *consecutive* multi-armed bandit. This variant differs from traditional multi-armed bandit problems in both decision-making and feedback mechanisms. In a standard stochastic multi-armed bandit problem, the decision maker selects an arm and immediately receives feedback, which is a sample of the reward (or cost) generated from the arm's unknown distribution. In contrast, in our *consecutive* variant, selecting an arm with index k results in receiving feedback from arm 1 to arm k consecutively over the next k rounds but getting k times a reward (or cost) sample from arm k. The decision maker then makes the subsequent arm selection only after receiving all the feedback. There are also other kinds of stochastic multi-armed bandit problems in the literature. For example, in combinatorial multi-armed bandit problem [11], [12], [22], the decision maker chooses a combination of arms rather than a single arm at each round, and the reward/cost is based on the combination of chosen arms. Another example is the batched multi-armed bandit problem [23], [10], [24] where the decisions are made in batches and the feedback is delayed by the sides of the batch. We summarize the comparison of these bandit problems in Table I.

Our proposed problem is also related to the Age-of-Information (AoI) optimization in communication networks [25], [26], [27] in terms of the mathematical formulation. AoI captures the freshness of information received at the destination, and the general goal for AoI optimization is determining the update process of the information to maintain a low AoI. Our problem shares a similar spirit of optimizing the model updating cost to maintain the model's freshness. Moreover, our problem can be applied to the AoI optimization problems when the penalty on aging information follows some unknown random distribution. As a comparison, in the AoI literature, the authors commonly consider that the penalty is given and deterministic has been studied in the literature, e.g., [28]. In addition, our mathematical formulation also covers the one in [29], [30] as a special case where the update cost is considered to be a constant. The single-source time-varying cost of AoI with bandit feedback studied in [29] is closely related to our study. In their study, the authors consider an online setting under the adversary penalty function on aging information. They introduce an epoch-based formulation and apply the standard online convex optimization framework to learn the optimal inter-update time with low regret. They do not explore the feedback structure to achieve an

TABLE II: Key notations.

Notation	Definition
T	Total Number of time slot
a_t	The learning model age at slot t
$g_t(a_t)$	The random model degradation cost given age a_t at slot t
$\tilde{g}(a_t)$	The expected model degradation cost given age a_t
M	Upper bound on the expected model degradation cost
$C(a)$	The model update cost to update a model with age a
C	Upper bound on the model update cost, i,e, $C(a) \leq C$
x_t	The update decision at the beginning of slot t
k_i	the i-th inter-update time interval
K	The maximum inter-update time
μ_k	The normalized average cost of arm k

improved regret to the problem. In our study, we focus on model degradation following some unknown distribution depending on the model age and discuss how to utilize the feedback structure of the problem. We compare our results in Sec. IV.

III. Model and Problem Formulation

We consider that after a learning model is implemented in a real-world system, the operator keeps track of its performance and determines the update timing of the learning model to maintain its freshness. The goal is to minimize the total model update cost and model degradation cost due to staleness. We summarize the key notations in Table II.

A. System Model

We consider a slotted time horizon with T slots. The learning model is implemented at the beginning of the first slot. Similarly, we consider that updates of the learning model (if exist) become available at the beginning of slots.

Freshness of Learning Models. We denote the age of a learning model as a_t, which evolves in the following manner. In the first slot, the age of the learning model is initialized as one. At slot t, if there is no model update at the beginning of the slot, the age of the current model is increased by one. Otherwise, the age of the model is reset to one[1]. Consequently, we have that

$$a_t = \begin{cases} 1, & \text{if } t = 1 \text{ or the model is updated at } t; \\ a_{t-1} + 1, & \text{otherwise.} \end{cases} \tag{1}$$

We note that a smaller a_t indicates a fresher model. In our problem, we consider that there is a minimum requirement on the model update, i.e., the model can not stay un-updated for more than K slots to guarantee a minimum level of freshness. In such case, we have $a_t \leq K, \forall t \in [T]$. We will further discuss it in the problem formulation. Our definition of model age is similar to the Age of Information (AoI) concept in communication networks [25], [26]. We discuss the similarities and differences in more detail later in Remark 1 at the end of this section.

Model Degradation Cost. We denote the updated learning model at slot t as M_t. At slot t, if the current age of the model is a_t, it means that we are applying the updated model at slot $t - a_t + 1$, i.e., M_{t-a_t+1}. We define the model degradation cost as the performance loss of applying M_{t-a_t+1} at slot t and denote it as $g_t(M_{t-a_t+1})$. It could be the operational error of the learning model for the inference tasks in real-world systems, e.g., false recognition in image classification or forecast error in prediction tasks. We note that the operation error of a learning model can be affected by different factors, including training data quality, model bias, dynamic inference requests, etc., which results in uncertain operation errors in practice. We take it into account by considering that the degradation cost learning model, $g_t(M_{t-a_t+1})$ can be random. We assume that given the age a_t, $g_t(M_{t-a_t+1})$ for all slot $t \in [T]$ are independent and identically distributed.[2] We consider that $g_t(M_{t-a_t+1})$ is upper bounded by a constant (denoted as M), i.e., $g_t(M_{t-a_t+1}) \leq M$, for all $t \in [T]$ and $a_t \leq K$.

More specifically, we let $f_{G|\Delta}(g \mid \Delta)$ denote the conditional probability density function of G given Δ, where G is the random variable representing the model degradation cost, and Δ is the age of the model. At slot t, $g_t(M_{t-a_t+1})$ is randomly generated from the distribution $f_{G|\Delta}(g \mid \Delta = a_t)$. For ease of presentation, we simply use $g_t(a_t)$ to represent $g_t(M_{t-a_t+1})$. And $g_t(a_t) \sim f_{G|\Delta}(g \mid \Delta = a_t)$. We denote the expected value of $g_t(a_t)$ as $\tilde{g}(a_t)$, which is given by

$$\tilde{g}(a_t) = \int_0^M g \cdot f_{G|\Delta}(g \mid \Delta = a_t) \, \mathrm{d}g. \tag{2}$$

[1]Note that setting the age of the model to start from one instead of zero is for the convenience of presentation of our problem formulation, algorithm, and analysis. Our results follow in the same way when considering the age to start from zero.

[2]We make these assumptions to facilitate our problem analysis and algorithmic design. While these assumptions may not hold in real-world scenarios, we will use real-world traces in our simulations to evaluate the performance of our algorithm. The simulation results demonstrate the practical applicability of our designs despite the potential deviations from our initial assumptions.

