Dynamic Planning and Learning under Recovering Rewards

David Simchi-Levi [1] **Zeyu Zheng** [2] **Feng Zhu** [1]

Abstract

Motivated by emerging applications such as live-streaming e-commerce, promotions and recommendations, we introduce a general class of multi-armed bandit problems that have the following two features: (i) the decision maker can pull and collect rewards from at most K out of N different arms in each time period; (ii) the expected reward of an arm immediately drops after it is pulled, and then non-parametrically recovers as the idle time increases. With the objective of maximizing expected cumulative rewards over T time periods, we propose, construct and prove performance guarantees for a class of "Purely Periodic Policies". For the offline problem when all model parameters are known, our proposed policy obtains an approximation ratio that is at the order of $1 - \mathcal{O}(1/\sqrt{K})$, which is asymptotically optimal when K grows to infinity. For the online problem when the model parameters are unknown and need to be learned, we design an Upper Confidence Bound (UCB) based policy that approximately has $\widetilde{\mathcal{O}}(N\sqrt{T})$ regret against the offline benchmark. Our framework and policy design may have the potential to be adapted into other offline planning and online learning applications with non-stationary and recovering rewards.

1. Introduction

In this paper, we introduce and solve a general class of multi-armed bandit (MAB) problems that have two unique features, which extends the class of classical MAB problems. First, in each time period, the decision maker can simultaneously pull and collect rewards from at most K arms out of N arms. Second, the expected reward of an arm immediately drops after it is pulled, and then gradually recovers if the arm is not pulled in the subsequent time periods. This class of problems are motivated by emerging applications such as live-streaming e-commerce, promotions and recommendations, and have the potential to include other applications with non-stationary and recovering rewards. We use the contexts of live-streaming e-commerce to briefly describe the problem background, before moving into the discussion of contributions, related literature, and specific problem settings.

Promotion in Live-streaming E-commerce: The rapidly growing live-streaming e-commerce is estimated to hit $100 billion in annual global sales in the year of 2020; see Greenwald (2020) and Kharif & Townsend (2020). A most common form of live-streaming e-commerce is session-based promotions held by a key opinion leader (KOL) on platforms such as Amazon Live and Taobao Live. In session-based promotions, a KOL holds sessions on consecutive days. The KOL chooses about $K = 50$ products to discuss their features, promote and sell during the session, one after another, often at deep discounts. The prepared inventory for the promoted products can often sell out in seconds. Many manufacturers pitch the KOL and create a large pool of products for the KOL to choose from, with an estimated ten to one selection ratio (Greenwald, 2020; Kharif & Townsend, 2020). Therefore, the entire pool of products can be of size $N = 500$. This emerging application of live-streaming e-commerce gives rise to the need for dynamic planning and learning with two unique features. First, in each time period, the decision maker selects $K = 50$ products (arms) to promote and sell (pull) out of a pool of $N = 500$. Second, if a product is promoted and on sale in some time period t with expected reward R, the expected reward of promoting and selling the same product again in time period $t + 1$ will drop and can be much smaller than R. On the other hand, as the idle time (number of time periods that a product is not promoted) increases, the expected reward of this product can increase. Because of this, even for the most popular product, a KOL will not promote and sell the product in every single session, but alternatively, select the product in every few other sessions.

Aside from the example of live-streaming e-commerce, there are a number of other applications that exhibit such rewards dropping and recovering phenomenon. For example, in the

[1]Institute for Data, Systems, and Society, Massachusetts Institute of Technology, Massachusetts, USA [2]Department of Industrial Engineering and Operations Research, University of California, Berkeley, USA. Correspondence to: David Simchi-Levi <dslevi@mit.edu>, Zeyu Zheng <zyzheng@berkeley.edu>, Feng Zhu <fengzhu@mit.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

recommendation domain, a customer may enjoy a particular product or service, but may temporarily get bored immediately after consuming the product or service. In this case, the reward of recommending the same product or service to the customer will temporarily drop after the most recent consumption, but may gradually recover as time passes by. Different products or services can have different dropping and recovering behavior, and it is of the decision maker's interest to strategically choose when and how often to recommend each product or service. For any given product, a resting time between each time it is recommended can be valuable, which intuitively offers a refreshing window for the customer.

1.1. Our Contributions

We summarize our contributions as follows.

1. **A General Recovering Model**: To characterize the dropping and recovering behavior of the rewards which is represented by a recovering function, we build a non-parametric bandit model with minimal assumptions on the recovery functions: monotonicity and boundedness. The recovery functions do not need to satisfy *any other* structural properties such as concavity. Our model relaxes assumptions and generalizes settings in previous work, thus allowing wider potential applications.

2. **An Offline Planning Framework**: Even if the recovery functions are fully known, the planning problem of optimally choosing K out of N arms in T time periods is computationally difficult. We construct an upper bound on the offline planning problem, which leads us to focus on a simple and intuitive class of policies called "Purely Periodic Policies". Surprisingly, the long-run offline approximation ratio (the ratio between the expected cumulative reward of the constructed policy and the optimal) only depends on K and not on other model parameters. This ratio, moreover, has a uniform lower bound $1/4$ and approaches 1 with a $\mathcal{O}(1/\sqrt{K})$ gap uniformly over $K \geq 1$. The framework of our policy design and analysis is novel, in the sense that we utilize power of 2 to bypass the difficulty of combinatorial planning, and address the trade-off between rounding error and scheduling error. Moreover, our policy design framework does not rely on standard assumptions and can potentially be adapted to various other applications.

3. **A Learning-while-Planning Algorithm**: We address the problem where we have no prior knowledge of the recovery functions, and must learn through online samples. Enlightened by the offline planning part, we propose an efficient online policy to reach a balance between planning and learning. The policy attains $\gamma_K(1-\epsilon)$ of the offline upper bound for arbitrary $\epsilon \in (0,1]$ with a $\widetilde{\mathcal{O}}(\sqrt{T})$ regret. The term ϵ appears only due to computational aspects, related to an FPTAS (Fully Polynomial-Time Approximation Scheme). Our design of the online learning algorithm exploits the sparse structure of the "Purely Periodic Policy" proposed in our offline planning framework, relates it to a generalization of the knapsack problem, and integrates it with upper confidence bounds, which may bring new insights to solve other online learning problems under recovering rewards.

