Reinforcement Learning Algorithm for Reusable Resource Allocation with Time-varying Reward

Ziwei Wang, Jingtong Zhao, Yixuan Liu, Jie Song

March 31, 2024

Abstract

We explore a scenario where a platform must decide on the price and type of reusable resources for sequentially arriving customers. The product is rented for a random period, during which the platform also extracts rewards based on a prearranged agreement. The expected reward varies during the usage time, and the platform aims to maximize revenue over a finite horizon. Two primary challenges are encountered: the stochastic usage time introduces uncertainty, affecting product availability, and the platform lacks initial knowledge about reward and usage time distributions. In contrast to conventional online learning, where usage time distributions are parametric, our problem allows for unknown distribution types. To overcome these challenges, we formulate the problem as a Markov Decision Process and model the usage time distribution using a hazard rate. We first introduce an offline allocation method with a $1/2$ performance guarantee and then develop a reinforcement learning algorithm to address the cold start problem, accommodating non-parametric distributions and time-varying rewards. We further prove that the algorithm achieves sublinear regret against the offline benchmark. Numerical experiments on synthetic data as well as a real dataset from TikTok demonstrate the superiority and effectiveness of our method over other benchmark policies.

Keywords: reusable resources, online resource allocation, episodic reinforcement learning

1 Introduction

In various industries, resource reuse has become a cost-effective and efficient practice for service providers. This approach is particularly relevant in product lending, where rewards are directly tied to the impact generated during the lending period. Resource reuse is prevalent across diverse sectors, including business, healthcare, and advertising. For example, hospitals can reuse appointment slots, operation rooms, and beds, with rewards linked to patients' recovery rates (Zhalechian et al. [2023]). In cloud computing, customers pay for resources based on actual workloads (Chen et al. [2017]), while in advertising, touchpoints on websites or apps can be reused, with rewards varying based on customer engagement (Kumar et al. [2006]). Recently, live stream commerce has become popular as a way to promote products. The resource could be the broadcasting hours of a certain influencer, and the reward is related to the sales volume in the showroom. This practice enables service providers to operate efficiently, deliver sustainable services, and earn impactful rewards.

Motivated by the successful practice of reusable services, our paper focuses on a critical decision of the service provider regarding resource allocation and pricing. Customers arrive in an i.i.d. and

sequential manner to the service platform within a finite time horizon. The platform, equipped with various product types and pricing levels, determines the product type to offer and the corresponding rental price with each arrival. Upon receiving an offer, the customer decides whether to accept it, paying the rental price, or to leave the platform immediately if dissatisfied. Once a deal is made, the customer uses the product for a random duration, contributing additional revenue to the platform in each period based on a prearranged agreement. The reward could vary across the rental period. For example, in the context of a website offering display slots or a livestream of a influencer selling products, rewards are proportional to the number of clicks or sales volume over time, a common model in Pay-Per-Click or Pay-Per-Sales advertising. Initially, the platform lacks information about the distribution of rewards and usage time, necessitating a learning process to make estimations in a non-parametric manner. The primary objective of our study is to maximize revenue over a finite horizon based on a data-driven policy.

In this problem, the platform faces two primary challenges when trying to maximize revenue or social welfare. Firstly, it must address the issue of resource allocation within the constraints of limited resources and uncertain demand. While all resources are reusable and don't deplete, devising an optimal allocation policy to meet customer needs proves to be a complex task. For instance, in scenarios where rewards diminish swiftly with prolonged renting times, providers should strategically offer products with shorter average rental durations to capture higher rewards more frequently. Additionally, if subsequent customers are willing to pay a higher price, the platform should conserve the resource to prevent early depletion. Secondly, service providers often lack information about the distribution of renting times and corresponding rewards during the usage period, necessitating the gradual acquisition of knowledge about the true distribution and rewards. Failing to understand these parameters can result in revenue loss as the platform has no idea which product generates the highest benefit.