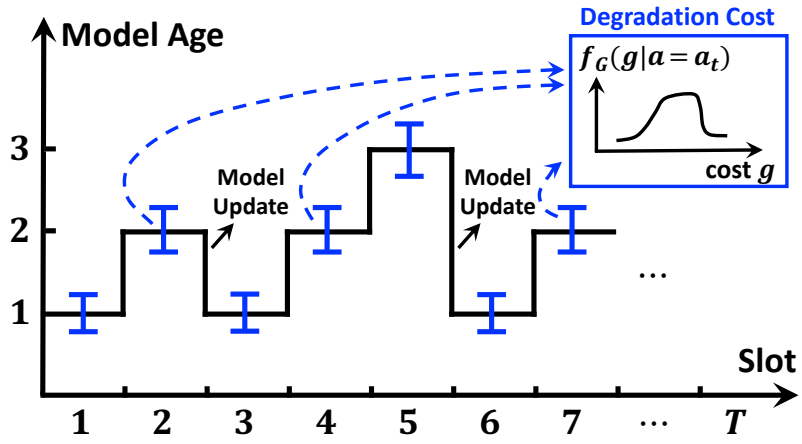

Fig. 1: An illustration of the evolution of model age and the associated degradation cost over time slots. At each time slot, the model age increments by one if no update is performed and resets to one upon updating the model. It incurs a random degradation cost following the distribution $f_{G|\Delta}(g \mid \Delta = a_t)$ when using an aging model with age a_t.

We assume that $\tilde{g}(a_t)$ is non-decreasing in a_t, which is intuitive that without updating for a longer period, it would result in a larger model gradation cost in expectation. This is also supported by empirical studies [5], [6] on temporal quality degradation in learning models like random forest and neural networks, etc.

Model Updating Cost. At each slot t, suppose the system operator decides to update the learning model. This update incurs a cost, which includes various components such as the cost of preparing additional training data, computing costs, deployment costs, and others [7]. We consider that the model update cost at slot t depends on the age of the model to update. Specifically, we define $C(a)$ as the cost to update a learning model with age a. We note that we consider a model update to occur at the beginning of the slot; that is, we are trying to update a model with age a_{t-1} if we update at slot t, and the update cost is $C(a_{t-1})$. We assume that $C_(a)$ is non-decreasing in a and is upper bounded by a constant denoted as C, i.e., $C(a) \leq C, \forall a \in [K]$. Our consideration covers a special case where the update cost is constant under different ages. It corresponds to the scenario when each update follows a similar procedure, incurring almost the same cost. In addition, we include the case that updating a more aging model may require more data preparation efforts and more computing resources, which will lead to a higher cost.

B. Problem Formulation

We consider the problem of optimizing the update decisions of the learning model to minimize the sum of the expected model degradation cost and the update cost.

$$\min \quad \sum_{t=1}^{T} \left(\tilde{g}(a_t) + C(a_{t-1}) \cdot x_t\right) \tag{3}$$

$$\text{s.t.} \quad a_t = \begin{cases} 1, & t = 1 \text{ or } x_t = 1, \\ a_{t-1} + 1, & t > 1 \text{ and } x_t = 0, \end{cases} \tag{4}$$

$$\text{var.} \quad x_t \in \{0, 1\}, a_t \leq K, \forall t \in [T]; x_1 = 0. \tag{5}$$

We use $x_t \in \{0, 1\}$ to represent the update decisions at slot t, where $x_t = 1$ means that we update the learning model at the beginning of slot t and $x_t = 0$ means not updating. We consider that our problem starts from the implementation of a most updated model and set $a_1 = 1$. There is no need to update the model at the beginning of the first slot, i.e., we set $x_1 = 0$.

C. Optimal Solution and Performance Metrics

Optimal Solution. We consider minimizing the long-term average total cost defined as

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \left(\tilde{g}(a_t) + C(a_{t-1}) \cdot x_t\right), \tag{6}$$

and derive the optimal solution when the information of the expected degradation cost $\tilde{g}(a_t), \forall a_t \leq K$, is available. This would serve as a baseline for evaluating the performance of an online approach without such information. The optimal solution to the problem could be derived as follows.

Theorem 1. *The optimal solution is to update the learning model periodically with the optimal period $k^* \in [K]$ such that*

$$k^* = \arg\min_{k\in[K]} \frac{\sum_{j=1}^{k} \tilde{g}(j) + C(k)}{k}. \tag{7}$$

And the optimal average cost is given by $\frac{\sum_{j=1}^{k^*} \tilde{g}(j)+C(k^*)}{k^*}$.

The proof is provided in Appendix A. We note that the consideration of long-term average total cost helps eliminate rounding issues when T is finite and makes the structure of the optimal solution clear.

Regret Minimization. We use regret as a performance metric for an online algorithm without the distribution information about the model degradation cost. We apply the optimal solution in Theorem 1 as a baseline. We denote the total cost of such baseline as $OPT(T)$ for a given period length T. Given an online algorithm, we denote its expected total cost as ALG. We define the regret $R(T)$ as the difference between these two total costs, which can be represented as follows,

$$R(T) = ALG(T) - OPT(T). \tag{8}$$

In general regret analysis, our goal is to develop online algorithms with low regret. Specifically, we aim to achieve sublinear regret, meaning that the regret $R(T)$ increases at a rate slower than linear with respect to T. Achieving sublinear regret ensures that the long-term average cost of the online algorithm converges to the same value as the optimal solution. In the next section, we will introduce our online algorithm and demonstrate that it achieves sublinear regret.

Remark 1. *AoI Minimization.* In communication networks, age-of-information is introduced to capture the freshness of information received at the destination [25], [26]. Our problem can be applied to the AoI optimization problems when the penalty on aging information follows some unknown random distribution, while the case when the penalty is given and deterministic has been studied in the literature, e.g., [28]. In addition, our problem formulation generalizes the one in [29], [30] where $C(a)$ is a constant for any $a \in [K]$, and we study a completely different online setting. We provided a more detailed comparison in the related work section; see Sec. II.

IV. Algorithm and Regret Analysis

In this section, we discuss our algorithm design and provide its performance analysis. We first show that our problem can be viewed as a new variant of the multi-armed bandit problem. Then, we design our algorithm by combining the idea of the arm elimination algorithm for standard multi-armed bandit problem [21] and the feedback and decision-making structure of our problem. We then show that our proposed algorithm achieves sublinear regret.

A. A New Variant of Multi-Armed Bandit Problem

Before we proceed, we first introduce some useful definitions. Inspired by the optimal periodic model update solution in Theorem 1, we consider the inter-update time between two successive updates. We define β_k as the expected model degradation cost aggregated by k slots from the last update, i.e.,

$$\beta_k = \sum_{j=1}^{k} \tilde{g}(j). \tag{9}$$

Suppose after the previous $(i-1)$-update, the online decision maker keeps the models for k_i slots before initializing a new update. We denote by U_T the total number of updates in T slots Then, the expected total cost can be represented as,

$$ALG(T) = \sum_{i=1}^{U_T} (\beta_{k_i} + C(k_i)) + \beta_{T-\sum_{i=1}^{U_T} k_i}. \tag{10}$$

And, the total cost of the optimal solution is given by,

$$OPT(T) = \lfloor T/k^* \rfloor \cdot (\beta_{k^*} + C(k^*)) + \beta_{T-\lfloor T/k^* \rfloor \cdot k^*}. \tag{11}$$

Then, we can represent the regret as follows.

$$\begin{aligned} R(T) \triangleq & \sum_{i=1}^{U_T} (\beta_{k_i} + C(k_i)) + \beta_{T-\sum_{i=1}^{U_T} k_i} \\ & - \left[\lfloor T/k^* \rfloor \cdot (\beta_{k^*} + C(k^*)) + \beta_{T-\lfloor T/k^* \rfloor \cdot k^*} \right]. \end{aligned} \tag{12}$$

Multi-Armed Bandit Paradigm. We now interpret our problem using the multi-armed bandit framework. We consider that we have K arms. We determine an inter-update time k after the last update, i.e., pull arm k. In the following k slots, we receive feedback on the degradation cost of a learning model with age from one to k consecutively, and it incurs a cost of $\beta_k + C(K)$.