1.2. Related Literature

Multi-armed Bandits: The Multi-armed Bandit (MAB) problem has aroused great interest in both academia and industry in the last decade. Since the pioneer work of Auer et al. (2002), researchers have developed many theoretically guaranteed algorithms based on the "optimism under uncertainty" principle. Readers could refer to Lattimore & Szepesvári (2019) and Slivkins et al. (2019) for a comprehensive discussion about this field. An important MAB variant is the combinatorial MAB problem, where in each time the player can pull a subset of arms instead of only one arm (see, e.g., Kveton et al. 2015). This better captures real-world applications, and we also adopt the setting.

Another variant of MAB arousing interest in recent years is non-stationary MAB, where the mean reward of each arm can change with time. Two main non-stationary models are restless bandits (Whittle, 1988) and rested bandits (Gittins, 1979). In restless bandits, the mean reward of an arm changes irrespective of the policy being used, while in rested bandits, the mean reward changes only when the arm is pulled by the policy. These two problems have been investigated through many different perspective, such as variation budget (Besbes et al., 2014), abrupt changes (Auer et al., 2019), rotting bandits (Levine et al., 2017), and applications in dynamic pricing (Keskin & Zeevi, 2017; Zhu & Zheng, 2020). We note that our setting is neither rested nor restless, as our reward distribution of an arm changes according to both the time period itself and whether the arm is selected by the policy or not. Our problem exhibits a recovering behavior: the mean reward of an arm first drops after a pulling, but will gradually recover as the idle time increases. Below we review the bandits literature with recovering rewards, which is most related to our work.

Bandits with Recovering Rewards: Recently there has been an increasing number of work studying bandits under different recovering environments. However, so far all the work only consider pulling one arm at a time. The work most relevant to ours is Kleinberg & Immorlica (2018). The authors develop a PTAS to obtain a long-run $(1-\epsilon)$-optimal scheduling for the offline problem of which the

time complexity is doubly exponential in $1/\epsilon$, and derive a $\widetilde{\mathcal{O}}\left(\frac{\sqrt{T}}{\epsilon^{\mathcal{O}(1/\epsilon)}}\right)$ regret for the online problem by adapting the offline result. Their analysis is heavily dependent on the assumption that the recovery function is non-decreasing and weakly concave on the whole $\mathbb{Z}_+$. We remove this somewhat restrictive assumption by only assuming the monotonicity, allowing more general behavior after a pull. In this case, the property of the randomized policy discussed in Kleinberg & Immorlica (2018) may no longer be valid. Further, we consider a setting more general than Kleinberg & Immorlica (2018): In each step we can pull K arms while Kleinberg & Immorlica (2018) allows only pulling one arm. It's unclear whether the techniques in Kleinberg & Immorlica (2018) can be adapted accordingly. Thus, we develop different methods and new techniques that extend to arbitrary K and get a $1 - O(1/\sqrt{K})$ approximation. Our solution is related with the periodic scheduling literature, which has been studied in some previous works such as Rothvoss (2009), Sgall et al. (2009) and Holte et al. (1989). We will elaborate on this point more throughout the discussion in Section 3.

A special case of our model is investigated in Basu et al. (2019), where an arm cannot be pulled for a while after it is pulled. The "sleeping time" of each arm is known a priori. The authors compete their online learning algorithm with the greedy policy which yields a long-run $1-1/e$ approximation guarantee. However, in our more general setting, the greedy policy may perform very bad compared to the true optimum. Also, in our online learning problem, we have no prior knowledge of the sleeping times. Cella & Cesa-Bianchi (2020) generalizes the setting in Basu et al. (2019) using non-parametric recovery. However, they assume that the recovery rate is the same among all arms, and only consider purely periodic policies where all selected arms are pulled at the same frequency with no constant ratio guarantees. We adopt a more generalized model and take a broader approach by allowing different recovery rates and different pulling frequencies for different arms, and yields a $1/4$ worst-case guarantee for the offline problem.

Another related work is Pike-Burke & Grunewalder (2019), where the expected reward functions are sampled from a Gaussian Process with known kernel. The authors compare their algorithm to a Bayesian d-step lookahead benchmark, which is the greedy algorithm optimizing the next d pulls given the decision maker's current situation. In comparison, our benchmark is concerned with the total reward of the whole time horizon T rather than a pre-fixed d. Some other related work include Mintz et al. (2020) and Yancey & Settles (2020). In Mintz et al. (2020), the recovery function is characterized via a parametric form, while the authors obtain a worst-case regret of $\widetilde{\mathcal{O}}(T^{\frac{2}{3}})$. Yancey & Settles (2020) considers a specific application for scheduling of reminder notifications. The authors demonstrate their policy via numerical experiments. We note that our problem is non-parametric in essence, and for the online problem we have a theoretical guarantee of $\widetilde{\mathcal{O}}(\sqrt{T})$ regret compared to our offline benchmark.

2. The Setup

In this section, we discuss the problem setting and the policy framework. We first present the model and assumptions in Section 2.1. We then point out that even when all the model parameters are known, the performance of a widely applied greedy policy can be bad. We then define the policy class of "Purely Periodic Policies" (PPP) in Section 2.2, and list the challenges to address.

2.1. The Model

There are N arms in total. In each time period $t = 1, 2, 3, \ldots$, the action is to simultaneously pull a subset of no more than $K \leq N$ arms. The revenue collected in each time period is the sum of the reward of each arm in the pulled subset. For each arm i, consider a set of non-negative scalars $\{R_i(d)\}_{d=0,1,2,\ldots}$ as the *recovery function* with respect to d. The reward of pulling arm i at time period t is a random variable with mean $R_i(t-t')$, where t' denotes the last time period before t when arm i is pulled. We let $R_i(0) = 0$ for completeness. The randomness in the reward of pulling an arm i at time t is assumed to be independent of all other sources of randomness in the system. The system runs from time period 1. To clearly define the initialization of the system at time period 1, one needs to specify when is the last time period that each arm is pulled before time period 1. There have been two common types of initialization. One is that the last time that each arm is pulled is $-\infty$; see Basu et al. (2019). The other is that the last time that each arm is pulled is time period 0, right before the start of the problem at time period 1; see Kleinberg & Immorlica (2018). Both types of initialization, as well as other types of initialization that are a mixture of the aforementioned two, do not in general affect the long-run results since their differences only impact the first pull for each arm. We adopt the second type of initialization. Next we describe and discuss assumptions on the recovery function.