Many studies contribute to addressing the above challenges regarding perishable resources (Ferreira et al. [2018], Pan et al. [2020]). However, reusability has brought new technical challenges to online resource allocation. Compared to decisions made for perishable products, choices in the context of reusable resources have a longer-lasting influence into the future, as the returning time of the product might influence the feasibility of certain actions and affect the learning process. The learning parameter could further influence future decisions, introducing correlations to decision making and learning. Papers studying data driven decisions with reusable resources like Jia et al. [2022a], Jia et al. [2022b] mainly consider a pricing decision under a queuing model while in our problem, we need to make both pricing and resource allocation decisions and assume customers do not wait in line. Another stream of papers (Basu et al. [2019], Basu et al. [2021]) use blocking bandit to solve resource allocation problems. They assume each type of product has a single unit of resource and the expected rewards generated in each usage period are the same, while we allow multiple units of resource for each type and time-varying rewards. Existing studies predominantly leverage online learning techniques to design policies, often assuming the distribution type of usage time is known in advance. However, conversations with managers at Minglue, a data consulting company, reveal that empirical distributions, particularly in the form of hazard rates, are prevalent in practice. Unfortunately, the limitations of online learning methods become apparent in this scenario. Due to their inability to retain the state of each product, they struggle to distinguish products returned after different usage times and estimate hazard rates accurately. This state information is crucial for understanding product availability and predicting future rewards. More-

over, studies like Zhalechian et al. [2023] and Basu et al. [2021] in the realm of resource allocation often assume a one-period or steady reward during the usage period, allowing them to neglect state information such as the duration a product has been rented. In our problem, rewards are intricately linked to the usage time, adding complexity to the solution.

In order to tackle these challenges, we formulate the problem into a Markov Decision Process (MDP) and design a Reinforcement Learning (RL) algorithm to make data driven decisions. There are several benefits of using this formulation. First, the returning probability from a rented product can be gracefully encoded into the transition probability of the MDP. The usage time distribution is now naturally related to the hazard rate conditioned on the current state or usage time, with the failing event representing the returning of the product. This hazard rate can be estimated by counting the returning events and allows for empirical estimations of arbitrary distributions. Second, it can handle the correlation of learning and decision making process better. Now that the decision represents an action made under a given state, the impact of this decision into the future is essentially captured by the state transition probability, making the adaptive learning process easier to analyze with the help of the MDP structure.

Main contributions: Our main contributions are as follows.

1. We cast the problem of pricing and allocating reusable resources into an MDP framework, capturing crucial aspects of practical applications. Our designed algorithm employs linear function approximation, ensuring a provable and polynomial-time approach that achieves a competitive ratio of $1/2$ against the optimal expected reward under the full information setting. Additionally, we derive several key properties of the policy.

2. We address the cold start problem within an episodic reinforcement learning framework. Our approach involves using empirical estimations for both reward and usage time distributions, treating them as inputs to the algorithm in the full information setting. A significant modification is the introduction of an additional bonus term to the reward, encouraging exploration. The design of this bonus term aligns with the Optimization in the face of uncertainty (OPFU) rule. To expedite the convergence of the learning and earning process while ensuring optimal performance, we demonstrate that the bonus term can be efficiently reduced by providing a tightened bound on the opportunity cost of saving a product instead of renting it.

3. We provide a comprehensive analysis of the regret incurred by our reinforcement learning algorithm compared to the expected revenue obtained by the algorithm under the full information setting. Our results reveal that a sublinear regret of $\tilde{O}(\sqrt{T})$ can be achieved, validating the theoretical effectiveness of our algorithm. Moreover, we believe our proposed reinforcement learning framework can be extended to more online model-based problems with a dynamic programming based oracle under full information settings.

4. Lastly, we empirically validate the efficacy of our proposed algorithm using synthetic data as well as real data from TikTok in comparison to several benchmarks. Our algorithm provides an efficient way to rapidly navigate the cold start period and significantly enhance overall revenue.