And only after that, we make the next arm-pulling decision. We call such a variant as *consecutive* multi-armed bandit problem. Our goal is to achieve a low regret compared with the optimal inter-update time k^* (i.e., the best arm k^*). Mapping our problem to the MAB paradigm allows us to adapt its theoretical foundations and algorithm design ideas to develop effective solutions to our problem.

With the above interpretation, we define the normalized average cost of pulling arm k as

$$\mu_k \triangleq \frac{\beta_k + C(k)}{k} \frac{1}{C+M}. \tag{13}$$

We note that $\mu_k \in [0,1]$ for all $k \in [K]$. We have the expected cost of pulling an arm k as $k \cdot \mu_k \cdot (C+M)$, and it occupied k slots. We denote $n_k(T)$ as the number of times that the arm k is pulled. Then, we can consider the following definition of regret, which is easy to interpret under the multi-armed bandit paradigm. Let

$$\tilde{R}(T) \triangleq (C+M) \cdot \left[\sum_{k=1}^{K} k \cdot \mu_k \cdot n_k(T) - T \cdot \mu_{k^*} \right]. \tag{14}$$

Also, we show with the following lemma that it is sufficient to consider achieving a low regret under $\tilde{R}(T)$.

Lemma 2. *We have*

$$\left| R(T) - \tilde{R}(T) \right| \leq C + (K+1)M. \tag{15}$$

We leave the proof in Appendix B. With the above, it shows that an online algorithm achieves that same order of regret bounds under $R(T)$ and $\tilde{R}(T)$. In the following discussion, we will focus on minimizing $\tilde{R}(T)$. Under $\tilde{R}(T)$, our goal is to achieve a lower regret compared with always pulling the optimal arm k^*. This is accomplished by strategically sampling arms in a way that balances exploration (trying out different arms to learn their costs) and exploitation (choosing the best-known arm to minimize cost).

Challenges. We may try to directly apply the standard methods for stochastic multi-armed bandit problems, i.e., treating the problem as a stochastic MAB with K arms and rewards distributed over $[0,1]$ with unknown means $\mu_k, \forall k \in [K]$. However, one challenge here is that, as we discuss above, each pulling of arm k will occupy k slots and suffer k times its cost to receive one feedback of its cost. This makes it harder for the online decision maker to learn the cost as compared with the standard MAB where each pulling takes one slot and suffers one cost to get one feedback. Indeed we can show that if we directly apply the standard algorithms (more generally, all online algorithms that ignore the additional structures of the problems) will suffer a regret of at least $\Omega(K\sqrt{T}/\log K)$ in the worst case, which is greater than the lower bound $\Omega(\sqrt{KT}/\log K)$ in the standard stochastic multi-armed bandit problem. We summarize the results in the following proposition.

Proposition 3. *Consider a multi-armed bandit problem with K arms and rewards distributed over $[0,1]$ with unknown means $\mu_k, \forall k \in [K]$. At each slot, the decision maker pulls an arm (say arm k). It will occupy k slots, incur k times its average cost, and receive feedback on its cost. Then, any online algorithm for the problem will incur a regret of at least $\Omega(K\sqrt{T}/\log K)$ in the worst case.*

The proof is provided in Appendix C. In the next subsection, we explore the feedback structure of our formulated multi-armed bandit paradigm where at each pulling (say arm k), we can also obtain (by computation) feedback of the cost of each arm in $[k]$. With that, we are able to design a variant of the arm elimination algorithm and show that it achieves a regret of $\tilde{O}(\sqrt{KT})$ (ignoring the logarithms regarding K and T).

Remark. We compare with the result for the online approach for the single-sourcing time-varying cost of AoI with bandit feedback studied in [29] (shown in its long version at arxiv.org.). The authors in [29] introduce an epoch-based formulation, where a strategy fixes one inter-update time in an epoch and receives feedback on its cost. The authors consider the adversary cost function setting and apply the standard EXP3 algorithm for their problem. Directly mapping their result to our setting, it achieves a regret of $\tilde{O}(K\sqrt{T})$. In more detail, they show a standard regret of $O(\sqrt{\tilde{T}\tilde{M}\log\tilde{M}})$, where $\tilde{T}$ is the number of epochs and $\tilde{M}$ is the length of the epoch (and also the maximum inter-update time), when the total cost in an epoch is normalized to be within $[0,1]$. However, in the case when $\tilde{M}$ is large, such a normalization will imply the cost at each slot will approach zero, which may not be practical. In our case, we consider the cost at each slot will not diminish to zero, and thus, the cost in an epoch will be in an order of $\tilde{M}$. Taken it into account, their regret becomes $O(\tilde{M}\sqrt{\tilde{T}\tilde{M}\log\tilde{M}})$. Noting the total number of slot $T = \tilde{T} \cdot \tilde{M}$ and $K = \tilde{M}$, applying their result to our case can achieve a regret $\tilde{O}(K\sqrt{T})$, when directly following their analysis.

B. Our Proposed Algorithm

We now discuss our proposed algorithm to achieve a low regret $\tilde{R}_{\mathcal{A}}(T)$. We combine the idea of arm elimination for standard multi-armed bandit problem [21] and the structure of our problem, i.e., the consecutive feedback and intermittent decision making to design the algorithm. We summarize the algorithm in Algorithm 1.