Assumption 1 (Monotonicity). *For any $i \in [N]$, the recovery function $\{R_i(d)\}_{d\geq 0}$ is non-decreasing in d.*

The non-decreasing property of the expected reward sequence $\{R_i(d)\}_{d\geq 0}$ reflects the modeling feature that the longer the time for which an arm has been idle (not pulled), the larger the expected reward once the arm is pulled again. In fact, the range of expected reward sequences implied by Assumption 1 includes that implied by the assumptions made in the literature — concave recharging bandits (Klein-

berg & Immorlica, 2018), sleeping/blocking bandits (Basu et al., 2019), recovering bandits with identical recovery rate (Cella & Cesa-Bianchi, 2020), to name a few.

Assumption 2 (Boundedness). *The reward of pulling an arm in each time period is independent, and is bounded by a known constant $R_{\max}$ uniformly over all $i \in [N]$.*

The assumption of bounded random rewards is common in many applications such as promotion in live-streaming e-commerce and recommendation. An alternate but slightly restrictive assumption to Assumption 2 for the mean rewards $\{R_i(d)\}_{i\in[N],d\geq 0}$ is called "Finite-time Recovery".

Assumption 2′ (Finite-time Recovery). *For any $i \in [N]$, $\exists d_i^{\max} \geq 1$ such that $R_i(d) = R_i(d_i^{\max})$ for all $d \geq d_i^{\max}$.*

Assumption 2′ means that the mean reward of each arm recovers to a nominal level and stays there after a finite number of time periods (see also, e.g., Basu et al. 2019; Pike-Burke & Grunewalder 2019). We will see that this assumption is added in Section 3 primarily for computational concerns. All of our theoretical results and bounds do not depend on $\{d_i^{\max}\}$ that can be large in practice.

Before proceeding to the policy class, we give an example showing that the greedy policy, where we always select K arms with the highest expected payout in each time step, may have arbitrarily bad expected total reward in terms of the ratio with the optimal. As a comparison, in the problem settings of Kleinberg & Immorlica (2018) and Basu et al. (2019), the greedy policy in an analogous form, despite of being sub-optimal, has a guaranteed lower bound in terms of the ratio between the expected total reward achieved by the greedy policy and the optimal.

Example 1. *Let $N = 2$ and $K = 1$. Let $R_1(d) = r$ and $R_2(d) = \mathbb{1}_{d=1} + R \cdot \mathbb{1}_{d>1}$, where $R \gg 1 > r$. Then under the greedy policy, we will always pull arm 2, yielding a total reward of T. However, if we pull arm 1 and 2 in turn, then the total reward is lower bounded by $\frac{R+r}{2}(T-1) \gg T$ when $R \gg 1$. Therefore, the ratio between the expected total reward achieved by the greedy policy and the optimal can be arbitrarily small.*

2.2. Purely Periodic Policy

Throughout the paper, our policy design stays in a class of policies called *purely periodic policies* defined as follows.

Definition 1 (Purely Periodic Policy). *A policy is called "purely periodic", if for each $i \in [N]$, the policy assigns $t_i \in \mathbb{Z}$ and $d_i \in \mathbb{Z}_+ \cup \{+\infty\}$ such that (i) $t_i \in (-d_i, 0]$, and (ii) arm i is pulled at time $t_i + k \cdot d_i$ ($k \in \mathbb{Z}_+$) until the time horizon T is exhausted.*

We note that for a purely periodic policy, when $d_i = +\infty$, $t_i \leq 0$ for some i, the policy never pulls arm i throughout the whole time horizon. In the following, we use "PPP" as the brief notation of "purely periodic policy".

Definition 2 (K-PPP). *Let $K \in \mathbb{Z}_+$. A policy is called a K-PPP, if the policy is a purely periodic policy, and meanwhile in each time period $t \geq 1$, at most K arms are pulled at the same time. Equivalently speaking,*

$$\sup_{t\geq 1} \#\{i \in [N] : d_i < +\infty, t \equiv t_i \pmod{d_i}\} \leq K.$$

We note that with an overall number of N arms, each PPP is always an N-PPP. For the general case where $K \leq N$, the set of K-PPPs is a subset within the class of PPPs. There are two challenges for constructing a K-PPP that delivers superior performance. First, we need to determine the set of frequencies $\{1/d_i\}$, or equivalently, the set of periods $\{d_i\}$. Recall that the period for an arm can be set as infinity so that the arm is never pulled. Second, we need to pair each d_i with some t_i, as defined in Definition 2. The $\{d_i\}$'s must be selected carefully, or there may not exist such $\{t_i\}$ that no more than K arms can be pulled at the same time. We illustrate this point through the example below.

Example 2. *Let $N = 2$ and $K = 1$. Then there exists no 1-PPP such that $d_1 = 2$ and $d_2 = 3$, since 2 and 3 are co-prime numbers. No matter how we choose t_1 and t_2, there must exist a time period such that both arms are scheduled to be pulled together, violating the constraint $K = 1$. However, if $d_1 = 2$ and $d_2 = 4$, then letting $t_1 = -1$ and $t_2 = -2$ yields a feasible 1-PPP.*

3. The Offline Problem

In this section, we consider an offline version of the problem defined in Section 2.1. Specifically, in the offline problem, the sequence of expected rewards $\{R_i(d)\}_{d\geq 0}$ is known a priori. The objective is to identify a K-PPP to maximize the expected total reward over T time periods. The randomness in the system does not impact the policy design for the offline problem, because all the model parameters are known in the offline problems and there is no need to learn those parameters under uncertainties. We note that solving the offline problem with general policies is computationally intractable. The problem setting in Basu et al. (2019) is a special case of ours, and they discuss in Section 3 the computational intractability of their problem setting. In the supplementary material, we give a proof demonstrating that finding the long-run optimal policy for the offline problem within the class of K-PPP is NP-Hard.

3.1. Bounding the Objective: Why Purely Periodic?

The goal of this part is to give an upper bound on the optimal objective value of the offline problem. Meanwhile, we will give an interpretation on the advantages of PPP via the

analysis on the upper bound. We start our discussion from the rewards recovery functions $\{R_i(d)\}_{i\in[N],d\geq 0}$.