The rest of the paper is organized as follows. We first give a literature review of existing papers related to our research in Section 2 and introduce the main model in Section 3. We give a detailed introduction of the methodology we use in Section 4, with a regret analysis carried out in Section 5. We provide numerical results in Section 6 and finally conclude the paper in Section 7.

2 Literature Review

Our research primarily revolves around two main themes. The first one pertains to reusable resource allocation, stemming from studies in revenue management, and it is closely tied to our proposed model. The second theme involves reinforcement learning with function approximations, aligning with the core methodology of our research.

2.1 Reusable Resource Allocation

In this section, we provide a comprehensive overview of reusable resource allocation in both the full information setting and the learning setting.

In the full information setting, where both reward and rental distributions are known, research in this field can be categorized into two main streams. The first stream focuses on achieving near-optimal performance when dealing with large-scale resources, while the second stream is concerned with developing algorithms with certain competitive ratios. For instance, in the context of admission control with a single reusable resource, Levi and Radovanović [2010] propose a Linear Program (LP) that serves as an upper bound on the long-run average reward. This LP guides their decision-making process, leading to the development of a class selection policy. In a more complex scenario involving multiple resources and service types with deterministic usage times, Lei and Jasin [2020] introduce a pricing policy that exhibits an average loss that decays as the system scales. Similar articles include Chen et al. [2017], Balseiro et al. [2023], and Xie et al. [2022]. Additionally, Zhang and Cheung [2022] explore a novel approach that employs multiple weight updates to address network resource allocation under various application settings.

Furthermore, many studies aim at achieving superior competitive ratios. For instance, Rusmevichientong et al. [2020] introduce an algorithm grounded in linear function approximation with backward induction. Their approach achieves a competitive ratio of $\frac{1}{2}$ when compared to the optimal total revenue. Subsequently, Rusmevichientong et al. [2023] explore the inclusion of advance bookings. Additionally, Baek and Ma [2022] generalize the problem to network reusable resource allocation, incorporating both reusable and nonreusable resources. By utilizing an upper-bound LP and applying backward induction, they achieve a competitive ratio of $\frac{1}{L+1}$, with L representing the maximum number of legs in a product. Inventory balancing methods have also been employed in various studies. For instance, Feng et al. [2019] and Feng et al. [2022] devise an online policy with inventory balancing. They demonstrate competitive ratios of $1 - \frac{1}{e}$ under fixed usage time and $(1 - \frac{1}{e})^2$ under stochastic i.i.d. time. Goyal et al. [2020] achieve an improved ratio by employing a LP-free linear system instead of the primal-dual approach. Additionally, Huo and Cheung [2023] consider scenarios where rewards are linearly related to usage time and usage time is deterministic, resulting in instance-dependent competitive ratios.

There is relatively little research focused on online reusable resource allocation while learning the rewards and rental distributions dynamically during the process. Some notable studies include Jia et al. [2022a] and Jia et al. [2022b], which investigate the learning of arrival and service rates that are linearly dependent on price in queuing systems. Zhong et al. [2022] also address this problem in a queuing context with multiple servers and introduces a "learn then schedule" policy. Another interesting line of research is exemplified by Basu et al. [2021] and Atsidakou et al. [2021], where they introduce the concept of a "blocking bandit." In this framework, arms that have been pulled become unavailable for

a deterministic or stochastic period, adding a unique dimension to the resource allocation problem. To combine learning and decision making together, all the prior research has relied on online learning methods to tackle this problem. As mentioned in the introduction, reinforcement learning techniques can provide greater flexibility and applicability to cope with the challenges for varying rewards and non-parametric distributions.

2.2 Reinforcement Learning with Function Approximation

Our research is closely related to reinforcement learning with function approximation for the value function, which represents the expected cumulative reward when starting from a specific state or state-action pair. This function becomes crucial when dealing with large state spaces where tabular methods like those in Jaksch et al. [2010], Jin et al. [2018], and Azar et al. [2017] cannot be directly applied. In resource allocation problems, the dimensionality of the problem grows exponentially with the number of resources, making it impracticable to solve using tabular methods.