Algorithm 1: Arm Elimination

1: Keep the model for K slots, receive degradation cost $g_t(a_t)$ with $a_t = t$, $\forall t \in [K]$. Perform an update at the end of slot K.
2: Initialize $\bar{\mu}_k = \frac{1}{(C+M)\cdot k}\left(C(k) + \sum_{i=1}^{k} g_i(i)\right)$ and $m(k) = 1$, $\forall k \in [K]$,
3: Initialize $n_k = 0, \forall k < K$ and $n_K = 1$.
4: Initialize the active set $\mathcal{A} = \{i | i \in [K]\}$.
5: **while** $t \leq T$ **do**
6: $j^* = \arg\min_{k\in[K]} \bar{\mu}_k$.
7: **for** $k \in \mathcal{A}$ **do**
8: **if** $\bar{\mu}_k - \bar{\mu}_{j^*} \geq 2\sqrt{\frac{2\log T}{m(t)}}$ **then**
9: $\mathcal{A} = \mathcal{A} - \{k\}$.
10: **end if**
11: **end for**
12: Let $i^* = \max_k\{k \in \mathcal{A}\}$.
13: **if** $t + i^* \leq T$ **then**
14: Keep the model for i^* slots, observe degradation cost $g_t(1), g_{t+1}(2) \cdots g_{t+i^*}(i^*)$ and perform an update at the end of $t + i^*$.
15: **for** $i \in \mathcal{A}$ **do**
16: Compute a sample of the cost of arm k, i.e., let $\hat{\mu}_i = \frac{1}{(C+M)\cdot i}\left(C(i) + \sum_{k=1}^{i} g_{t+k}(k)\right)$.
17: Update $\bar{\mu}_i = \frac{m(t)\cdot\bar{\mu}_i + \hat{\mu}_i}{m(t)+1}$.
18: **end for**
19: Update $m(t + i^*) = m(t) + 1$.
20: Update $n_{i^*} = n_{i^*} + 1$.
21: $t = t + i^*$
22: **else**
23: Keep the model till T, and let $t = T$.
24: **end if**
25: **end while**

We maintain a set $\mathcal{A}$ of candidate arms that are potentially the optimal arm k^*. After making the last update, we check our estimation of the cost of remaining arms in $\mathcal{A}$. If it is far away from the optimal arm we observed so far, then we eliminate such arm from $\mathcal{A}$. Note that $m(t)$ equals the number of samples we have so far for the cost of each arm in $\mathcal{A}$. Following Hoeffding Inequality, we can confidently eliminate an arm if the condition in Line 8 holds, i.e., such an arm is very likely not the optimal one. These correspond to Line 6 to Line 11 in Algorithm 1.

Then, we make our next arm-pulling decision (correspondingly, our next update timing following the multi-armed bandit interpretation in Sec. IV-A). We pull the arm with the maximum index in $\mathcal{A}$ to determine the next update time. According to the feedback structure of our problem, such a decision allows us to receive degradation cost feedback of all model ages within such maximum index and update the estimations of $\mu_k, \forall k \in \mathcal{A}$. We then update our estimation for all arms in $\mathcal{A}$. Such a design efficiently utilizes the feedback structure in that we only need to pull one arm to update the cost estimations for all arms in $\mathcal{A}$. As a comparison, the standard arm elimination solution requires pulling each arm in $\mathcal{A}$ once to update their estimations. We use $m(t)$ to denote the number of arm pulling or update decisions, and it also equals the number of samples we have so far for the cost of an arm in $\mathcal{A}$ in our design. These correspond to Line 12 to Line 24 in Algorithm 1. We summarize the performance guarantee of our proposed algorithm in the following theorem.

Theorem 4. *The regret of Algorithm 1 is upper bounded by* $\tilde{O}(\sqrt{KT})$.

We leave the proof in Appendix D. According to Theorem 4, it shows that our achieved an improved sublinear regret compared with not applying the problem structure. It implies that the long-term average cost of our proposed algorithm converges to the same value as the optimal solution. Our approach could effectively learn to make decisions that are nearly as good as those made with perfect knowledge.

V. Simulation

In this section, we compare our algorithm with state-of-the-art algorithms in real-world scenarios. Here is a brief introduction to all the algorithms we implement and evaluate.

- **CARA:** The algorithm proposed by [8]. It specifies a threshold from historical data. In the online phase, they will update the model only if the cumulative cost exceeds the threshold.

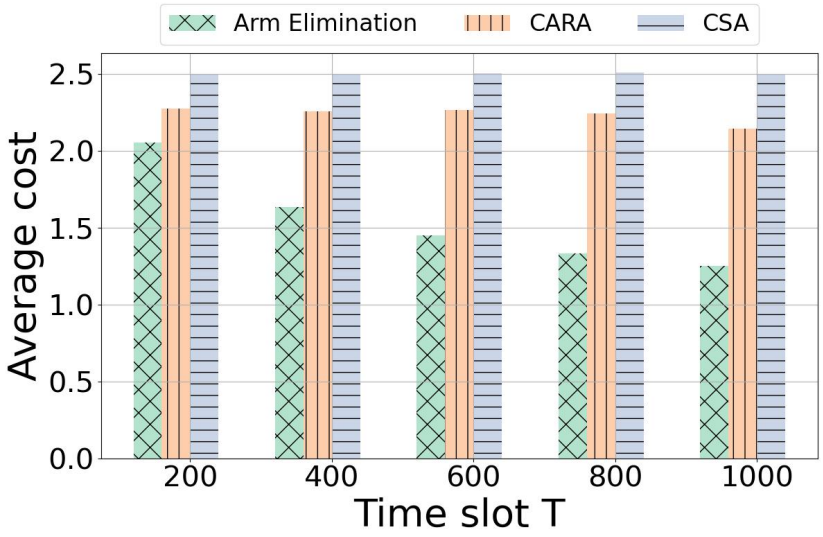

Fig. 2: The real-world scenario performances under different T.

- **CSA:** The algorithm adapted from [7]. It makes a balance between the improved performance and the update cost. They set a threshold and if the ratio of improved performance over update cost is larger than the threshold, they will update the model.
- **Arm Elimination:** Our proposed algorithm shown in Algorithm 1.

A. Recommendation Systems

Recommendation systems are widely used in real-world scenarios. In our simulation, we build a recommendation system that can recommend the top three business entities according to the user's features and preferences. We use a convolutional neural network to accomplish the task. We use the **YELP** dataset [31] to simulate the decay of the model in real-world scenario settings. The Yelp dataset comprises user reviews, business details, and check-in data, useful for research in text mining, sentiment analysis, and recommendation systems. As of the latest release, the Yelp dataset contains over 8 million user reviews and is approximately 9 GB in size. In our work, we use the dataset to simulate a recommendation system and observe the performance of the model when different algorithms are applied. The model's task is to predict the user's most suitable business entities.

1) Simulation Setups: In this scenario, we let the model degradation cost $g(a_t)$ be the **unsuccessful recall rate**. The task we set to the model is to predict the three most suitable business entities, if at least two can be found in the real top-three list, it can be deemed as a success. In our evaluation, we consider the case that the update cost $C(a) = s \cdot a + b$, we set $s = 1$, and $b = 1$,.

2) Performance Evaluation: In this subsection, we compare our algorithms with the conceivable alternatives and show the effectiveness of our algorithms. We use 6* Xeon E5-2678 v3 CPU which has 30G memory. The coding language we use is Python.

We can show that the worthy observation can be summarized in two points: our algorithms are robust when the parameters of this model change; our proposed algorithms outperform the conceivable alternatives.

Although in the theory analysis, we assume that $\tilde{g}(a_t)$ will only be relevant with $\tilde{g}(a_t)$, in the real simulation, the $g(a_t)$ shows some fluctuations which may indicate that $\tilde{g}(a_t)$ will slightly change according to t, yet our algorithms are robust to it and show good performance when some theoretical settings are violated.