For given i, let $\{R_i^{\text{cc}}(d)\}_{d\geq 0}$ be the upper concave envelope of $\{R_i(d)\}_{d\geq 0}$. That is, R_i^{cc} is the point-wise minimum of all concave functions $F : \mathbb{Z}_+ \cup \{0\} \to \mathbb{R}$ that satisfy $F(d) \geq R_i(d)$ for all non-negative integer d. We define *supporting points* as the set of points $\{d \geq 0 : R_i(d) = R_i^{\text{cc}}(d)\}$. Then these supporting points can be characterized inductively as follows. $d_i^{(0)} = 0$. For any $k \geq 1$,

$$d_i^{(k)} = \min\left\{\arg\max_{d>d_i^{(k-1)}}\left\{\frac{R_i(d) - R_i(d_i^{(k-1)})}{d - d_i^{(k-1)}}\right\}\right\}.$$

When $\{R_i(d)\}$ is bounded, for any $k \geq 1$, we have

$$\lim_{d\to+\infty}\frac{R_i(d) - R_i(d_i^{(k-1)})}{d - d_i^{(k-1)}} = 0.$$

If

$$\max_{d>d_i^{(k-1)}}\left\{\frac{R_i(d) - R_i(d_i^{(k-1)})}{d - d_i^{(k-1)}}\right\} > 0,$$

then $d_i^{(k)} < +\infty$. Otherwise, $d_i^{(k)} = d_i^{(k-1)} + 1$. In both cases, $\{d_i^{(k)}\}_{k\geq 0}$ is an infinite sequence for any given $i \in [N]$. When Assumption 2′ holds, the supporting points of the concave envelope for arm i can be efficiently obtained by only examining $\{R_i(d)\}_{0\leq d\leq d_i^{\max}}$.

Let $F_{i,T}(x)$ be the value of the following problem.

$$\begin{aligned}\max_{s}\quad & \frac{1}{T}\sum_{j=1}^{J} R_i(s_j - s_{j-1}) \\ \text{s.t.}\quad & J \leq x\cdot T, \\ & 0 = s_0 < s_1 < \cdots < s_J \leq T.\end{aligned} \tag{1}$$

Intuitively, $F_{i,T}(x)$ is the optimal average reward of i given that we pull arm i no more than $x \cdot T$ times. Let $F_i(x) = \limsup_T F_{i,T}(x)$ be the point-wise long-run optimal average reward of pulling arm i under x. We have Lemma 1 that fully characterizes the nice shape of $F_i(x)$.

Lemma 1. *$F_i(x)$ is a piece-wise linear concave function on $[0, 1]$ that satisfies:*

- *The changing points are in $\{1/d_i^{(k)}\}_{k\geq 1}$.*
- *$F_i(1/d_i^{(k)}) = R_i(d_i^{(k)})/d_i^{(k)}$ for any $k \geq 1$. $F_i(x) = F_i(1/d_i^{(1)})$ for all $x \in [1/d_i^{(1)}, 1]$.*

Meanwhile, $F_i(x) \geq F_{i,T}(x), \forall i \in [N], T \geq 1, x \in [0, 1]$.

We are then ready to present the upper bound (2).

$$\begin{aligned}\max_{x\in[0,1]^N}\quad & T\sum_{i=1}^{N} F_i(x_i) \\ \text{s.t.}\quad & \sum_{i=1}^{N} x_i \leq K.\end{aligned} \tag{2}$$

Lemma 2. *The optimal objective value of (2) is an upper bound on that of the offline problem. Further, problem (2) has an optimal solution $\{x_i^*\}$, such that $x_i^* \leq 1/d_i^{(1)}$ ($\forall i \in [N]$), and for at least all $i \in [N]$ but one, $x_i^* \in \{0\} \cup \{1/d_i^{(k)}\}_{k\geq 1}$ holds.*

We give some remarks on Lemma 1 and 2. The two lemmas are generalizations of Lemma III.1 and III.2 in Kleinberg & Immorlica (2018) as we do not assume *any* concavity of recovery functions. The proofs of both lemmas only require the existence of *supporting points* as well as the monotonicity of the recovery functions. Moreover, we go beyond their results by observing that almost each component of the optimal solution is either 0 or $1/d$. In the proof of Lemma 2, we demonstrate that any feasible solution can be transformed into one that satisfies the property stated in Lemma 2 in $\mathcal{O}(N)$ time, while at the meantime the objective value is not decreased. This crucial property can be interpreted as in the optimal planning, we either pull arm i once every d times, or we do not pull it at all, which is exactly the form of PPP (see Definition 1). In the following discussion, we denote the objective value of (2) as $\text{UB}[N, K] \cdot T$.

Before ending this section, we give some remarks on the computation of (2). Under Assumption 2′, an optimal solution of the concave program (2) can be efficiently computed because (2) can be re-written as a linear program. Kleinberg & Immorlica (2018) considered the case where $K = 1$ and $d_i^{(k)} = k$ ($\forall k \geq 0$), and suggests that (2) admits an FTPAS even if Assumption 2′ is violated. It is thus interesting to investigate whether (2) admits a polynomial-time algorithm in general and we leave this to future work.

3.2. Constructing a K-PPP: Powerful Power of 2

In this section, we discuss how to take any $x \in (0, 1]^N$ and outputs a K-PPP through two steps. Here x does no need to satisfy the first constraint of (2), but the output schedule is always guaranteed to satisfy the hard constraint K. We emphasize that in contrast to previous work such as Kleinberg & Immorlica (2018), the pattern of the supporting points $\{d_i^{(k)}\}$ in Section 3.1 is not controllable in our general setting even for one single arm. R and R^{cc} only coincide on these supporting points, and there can be substantial gaps on non-supporting points. Also, different arms may have very different supporting points, complicating the design of pulling times. To circumvent the challenges, we develop the

rounding-scheduling framework in the following.

STEP 1: ROUNDING

We fix a set $\mathcal{D}[a] = \bigcup_{j=1}^{a} \mathcal{D}_j$, where a is a positive integer to be specified later, and where

$$\mathcal{D}_j = \{(2j-1) \times 2^{\ell}\}_{\ell \geq 0}.$$

We then round the solution $\{x_i\}$ to $\{1/d_i\}$ such that $d_i \in \mathcal{D}[a]$. More precisely, we let d_i be the element in $\mathcal{D}[a]$ closest to $1/x_i$ while no smaller than $1/x_i$, i.e.,

$$d_i = \min\{d \geq 1/x_i, d \in \mathcal{D}[a]\}.$$

We have the following lemma that lower bounds the average reward after rounding.

Lemma 3. *If $1/x_i \in \{d_i^{(k)}\}_{k \geq 1}$, then*

$$\frac{R_i(d_i)/d_i}{F_i(x_i)} \geq \frac{a}{a+1}.$$

STEP 2: SCHEDULING

In this step, we construct a K-PPP based on $\{d_i\}$ obtained in Step 1. To achieve this, we first relax the hard constraint K to some positive integer $K[a] \geq K$, which means $K[a]$ arms are allowed to be pulled at the same time. This can be fulfilled with the help of the following lemma.

Lemma 4. *Fix a positive integer j. Let $\mathcal{I}_j = \{i : d_i \in \mathcal{D}_j\} = \{i_1, i_2, \cdots, i_{|\mathcal{I}_j|}\}$ and $K_j = \sum_{i \in \mathcal{I}_j} 1/d_i$. Assume $d_{i_1} \leq d_{i_2} \leq \cdots d_{i_{|\mathcal{I}_j|}}$.*

- *If $K_j > 1$, then in $\mathcal{O}(|\mathcal{I}_j|)$ time, we can find at most $\lceil K_j \rceil$ disjoint sets $\{\mathcal{I}_{j_s}\}_s$ such that $\bigcup_s \mathcal{I}_{j_s} = \mathcal{I}_j$ and $\sum_{i \in \mathcal{I}_{j_s}} 1/d_i \leq 1$ $(\forall s)$.*
- *If $K_j \leq 1$, then in $\mathcal{O}\left(|\mathcal{I}_j| \log d_{i_{|\mathcal{I}_j|}}\right)$ time, we can specify a 1-PPP such that we pull each arm $i \in \mathcal{I}_j$ at frequency $1/d_i$.*

Now we elaborate on how to use Lemma 4 to achieve our goal. Fix any $j \in [a]$, we split $\mathcal{I}_j$ into several groups by using the first statement in Lemma 4, where in each group the sum of frequencies is within 1. Then for each group, we apply the second statement in Lemma 4 to construct a 1-PPP. Repeating over all $j \in [a]$ leads to the construction.

We have two observations from the procedure above. First, each arm i is included in exactly one group and is pulled with frequency $1/d_i$. Second, the number of arms pulled at the same time can be bounded by the total number of groups $K[a]$, which is

$$\begin{aligned} K[a] &\triangleq \sum_{j \in [a]} \left\lceil \sum_{i \in \mathcal{I}_j} 1/d_i \right\rceil \\ &< \sum_{j \in [a]} \sum_{i \in \mathcal{I}_j} 1/d_i + \sum_{j \in [a]} 1 \\ &\leq \sum_i x_i + a. \end{aligned}$$

Now that we have $K[a]$ groups where for each group we have constructed a feasible 1-PPP, and each arm appears in exactly one group, we choose K of these groups that obtain the largest long-run average reward. These K 1-PPPs together constitute a K-PPP.

We would like to give some brief remarks before proceeding. Our rounding method (Step 1) only requires that the recovery functions are non-decreasing, and our scheduling method (Step 2) does not require any additional assumptions. These may help generalize potential applications of our technique when dealing with more complicated settings.

COMBINING TOGETHER

Suppose x^* is an optimal solution of (2). We can always assume that x^* satisfies the property stated in Lemma 2. Combining the two steps above, we can see that, if $x_i^* \in \{1/d_i^{(k)}\}_{k \geq 1}$ $(\forall i \in [N])$, then

$$K[a] < \sum_{i \in [N]} x_i + a \leq K + a,$$

and thus $K[a] \leq K + a - 1$. By tuning a appropriately, the long-run approximation ratio of our policy is lower bounded by

$$\begin{aligned} &\max_a \left\{ \frac{\sum_{i \in [N]} F_i(1/d_i)}{\sum_{i \in [N]} F_i(x_i^*)} \cdot \frac{K}{K[a]} \right\} \\ \geq &\max_{a \in \mathbb{Z}_+} \frac{a}{a+1} \cdot \frac{K}{K+a-1}, \end{aligned}$$

In the following, we consider dealing with the most general case where not all components of the optimal solution are of form $1/d$. If there is an $x_{i_0}^* = 0$, then we just ignore arm i_0, and this does not hurt the performance. If there is an $x_{i_0}^* > 0$ such that $1/x_{i_0}^* \notin \{d_i^{(k)}\}_{k \geq 1}$, we first round $x_{i_0}^*$ to $\widetilde{x}_{i_0}^* = \min\left\{y \geq x_{i_0}^* : 1/y \in \{d_{i_0}^{(k)}\}_{k \geq 1}\right\}$ and then feed $\{x_i^*\}_{i \neq i_0} \cup \{\widetilde{x}_{i_0}^*\}$ to Step 1. Note that we still have $\widetilde{x}_{i_0}^* \leq 1/d_{i_0}^{(1)}$, and $F_{i_0}(\widetilde{x}_{i_0}^*) \geq F_{i_0}(x_{i_0}^*)$. Note again that there exists at most one such i_0. All the analysis in Step 1 and Step 2 are valid, except that the upper bound of $K[a]$ increases from $K + a - 1$ to $K + a$, which means we allow

pulling $K+a$ arms at the same time in Step 2. This is because we can only have the bound

$$\sum_{i\neq i_0} x_i^* + \widetilde{x}_{i_0}^* + a < K + 1 + a.$$

And the ratio becomes

$$\max_{a\in\mathbb{Z}_+} \frac{a}{a+1}\cdot\frac{K}{K+a}.$$

Algorithm 1 describes the complete paradigm for the offline problem. Theorem 1 provides the theoretical performance of Algorithm 1, showing that its long-run approximation ratio is lower bounded by γ_K.