Performance Comparison under Real-world Scenario. In the real-world simulation, in order to comprehensively verify our algorithm's effectiveness in various situations, we evaluate the algorithm's performance by choosing varying parameters. More specifically, we compare the performance of the above algorithms under different choices of the total time slot T, and the maximum inter-update interval K that is allowed between two updates. We present the experimental results in Figure 2 and Figure 3

In Figure 2, we set $K = 30$ to observe the performance of different algorithms under different values of total time slots T. As shown in Figure 2, our proposed algorithm outperforms the conceivable alternatives under all choices of total time slots, achieving an expanded performance improvement from around 18.9% to 48.7%.

In Figure 3, we show how the performance of algorithms behavior when the maximum inter-update interval K changes, while setting and $T = 100$. We find that the CARA and CSA's performance are almost the same regardless of the change of K and the slight difference may be attributed to the randomness of the model, indicating that they may update the models over

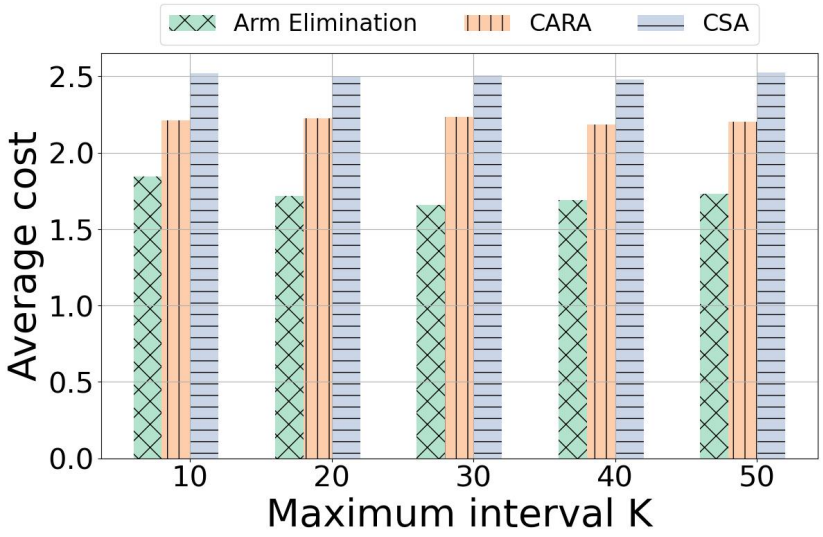

Fig. 3: The real-world scenario performances under different maximum intervals K.

frequently that fail to utilize the relaxed requirement of the maximum interval K Our algorithm, however, shows its ability to adapt to the change of K and achieves about 10.8% total cost deduction.

VI. Conclusion

In this paper, we study the problem of optimizing the update timing of a learning model to jointly minimize the model degradation cost and the model update case. We consider a realistic setting where the model degradation cost is stochastic and follows an unknown distribution. We need to learn the model degradation cost while making model update decisions. We propose an online algorithm using the idea of arm elimination and the feedback structure of our problem. Our proposed online algorithm achieves sublinear regret compared with the optimal solution that knows the degradation cost distribution. Through simulation, we show that under real-world traces, our approach consistently improves the performance against existing algorithms in various settings.

As for future work, it is interesting to further investigate the consecutive multi-arm bandit problem, including the fundamental performance limits. Another interesting direction is to apply our algorithm and ideas to broader scenarios, including the age-of-information optimization in communication networks, product age optimization in business, etc.

Appendix

A. Proof of Theorem 1

Proof. Consider an arbitrary online algorithm ALG, given its output (or any output realization of a randomized algorithm), we compute its inter-update times and denote it at k_i for the i-th update. Suppose there are U_T updates. Then, We can compute its total cost as follows

$$ALG(T) = \sum_{i=1}^{U_T} k_i \cdot \frac{\sum_{j=1}^{k_i} \tilde{g}(j) + C(k_i)}{k_i} + \beta_{T-\sum_{i=1}^{U_T} k_i}. \tag{16}$$

We have that

$$ALG(T) \geq \sum_{i=1}^{U_T} k_i \frac{\sum_{j=1}^{k^*} \tilde{g}(j) + C(k^*)}{k^*}. \tag{17}$$

Note that $T - \sum_{i=1}^{U_T} k_i \leq K$ as that the maximum un-update period must be no larger than K. Then we have

$$ALG(T) \geq (T-K) \cdot \frac{\sum_{j=1}^{k^*} \tilde{g}(j) + C(k^*)}{k^*}. \tag{18}$$

We can check that the solution defined in the theorem gives the following total cost

$$ALG^*(T) = \lfloor T/k^* \rfloor \cdot k^* \cdot \frac{\sum_{j=1}^{k^*} \tilde{g}(j) + C(k^*)}{k^*} + \beta_{T-\lfloor T/k^* \rfloor \cdot k^*}. \tag{19}$$

We have that

$$ALG^*(T) \leq T \cdot \frac{\sum_{j=1}^{k^*} \tilde{g}(j) + C(k^*)}{k^*} + K \cdot M. \tag{20}$$

Thus we have

$$ALG^*(T) - ALG(T) \tag{21}$$
$$\leq K \cdot \frac{\sum_{j=1}^{k^*} \tilde{g}(j) + C(k^*)}{k^*} + K \cdot M \tag{22}$$
$$\leq C \cdot K + (K+1) \cdot M \tag{23}$$

By taking the average over T and letting T go to infinity, we conclude the theorem. □

B. Proof of Lemma 2

Proof. According to our definitions and interpretation, it is not hard to check that,

$$\begin{aligned} &\Bigg|(C+M) \cdot \sum_{k=1}^{K} k \cdot \mu_k \cdot n_k(T) \\ &- \sum_{i=1}^{U_T} (\beta_{k_i} + C(k_i)) + \beta_{T-\sum_{i=1}^{U_T} k_i}\Bigg| \\ \leq &\beta_{T-\sum_{i=1}^{U_T} k_i} \leq K \cdot M, \end{aligned} \tag{24}$$

and

$$\begin{aligned} &\left|\left[\lfloor T/k^* \rfloor \cdot (\beta_{k^*} + C(k^*)) + \beta_{T-\lfloor T/k^* \rfloor \cdot k^*}\right] - T \cdot \mu_{k^*}\right| \\ \leq &C + M. \end{aligned} \tag{25}$$