Algorithm 1 Offline Purely Periodic Planning

Input: $\{R_i(d)\}_{i\in[N],d\geq 1}$
$a^* = \arg\max_a \frac{aK}{(a+1)(K+a)}$.
Initialize the supporting points $\{d_i^{(k)}\}_{i\in[N],k\geq 0}$.
Initialize $\{F_i(x)\}_{i\in[N],x\in[0,1]}$ using Lemma 1.
Solve (2) and obtain an optimal solution x^* that satisfies the property of Lemma 2. Exclude all i such that $x_i^*=0$.
if $\exists i_0\in[N]$ s.t. $1/x_{i_0}^* \notin \{d_{i_0}^{(k)}\}_{k\geq 1}\cup\{+\infty\}$ **then**
  $x_{i_0}^* \leftarrow \min\left\{y\geq x_{i_0}^* : 1/y\in\{d_{i_0}^{(k)}\}_{k\geq 1}\right\}$.
end if
Let $\mathcal{D}[a^*]=\bigcup_{j=1}^{a^*}\left\{(2j-1)\times 2^\ell\right\}_{\ell\geq 0}$.
for $i=1$ **to** N **do**
  $d_i \leftarrow \min\{d\geq 1/x_i^*, d\in\mathcal{D}[a^*]\}$.
end for
for $j=1$ **to** a^* **do**
  Let $\mathcal{I}_j=\{i: d_i\in\mathcal{D}_j\}$.
  Construct $\lceil\sum_{i\in\mathcal{I}_j} 1/d_i\rceil$ 1-PPPs using Lemma 4.
end for
$K[a^*]\leftarrow\sum_{j=1}^{a^*}\lceil\sum_{i\in\mathcal{I}_j}1/d_i\rceil$.
Select K of $K[a^*]$ 1-PPPs that attain the largest long-run average reward. These 1-PPPs then constitute a K-PPP.

Theorem 1. *For any $T\geq 1$, the total reward of the schedule returned by Algorithm 1 is lower bounded by*

$$\gamma_K\cdot \mathrm{UB}[N,K]\cdot T - \mathcal{O}(N),$$

where

$$\gamma_K = \max_{a\in\mathbb{Z}_+}\frac{a}{a+1}\cdot\frac{K}{K+a}.$$

We give some remarks on Theorem 1. First, the long-run approximation ratio γ_K is between our K-PPP policy and $\mathrm{UB}[N,K]$. Since $\mathrm{UB}[N,K]$ is always an upper bound on the original problem, our offline approximation ratio is over any policy. Second, γ_K satisfies 3 nice properties: (i) Independence with N and T; (ii) Uniformly lower bounded by $1/4$; (iii) Asymptotically approaches 1 with a gap of $\mathcal{O}(1/\sqrt{K})$. (iii) holds because the optimal a is either $\lfloor\sqrt{K}\rfloor$ or $\lceil\sqrt{K}\rceil$, and as a result, $1-\gamma_K=\mathcal{O}(1/\sqrt{K})$. Finally, the term $\mathcal{O}(N)$ appears because for each arm, we may incur loss at the beginning or the end of the whole time horizon.

Before proceeding, we would like to discuss the relations between our results and the periodic scheduling literature. Our work differs from previously studied periodic scheduling problems in that there is no hard constraint between consecutive occurrences of a job; instead, through our analysis of $\mathrm{UB}[N,K]$ and design of K-PPP, it turns out that a carefully designed PPP is able to achieve near-optimal behavior. Our rounding method is related with that in Sgall et al. (2009) where jobs are also scheduled in a purely periodic way, but has two main differences. First, the optimal solution to (2) may have a component of non-supporting point, which is not faced in typical periodic scheduling problems. This adds some technical difficulties in our setting. Second, since we allow pulling K arms, our technique allows rounding components to numbers in different $\mathcal{D}_j$. As a result, the ratio γ_K tends to 1 as K grows. This is in contrast to selecting one single $\mathcal{D}_j$ as in Sgall et al. (2009). Our scheduling framework has some relation with that in Holte et al. (1989). The class $\mathcal{D}_j$ in Lemma 4 actually belongs to C_M in Section 3 of Holte et al. (1989). Nevertheless, there are two different highlights in Lemma 4. First, the first bullet gives a more general picture of efficiently dealing with the situation when the sum of frequencies >1 during the scheduling process, which may happen even when $K=1$. Second, in the proof of the second bullet, we give a more computationally efficient scheduling method for our case, while the value of m in SimpleGreedy of [3] may appear quite large.

4. The Online Problem

In this section, we turn to address the online counterpart of the offline planning problem. In the online problem, unlike in Section 3, the recovery function $\{R_i(d)\}_{i,d}$ is not known a priori and should be learned from the sequential samples for any i. Our goal is to construct an online learning policy that achieves small regret compared to the offline result in Section 3. We discuss our results under only Assumption 1 and 2.

Broadly speaking, our policy design is built upon the "optimism under uncertainty" principle that has been successfully applied in many online learning problems. However, there are several main difficulties we have to confront in this recovering setting.

1. $\{R_i(d)\}$ is in essense non-parametric, and a complete sample of $R_i(d)$ requires at least $d+1$ time periods since we have to wait for the delay and plan for d time periods in the future. This planning issue is not faced

in classical stochastic bandit problems.

2. Due to sampling error, the upper confidence bounds of the estimation of $\{R_i(d)\}_{d\geq 1}$ may not be non-decreasing, i.e., violating Assumption 1. This impedes us from directly plugging the upper confidence bounds into $F_i(\cdot)$.

3. To make things more complicated, we have no prior knowledge of $\{d_i^{(k)}\}_{i\in[N],k\geq 1}$, which is crucial for estimating $\{F_i(\cdot)\}$. In previous work such as Kleinberg & Immorlica (2018) and Basu et al. (2019), $\{d_i^{(k)}\}_{i\in[N]}$ are always known a priori. Under Assumption 1 and 2, $\{R_i(d)\}_{d\geq 1}$ may appear to be an irregular shape, causing trouble for recovering $F_i(\cdot)$.

To address the first difficulty, we divide the whole time horizon into several phases of length ϕ. At the beginning of each phase j, we run an offline oracle to plan for a schedule in that phase. At the end of each phase, we update our estimation of $\{R_i(d)\}$, construct its corresponding set of UCBs $\{\hat{R}_{i,j}(d)\}$, and feed it to the offline oracle for planning the next phase $j+1$. We note that ϕ has to be carefully tuned because there is a trade-off: If ϕ is too large, we might explore too much on a sub-optimal schedule in some phase. While if ϕ is too small, we cannot estimate $R_i(d)$ for some larger d, causing us to get stuck in a sub-optimal schedule.