Consequently, we have $\left|R(T) - \tilde{R}(T)\right| \leq C + (k+1)M$. □

C. Proof of Proposition 3

Proof. We now discuss the proof of Proposition 3. Our proof is adapted from [32] by further considering that in our case, pulling an arm k will take k slot and k times of cost to receive one feedback. We consider Δ a parameter to be determined. We consider the following $K+1$ problem instances. Let $\mu^{(i)} \in \mathbb{R}^K$ be the vector of the means of the arms in the i-th instance, $\forall i \in \{0, 1, \cdots K\}$, and $\mu_j^{(i)} = 1/2 + 1/2(1 - \mathbb{I}_{i=j})\Delta, \forall i, j \in [k]$, and $\mu_j^{(0)} = \Delta, \forall j \in [K]$. We consider that the cost of arm j under instance i follows a Bernoulli distribution with mean $\mu_j^{(i)}$. We let $\mathbb{E}_i[\cdot] = \mathbb{E}_{\mu^i}[\cdot] =$. We define R_i as the expected regard of an online algorithm under instance i. We define $n_k(T)$ as the number of arm k is pulled. We can show that

$$R_i = \sum_{k=1, k\neq i}^{K} k \cdot \mathbb{E}_i[n_k(T)]\Delta, \tag{26}$$

Also, we have that (which is adapted from Lemma A.1 from [32], noting that $n_i[T]$ is a random variable bounded by T/i and the KL divergence between Bernoulli($\mu_i^{(0)}$) and Bernoulli($\mu_i^{(i)}$) is bounded by $2\Delta^2$ when Δ is sufficiently small.)

$$\mathbb{E}_i[n_i(T)]] \leq \mathbb{E}_0[n_i(T)] + \frac{T}{i}\sqrt{\frac{1}{4}\mathbb{E}_0[n_i(T)]\Delta^2}, \tag{27}$$

Then, by Jensen's inequality and the identity that $\sum_{i=1}^{K} i \cdot \mathbb{E}_0[n_i(T)] = T$ and Cauchy–Schwarz inequality, we have

$$\sum_{i=1}^{K} i \cdot \mathbb{E}_i[n_i(T)] \leq \sum_{i=1}^{K} i \cdot \left(\mathbb{E}_0[n_i(T)] + \frac{T}{i}\sqrt{\frac{1}{4}\mathbb{E}_0[n_i(T)]\Delta^2}\right) \tag{28}$$
$$\leq T + \sum_{i=1}^{K} T\sqrt{\frac{1}{4}\mathbb{E}_0[n_i(T)]\Delta^2} \tag{29}$$
$$= T + \frac{1}{2}T\Delta\sum_{i=1}^{K}\sqrt{\frac{1}{i}i\mathbb{E}_0[n_i(T)]} \tag{30}$$
$$\leq T + \frac{1}{2}T\Delta\sqrt{\sum_{i=1}^{K}\left(\sqrt{\frac{1}{i}}\right)^2}\sqrt{\sum_{i=1}^{K}\left(\sqrt{i\mathbb{E}_0[n_i(T)]}\right)^2} \tag{31}$$
$$\leq T + \frac{1}{2}T\Delta\sqrt{T}\sqrt{\log K} \tag{32}$$

Then we have that

$$\sum_{i=1}^{K} R_i = \Delta \sum_{i=1}^{K} (T - i \cdot \mathbb{E}_i[n_i(T)]) \tag{33}$$

$$\geq \Delta \cdot (K \cdot T - T - \frac{1}{2} T\Delta\sqrt{T}\sqrt{\log K}) \tag{34}$$

$$\geq \frac{K^2}{8}\sqrt{\frac{T}{\log K}} \tag{35}$$

where the last equality is achieved by taken $\Delta = \frac{1}{2}\frac{K}{\sqrt{T \log K}}$ and noting that $K \geq 2$. We conclude that exist an $i \in [K]$ such that $R_i \geq \frac{1}{8} K\sqrt{T/\log K}$. □

D. Proof of Theorem 4

Proof. We conduct a more detailed proof to show a tighter upper bound for Algorithm 1. Suppose that the optimal arm is with index k^*, and we denote the difference between the expected cost of arm k and arm k^* as $\Delta_k = \mu_k - \mu_{k^*}$. We note that we have

$$\Delta_k \leq \Delta_{k+1}, \forall k^* + 1 \leq k \leq T - 1. \tag{36}$$

We denote $\bar{\mu}_k(t)$ as the latest updated value of $\bar{\mu}_k$ at slot t, which follows Line 17 of Algorithm 1. We define $m_k(t)$ as the number of updates of $\bar{\mu}_k$ before and at slot t. We consider a clean event defined as,

$$\mathcal{E} \triangleq \{|\bar{\mu}_k(t) - \mu_k| \leq r_k(t), \forall t, k\}, \text{where } r_k(t) \triangleq \sqrt{\frac{2\log T}{m_k(t)}}. \tag{37}$$

According to Hoeffding's inequality (and assuming $K \leq T$), we can show that

$$\Pr[\mathcal{E}] \geq 1 - 2T^{-2}. \tag{38}$$

We consider the case when then clean event $\mathcal{E}$ happens. We note that under event $\mathcal{E}$, the optimal arm k^* will always be in $\mathcal{A}$. Thus, according to our algorithm, arm $k < k^*$ will never be pulled, and in the following, we focus on $k \geq k^*$.

Recall $n_k(t)$ as the number of times arm k is pulled before or at slot t. We note that $m(t)$ records the total number of arm pulling. Thus, we have $m(t) = \sum_k n_k(t)$. We denote $\mathcal{A}(t)$ as the latest updated value of $\mathcal{A}$ at slot t, which follows Line 9 in Algorithm 1. We note that following Algorithm 1, we update the estimation for each arm in $\mathcal{A}$ each time an arm is pulled. Thus, for all the arms in $\mathcal{A}(t)$, the number of times we update their estimations is equal, and we have $m_k(t) = m(t)$.

We consider that three types of arms in $[k^*+1, K]$. First, suppose an arm k is eliminated at slot t, and arm i is the maximum arm in $\mathcal{A}(t)$. In such case, we have the total number of pulling arm k in the whole period, $n_k(T) = n_k(t)$. We also have $n_k(t) = 0, \forall k \leq i$. This is because, as arm i is not eliminated before time t, no arm with an index less than i would be selected before t according to our arm selection rule, i.e., Line 12. Also, as arm k is the maximum arm in $\mathcal{A}(t)$, we have that all arms with index larger than i have been eliminated, and thus, $n_i(T) = n_i(t), \forall i > k$. Combining these facts, we have the number of updates of $\bar{\mu}_k$,

$$m_k(T) = m(t) = \sum_{i=k}^{K} n_i(t) = \sum_{i=k}^{K} n_i(T). \tag{39}$$

We must have

$$m_k(T) \leq \frac{32 \cdot \log T}{\Delta_k^2} + 1. \tag{40}$$

Otherwise, we will have that

$$\bar{\mu}_k - \bar{\mu}_{k^*} \geq \mu_k - \mu_{k^*} - 2\sqrt{\frac{2\log T}{m(t) - 1}} \geq 2\sqrt{\frac{2\log T}{m(t) - 1}}, \tag{41}$$

and arm k will have been eliminated at the previous update. Thus, we have

$$\sum_{i=k}^{K} n_i(T) \leq \frac{32 \cdot \log T}{\Delta_k^2} + 1. \tag{42}$$

Second, if an arm k remains in $\mathcal{A}(T)$, we have the following equation similarly,

$$m_k(T) = m(t) = \sum_{i=k}^{K} n_i(t) = \sum_{i=k}^{K} n_i(T). \tag{43}$$

We must have

$$\sum_{i=k}^{K} n_i(T) = m_k(T) \leq \frac{32 \cdot \log T}{\Delta_k^2} \leq \frac{32 \cdot \log T}{\Delta_k^2} + 1. \tag{44}$$

Otherwise, arm k will be eliminated.