The UCBs are constructed as follows. Given a phase j, for each $i\in[N]$ and $d\geq 1$, we let $n_{i,j-1}(d)$ be the number of samples we have collected for $R_i(d)$ prior to phase j, and $\bar{R}_{i,j-1}(d)$ be the empirical mean of $R_i(d)$ prior to phase j. We let $n_{i,0}(d)=0$ and $\bar{R}_{i,0}(d)=0$. Then the upper confidence bound of $R_i(d)$ is

$$\hat{R}_{i,j}(d) \triangleq \min\left\{\bar{R}_{i,j-1}(d)+ R_{\max}\sqrt{\frac{2\log(KT)}{\max\{n_{i,j-1}(d),1\}}},\ R_{\max}\right\}. \tag{3}$$

To address the second and third difficulty, we need to re-interpret what Section 3 implies us on constructing a proper offline oracle. A crucial observation is as follows: The approximation ratio γ_K for the offline problem is achieved (even) if we (i) confine ourselves only to the class of PPP, and meanwhile (ii) restrict the possible periods to a sparse set $\mathcal{D}[a]$ with appropriate a. This observation leads to the offline oracle we describe as follows.

Fix a phase j and $a\in\mathbb{Z}_+$. Define $\mathcal{D}_\phi[a]=\{d: d\leq \phi/2, d\in\mathcal{D}[a]\}$ as the set of periods that are included in $\mathcal{D}[a]$ but at the meantime no larger than $\phi/2$. This is imposed such that in each phase all selected periods can be estimated at least once. Let $\{\hat{R}_{i,j}(d)\}_{i\in[N],d\in\mathcal{D}_\phi[a]}$ be some (estimated) UCBs over $\{R_i(d)\}_{i\in[N],d\in\mathcal{D}_\phi[a]}$. Let $x_{i,j,d}$ denote whether we pull arm i with period $d\in\mathcal{D}_\phi[a]$. Consider the following problem:

$$\begin{aligned}
\max_{x}\quad & \sum_{i\in[N]}\sum_{d\in\mathcal{D}_\phi[a]}\hat{R}_{i,j}(d)x_{i,j,d}/d \\
\text{s.t.}\quad & \sum_{i\in[N]}\sum_{d\in\mathcal{D}_\phi[a]} x_{i,j,d}/d\leq K+1, \\
& \sum_{d\in\mathcal{D}_\phi[a]} x_{i,j,d}\leq 1,\quad \forall i\in[N], \\
& x_{i,j,d}\in\{0,1\},\quad \forall i\in[N], d\in\mathcal{D}_\phi[a].
\end{aligned} \tag{4}$$

We give some explanations on (4). (4) can be regarded as a generalization of the knapsack problem that combines solving (2) in Section 3.1 with Step 1 in Section 3.2. We seek to maximize the long-run average reward with a frequency constraint, where $K+1$ is a frequency parameter. In the knapsack problem, each item has two choices: to be selected or not. While in (4), an arm has different versions indexed by $d\in\mathcal{D}_\phi[a]$, and we can choose at most one of the versions for each arm. Note that to implement (4), it's sufficient to collect samples for periods only in $\mathcal{D}_\phi[a]$ rather than the overall positive integers.

In the following, we analyze the properties of (4). Thanks to Assumption 2, we have Lemma 5 that relates the objective value of (4) to that of (2), given that the true rewards are dominated by the corresponding upper confidence bounds.

Lemma 5. *If $\hat{R}_{i,j}(d)\geq R_i(d)$ $(\forall i\in[N],\forall d\in\mathcal{D}_\phi[a])$, then the objective value of (4) is lower bounded by*

$$\frac{a}{a+1}\mathrm{UB}[N,K]-\frac{2NR_{\max}}{\phi}.$$

Next, we address the computation issue. (4) is an integer programming, and is in general not polynomial-solvable. Nevertheless, similar to the knapsack problem, (4) admits an efficient FPTAS.

Lemma 6. *We can obtain a $(1-\epsilon)$-optimal solution of (4) in $\mathcal{O}\left(\frac{N^3 a\log_2\phi}{\epsilon}\right)$ time.*

Once we obtain a $(1-\epsilon)$-optimal solution $\{x^\epsilon_{i,j,d}\}$ of (4), we let

$$1/d^\epsilon_{i,j}=\sum_{d\in\mathcal{D}_\phi[a]} x^\epsilon_{i,j,d}/d$$

be the frequency of pulling arm i. The final stage is to construct a K-PPP. Note that $\{d^\epsilon_{i,j}\}_{i\in[N]}$ itself does not necessarily constitute a K-PPP. We feed $\{d^\epsilon_{i,j}\}_{i\in[N]}$ into Step 2 described in Section 3.2 to obtain feasible periods $\{d_{i,j}\}$ and starting times $\{t_{i,j}\}$ which together constitute a K-PPP in phase j.

The remaining issue is to tune a appropriately. Letting

$a^* = \arg\max_a \frac{aK}{(a+1)(K+a)}$ yields

$$\begin{aligned}
&\sum_{i\in[N]} \hat{R}_{i,j}(d_{i,j})/d_{i,j} \\
&\geq \sum_{i\in[N]} \hat{R}_{i,j}(d^{\epsilon}_{i,j})/d^{\epsilon}_{i,j} \cdot \frac{K}{K+1+a^*-1} \\
&\geq \frac{a^*K(1-\epsilon)}{(a^*+1)(K+a^*)}\mathrm{UB}[N,K] - \frac{2NR_{\max}}{\phi} \\
&= \max_a \frac{aK(1-\epsilon)}{(a+1)(K+a)}\mathrm{UB}[N,K] - \frac{2NR_{\max}}{\phi},
\end{aligned}$$

where the second inequality holds from Lemma 5. Therefore, with suitable a, we can always guarantee that

$$\sum_{i\in[N]} \hat{R}_{i,j}(d_{i,j})/d_{i,j} \geq \gamma_K(1-\epsilon)\mathrm{UB}[N,K] - \frac{2NR_{\max}}{\phi}. \tag{5}$$

Algorithm 2 describes the complete procedure for the online learning problem. Theorem 2 states that compared to the offline benchmark, the total reward obtained from Algorithm 2 incurs a $\widetilde{\mathcal{O}}(\sqrt{T})$ regret.