Third, if an arm is eliminated before it is the maximum arm in $\mathcal{A}(t)$. We have that $n_k(T) = 0$. Let $\hat{t}$ be the minimum k such that k is either an arm in the first situation or in the second case. We have that (noting $n_i(T) = 0, \forall i \in [k, \hat{k} - 1]$ and $\Delta_k \leq \Delta_{\hat{k}}$)

$$\sum_{i=k}^{K} n_i(T) = \sum_{i=\hat{k}}^{K} n_i(T) \leq \frac{32 \cdot \log T}{\Delta_{\hat{k}}^2} + 1 \leq \frac{32 \cdot \log T}{\Delta_k^2} + 1. \tag{45}$$

In summary, we have

$$\sum_{i=k}^{K} n_i(T) \leq \frac{32 \cdot \log T}{\Delta_k^2} + 1, \forall k \in [k^* + 1, K]. \tag{46}$$

In addition, noting that $\sum_{i=1}^{K} i \cdot n_i(T) = T$, we have, for any k,

$$\sum_{i=k}^{K} n_i(T) = \sum_{i=k}^{K} \frac{i}{i} n_i(T) \leq \sum_{i=k}^{K} \frac{i}{k} n_i(T) \leq \frac{T}{k}. \tag{47}$$

We now show an upper bound on the regret of the algorithm. We first have that, noting $\sum_{k=1}^{K} k \cdot n_k(T) = T$ and $n_k(T) = 0, \forall k < k^*$,

$$\tilde{R}(T)/(C + M) = \sum_{k=1}^{K} k \cdot \mu_k \cdot n_k(T) - T \cdot \mu_{k^*} = \sum_{k=k^*+1}^{K} k \cdot n_k(T) \cdot \Delta_k. \tag{48}$$

We can consider the LP of maximizing the right hand size of (48) subject to conditions (46) and condition (47) with decision variables $n_k(T), \forall k \in [k^* + 1, K]$ to provide an upper bound on $\tilde{R}(T)$. We denote the above LP as $\mathcal{P}$. We can further consider the following two LPs, the sum of which provides an upper bound to the previous LP and thus, to the regret of the algorithm. Specifically, we separate $k > k^*$ into two sets:

$$\mathcal{K}_1 := \{k > k^* | T/k \geq \frac{32 \log T}{\Delta_k^2} + 1\}, \text{ and } \mathcal{K}_2 := \{k > k^* | T/k < \frac{32 \log T}{\Delta_k^2} + 1\}. \tag{49}$$

We define the regret incurs by the arms in $\mathcal{K}_1$ and $\mathcal{K}_2$ as follows.

$$\tilde{R}_1(T)/(C + M) := \sum_{k \in \mathcal{K}_1} k \cdot n_k(T) \cdot \Delta_k, \tag{50}$$

and

$$\tilde{R}_2(T)/(C + M) := \sum_{k \in \mathcal{K}_2} k \cdot n_k(T) \cdot \Delta_k, \tag{51}$$

We consider two relaxed conditions to (46) and condition (47),

$$\sum_{i \geq k, i \in \mathcal{K}_2}^{K} n_i(T) \leq \frac{32 \cdot \log T}{\Delta_k^2} + 1, \forall k \in \mathcal{K}_1. \tag{52}$$

and

$$\sum_{i \geq k, i \in \mathcal{K}_2}^{K} n_i(T) \leq \frac{T}{k}, \forall k \in \mathcal{K}_2. \tag{53}$$

It is not hard to see that, all solutions satisfying (46) and condition (47) will also be feasible under (52) and (53).

We define the first LP as maximizing (50) subject to (52) with decision variables $n_k(T), \forall k \in \mathcal{K}_1$, denoted as $\mathcal{P}_1$. We define the second LP as maximizing (51) subject to (53) with decision variables $n_k(T), \forall k \in \mathcal{K}_2$, denoted as $\mathcal{P}_2$. It is then not hard to check that the sum of these two LP will provide an upper bound to the LP $\mathcal{P}$.

We now consider the first LP, $\mathcal{P}_1$. Noting that such both LP has a special structure that the weight on $n_k(T)$ increases in k and there all budget limits on subsequence of $n_k(T)$. We list the element in $\mathcal{K}_1$ in an increasing order, i.e., $k_1 \leq k_2 \cdots \leq k_{|\mathcal{K}_1|}$. We can easy show that an optimal solution to the LP is by setting (for ease of presentation, we ignore the notation of (T))

$$n_{k_i}^* = 32 \log T \cdot \left(\frac{1}{\Delta_{k_i}^2} - \frac{1}{\Delta_{k_{i+1}}^2} \right), \forall i < |\mathcal{K}_1|, \tag{54}$$

and

$$n^*_{k_{|\mathcal{K}_1|}} = 32\log T \cdot \frac{1}{\Delta^2_{k_{|\mathcal{K}|_1}}} + 1. \tag{55}$$

We have the following argument,

$$\begin{aligned}
\frac{\tilde{R}_1(T)}{(C+M)} &= \sum_{i=1}^{|\mathcal{K}_1|} k_i \cdot n_{k_i}(T) \cdot \Delta_{k_i} && (56)\\
&\leq \sum_{i=1}^{|\mathcal{K}_1|-1} k_i \cdot \Delta_{k_i} \cdot 32\log T \cdot \left(\frac{1}{\Delta^2_{k_i}} - \frac{1}{\Delta^2_{k_{i+1}}}\right) + k_{|\mathcal{K}_1|} \cdot \Delta_{k_{|\mathcal{K}_1|}} \cdot \left(32\log T \cdot \frac{1}{\Delta^2_{|\mathcal{K}_1|}} + 1\right) && (57)\\
&= 32\log T \cdot \left(\frac{1}{\Delta^2_{k_1}}(k_1)\cdot\Delta_{k_1} + \sum_{i=2}^{|\mathcal{K}_1|}\frac{1}{\Delta^2_{k_i}}(k_i\cdot\Delta_{k_i} - (k_{i-1})\cdot\Delta_{k_{i-1}})\right) + k_{|\mathcal{K}_1|}\cdot\Delta_{k_{|\mathcal{K}_1|}} && (58)\\
&\overset{(49)}{\leq} 32\log T \cdot \left(\sqrt{\frac{T\cdot k_1}{32\cdot\log T}} + \sum_{i=2}^{|\mathcal{K}_1|}\sqrt{\frac{T\cdot k_i}{32\cdot\log T}}\frac{1}{k_i\Delta_{k_i}}(k_i\Delta_{k_i} - k_{i-1}\Delta_{k_i-1})\right) + k_{|\mathcal{K}_1|}\cdot\Delta_{k_{|\mathcal{K}_1|}} && (59)\\
&\leq 32\log T \cdot \left(\sqrt{\frac{T\cdot K}{32\cdot\log T}} + \sqrt{\frac{T\cdot K}{32\cdot\log T}}\sum_{i=2}^{|\mathcal{K}_1|}\frac{1}{k_i\Delta_{k_i}}(k_i\Delta_{k_i} - k_{i-1}\Delta_{k_i-1})\right) + |\mathcal{K}_1|\cdot\Delta_{k_{|\mathcal{K}_1|}} && (60)\\
&\leq 32\log T \cdot \left(\sqrt{\frac{T\cdot K}{32\cdot\log T}} + \sqrt{\frac{T\cdot K}{32\cdot\log T}}\log\frac{k_{|\mathcal{K}_1|}\cdot\Delta_{k_{|\mathcal{K}_1|}}}{k_1\cdot\Delta_{k_1}}\right) + k_{|\mathcal{K}_1|}\cdot\Delta_{k_{|\mathcal{K}_1|}} && (61)\\
&\leq 32\log T \cdot \left(\sqrt{\frac{T\cdot K}{32\cdot\log T}} + \sqrt{\frac{T\cdot K}{32\cdot\log T}}\log\frac{K\cdot\Delta_K}{\Delta_{k_1}}\right) + K\cdot\Delta_K && (62)\\
&\leq \sqrt{32\cdot TK\log T}\left(1 + \log\frac{2K\sqrt{T}}{\sqrt{32\log T}}\right) + 2K && (63)\\
& && (64)
\end{aligned}$$

where the last equality is by noting that $\Delta_K \leq 2$ (as μ_K,$\mu_{k^*} \in [0,1]$).

Similarly, for the second LP, we arrange the terms in an increasing order such that $k_1 \leq k_2 \cdots \leq k_{|\mathcal{K}_1|}$. We have that the optima solution to the problem is by setting

$$n^*_{k_i} = T/k_i - T/k_{i+1}, \forall i < |\mathcal{K}_2|, \tag{65}$$

and

$$n^*_{k_{|\mathcal{K}_2|}} = T/k_{|\mathcal{K}|_2}. \tag{66}$$

We have that We have the following argument,

$$\begin{aligned}
\frac{\tilde{R}_2(T)}{(C+M)} &= \sum_{i=1}^{|\mathcal{K}_1|} k_i \cdot n_{k_i}(T) \cdot \Delta_{k_i} && (67)\\
&\leq \sum_{i=1}^{|\mathcal{K}_1|-1} k_i\cdot\Delta_{k_i}\cdot(T/k_i - T/k_{i+1}) + k_{|\mathcal{K}_2|}\cdot\Delta_{k_{|\mathcal{K}_2|}}\cdot\left(T/k_{|\mathcal{K}|_2}\right) && (68)\\
&\overset{(49)}{\leq} \sum_{i=1}^{|\mathcal{K}_1|-1}\sqrt{\frac{TK}{32\log T}}\cdot(1/k_i - 1/k_{i+1}) + \sqrt{\frac{TK}{32\log T}}\left(1/k_{|\mathcal{K}|_2}\right) && (69)\\
&\leq \sqrt{\frac{TK}{32\log T}} && (70)
\end{aligned}$$

By combining the results for $\mathcal{P}_1$ and $\mathcal{P}_\in$, we can conclude that

$$\tilde{R}(T)/(C+M) \leq \frac{\tilde{R}_1(T)}{(C+M)} + \frac{\tilde{R}_2(T)}{(C+M)} \leq \tilde{O}(\sqrt{KT}). \tag{71}$$

□

REFERENCES

[1] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian informatics journal*, vol. 16, no. 3, pp. 261–273, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2020.

[4] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications surveys & tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[5] D. Vela, A. Sharp, R. Zhang, T. Nguyen, A. Hoang, and O. S. Pianykh, "Temporal quality degradation in ai models," *Scientific Reports*, vol. 12, no. 1, p. 11654, 2022.

[6] H. Lee, S. Yoo, D. Lee, and J. Kim, "How important is periodic model update in recommender system?" in *Proceedings of ACM SIGIR*, 2023, p. 2661–2668.

[7] I. Žliobaitė, M. Budka, and F. Stahl, "Towards cost-sensitive adaptation: When is it worth updating your predictive model?" *Neurocomputing*, vol. 150, pp. 240–249, 2015.

[8] A. Mahadevan and M. Mathioudakis, "Cost-aware retraining for machine learning," *Knowledge-Based Systems*, vol. 293, p. 111610, 2024.

[9] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[10] Z. Gao, Y. Han, Z. Ren, and Z. Zhou, "Batched multi-armed bandits problem," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[11] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 151–159.

[12] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, "Combinatorial multi-armed bandit with general reward functions," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[13] S. Cayci, A. Eryilmaz, and R. Srikant, "Learning to control renewal processes with bandit feedback," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 2, Jun. 2019. [Online]. Available: https://doi.org/10.1145/3341617.3326158

[14] J. Diaz-de Arcaya, A. I. Torre-Bastida, G. Zárate, R. Miñón, and A. Almeida, "A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey," *ACM Comput. Surv.*, vol. 56, no. 4, oct 2023.

[15] I. Frías-Blanco, J. d. Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2015.

[16] B. Krawczyk, B. Pfahringer, and M. Woźniak, "Combining active learning with concept drift detection for data stream mining," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2239–2244.

[17] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[18] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 8, pp. 5362–5383, 2024.

[19] Y. Wu, E. Dobriban, and S. Davidson, "DeltaGrad: Rapid retraining of machine learning models," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 13–18 Jul 2020, pp. 10 355–10 366.

[20] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1749–1758.

[21] A. Slivkins *et al.*, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

[22] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial multi-armed bandit and its extension to probabilistically triggered arms," *Journal of Machine Learning Research*, vol. 17, no. 50, pp. 1–33, 2016.

[23] V. Perchet, P. Rigollet, S. Chassang, and E. Snowberg, "Batched bandit problems," *The Annals of Statistics*, vol. 44, no. 2, pp. 660–681, 2016.

[24] S. Cao, S. He, R. Jiang, J. Xu, and H. Yuan, "Best arm identification in batched multi-armed bandit problems," *arXiv preprint arXiv:2312.13875*, 2023.

[25] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks*. IEEE, 2011, pp. 350–358.

[26] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2731–2735.

[27] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[28] B. T. Bacinoglu, Y. Sun, E. Uysal, and V. Mutlu, "Optimal status updating with a finite-battery energy harvesting source," *Journal of Communications and Networks*, vol. 21, no. 3, pp. 280–294, 2019.

[29] V. Tripathi and E. Modiano, "An online learning approach to optimizing time-varying costs of aoi," in *Proceedings of ACM Mobihoc '21*, 2021, p. 241–250.

[30] ——, "A whittle index approach to minimizing functions of age of information," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 1160–1167.

[31] "Yelp open dataset," https://www.yelp.com/dataset, Yelp Inc., 2024.

[32] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.