Algorithm 2 Online Purely Periodic Learning

Input: $\phi, R_{\max}, \epsilon$
$a^* \leftarrow \arg\max_a \frac{aK}{(a+1)(K+a)}$.
$t \leftarrow 0, j \leftarrow 1$.
$\mathcal{D}_\phi[a^*] = \{d : d \leq \phi/2, d \in \mathcal{D}[a^*]\}$.
repeat
  Construct UCBs $\{\hat{R}_{i,j}(d)\}_{i\in[N],d\in\mathcal{D}_\phi[a^*]}$ as in (3).
  Solve (4) with the UCBs and obtain a $(1-\epsilon)$-optimal solution $\{x^{\epsilon}_{i,j,d}\}_{i\in[N],d\in\mathcal{D}_\phi[a^*]}$ by Lemma 6.
  Feed $\{\sum_{d\in\mathcal{D}_\phi[a^*]} x^{\epsilon}_{i,j,d}/d\}_{i\in[N]}$ into Step 2 in Section 3.2 and obtain a K-PPP.
  Run this K-PPP for $\min\{\phi, T-t\}$ time periods.
  $t \leftarrow t+\phi, j \leftarrow j+1$.
until $t \geq T$

Theorem 2. *Let $\phi = \Theta\left(\sqrt{\frac{T}{\log(K+1)}}\right)$ and $\epsilon = \Theta\left(T^{-\frac{1}{2}}\right)$, then the expected overall reward achieved by Algorithm 2 can be lower bounded by*

$$\gamma_K \cdot \mathrm{UB}[N,K] \cdot T - \widetilde{\mathcal{O}}\left(\max\{N, N^{\frac{1}{2}}K^{\frac{3}{4}}\}\sqrt{T}\right).$$

We leave the detailed logarithm terms in the regret bound to the supplementary material. We would like to give two remarks on Algorithm 2 and Theorem 2. First, the design of (4) serves as a crucial role in the proof of Theorem 2. The main insight of (4) is that it utilizes the offline design in Section 3 and only requires exploring the periods on a sparse set on $\mathbb{Z}_+$ for each arm. A naïve UCB may not suffice. Second, Algorithm 2 itself does not explicitly utilize the monotonic property, but its theoretical guarantee is implied by the offline result, which in turn relies on the monotonicity assumption. One advantage is that the algorithm is adaptive, in the sense that any theoretical improvement on the offline ratio guarantee of our framework can be directly transformed into an online result.

5. Conclusion

In this work, we consider the problem of dynamic planning and learning under a general bandit model of non-stationary recovering rewards. The non-stationarity stems from both the time elapsed and the policy itself. Solving the offline problem where all recovery functions are known is computationally hard, so we focus on a simple class of "Purely Periodic Policies". We develop a new framework for rounding and scheduling to obtain a policy with provable performance guarantee. The long-run approximation ratio is shown to be uniformly lower bounded by $1/4$ and asymptotically optimal with respect to the number of arms allowed to be pulled. We then show how to solve the online learning problem with $\widetilde{\mathcal{O}}(\sqrt{T})$ regret compared to the offline benchmark. We design our algorithm through a novel combination of our offline result, the knapsack problem, and upper confidence bounds, bypassing the difficulties of planning-while-learning under recovering rewards.

There is future work in line apart from improving the computational efficiency of our offline algorithm without Assumption 2′. An interesting direction we are working on is to examine whether the $\mathcal{O}(1/\sqrt{K})$ gap is um-improvable within the class of K-PPP. In the online problem, it's worth investigating whether we can obtain instance-dependent regret bounds or improved worst-case regret bounds under some additional assumptions, since assuming the recovery functions as in a completely non-parametric form and differ between arms will incur large learning cost. Also, we would also like to conduct experiments to see the practical performance of our policies for various application needs.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Auer, P., Gajane, P., and Ortner, R. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on Learning Theory*, pp. 138–158, 2019.

Basu, S., Sen, R., Sanghavi, S., and Shakkottai, S. Blocking bandits. In *Advances in Neural Information Processing Systems*, pp. 4784–4793, 2019.

Besbes, O., Gur, Y., and Zeevi, A. Stochastic multi-armed-

bandit problem with non-stationary rewards. In *Advances in neural information processing systems*, pp. 199–207, 2014.

Cella, L. and Cesa-Bianchi, N. Stochastic bandits with delay-dependent payoffs. In *International Conference on Artificial Intelligence and Statistics*, pp. 1168–1177, 2020.

Gittins, J. C. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

Greenwald, M. Live streaming e-commerce is the rage in china. is the u.s. next? *Forbes*, 2020. .

Holte, R., Mok, A., Rosier, L., Tulchinsky, I., and Varvel, D. The pinwheel: A real-time scheduling problem. In *Proceedings of the 22nd Hawaii International Conference of System Science*, pp. 693–702, 1989.

Keskin, N. B. and Zeevi, A. Chasing demand: Learning and earning in a changing environment. *Mathematics of Operations Research*, 42(2):277–307, 2017.

Kharif, O. and Townsend, M. Livestreams are the future of shopping in america. *Bloomberg*, 2020. .

Kleinberg, R. and Immorlica, N. Recharging bandits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 309–319. IEEE, 2018.

Kveton, B., Wen, Z., Ashkan, A., and Szepesvari, C. Tight regret bounds for stochastic combinatorial semi-bandits. In *Artificial Intelligence and Statistics*, pp. 535–543, 2015.

Lattimore, T. and Szepesvári, C. *Bandit algorithms*. Cambridge University Press (preprint), 2019.

Levine, N., Crammer, K., and Mannor, S. Rotting bandits. In *Advances in neural information processing systems*, pp. 3074–3083, 2017.

Mintz, Y., Aswani, A., Kaminsky, P., Flowers, E., and Fukuoka, Y. Nonstationary bandits with habituation and recovery dynamics. *Operations Research*, 68(5):1493–1516, 2020.

Pike-Burke, C. and Grunewalder, S. Recovering bandits. In *Advances in Neural Information Processing Systems*, pp. 14122–14131, 2019.

Rothvoss, T. On the computational complexity of periodic scheduling. Technical report, EPFL, 2009.

Sgall, J., Shachnai, H., and Tamir, T. Periodic scheduling with obligatory vacations. *Theoretical computer science*, 410(47-49):5112–5121, 2009.

Slivkins, A. et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

Whittle, P. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, pp. 287–298, 1988.

Yancey, K. P. and Settles, B. A sleeping, recovering bandit algorithm for optimizing recurring notifications. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3008–3016, 2020.

Zhu, F. and Zheng, Z. When demands evolve larger and noisier: Learning and earning in a growing environment. In *International Conference on Machine Learning*, pp. 11629–11638. PMLR, 2020.