One common approach is linear function approximation. Melo and Ribeiro [2007] analyze the convergence of Q-learning under this setting in an asymptotic manner. Subsequently, sample complexity is examined under the assumption of a generative model, as seen in Yang and Wang [2019]. However, the necessity of a simulator in this study poses certain restrictions, and Jin et al. [2020] introduce a method called LSVI-UCB. This method stands out as the first algorithm to achieve polynomial sample complexity in a linear setting without such stringent requirements. LSVI-UCB is a value-based method that employs backward induction, starting from the end of each episode, to derive value functions. Additionally, He et al. [2021] demonstrate that under specific conditions, regret can be further improved from $\tilde{O}(\sqrt{T})$ to $O(\log(T))$. Since then, linear function approximation has been widely explored in adversarial settings (Cai et al. [2020]), nonstationary settings (Zhou et al. [2020]), and even offline settings (Jin et al. [2021]). However, the above studies are restricted to linear MDP problems. Some studies like Zhou et al. [2021] and Zhou and Gu [2022] relax this assumption to linear mixture MDP and proposed UCRL-VTR+ method with $O(\sqrt{T})$ regret.

In our reusable resource allocation problem, the probability of each product being returned is largely independent. The transition probability requires the multiplication of the return probabilities for each rented product, rendering it unable to be represented as a linear combination of basis kernels. This brings challenges to applying existing methods. To address this challenge, we employ a modified kernel for linear function approximation, without the assumption of a linear MDP structure. Additionally, we design a bonus term for exploratory actions tailored to this modified kernel. We believe this approach can be extended to a wide range of resource allocation problems with linear function approximation derived from backward inductions.

3 Model Formulation

We now formally introduce our reusable resource allocation and pricing model. This section will begin with a detailed description of the model, followed by a MDP formulation of the problem.

3.1 Problem Description

Suppose a service provider offers N types of products indexed by $\mathcal{I} = \{i : i = 1, 2, \ldots, N\}$. The length of the planning horizon is T, which is divided into K episodes, each with length H satisfying $T = KH$. At the beginning of each episode k, the platform will update the initial capacity represented by a vector $C_k = (C_1^k, ..., C_N^k)^\top$, whose i^{th} component corresponds to the capacity of product i at the k^{th} episode. Besides, it will retrieve all the resources being used in the last episode. This assumption is grounded in empirical observations from the real world. For instance, companies in the cloud computing industry may periodically acquire new computing resources to increase their total capacity. They may also perform maintenance and repair on existing resources as needed. Similarly, websites can periodically enhance their content to create additional touchpoints for advertisements, expanding the opportunities for ad placements. The total inventory of all products is given by $C_k = \sum_{i=1}^N C_i^k$. The platform has a set of possible prices denoted by $\mathcal{B} = \{p_j : j \in \mathcal{J}\}$, where $\mathcal{J} = \{j : j = 1, ..., M\}$. Customers arrive sequentially at each time $t = 1, 2, \ldots, T$, and the provider can choose to offer a product and price from the set or simply reject the customer. Once a product i is borrowed, it can be used for at most $D_i + 1 \leq D_{max}$ steps.

The usage time of the product is a random variable, and the probability of returning the product depends on how long it has been used and the price offered. During the usage time, the provider may also benefit from the customer, such as through advertising revenue if the customer is a firm with advertising requests. The actual reward depends on the specific product and the usage time. The expected reward received at each time for product i rented for l periods is denoted as $\{r_{i,l} : l = 1, 2, ..., L\}$. Note that we assume the reward is unrelated to the price, as in most services, the reward mainly depends on the facility being rented or the specific influencer in advertising scenarios. However, our model can be easily extended to the case where the reward varies in conjunction with the price. The time-varying reward is employed especially in cases like hospital management, where smart devices can monitor the recovery of specific patients in real time, or under a live stream commerce, where the customers arrive to the influencer's channel and the system can show the real time sales volume.

The probability of a customer returning product i after l periods with price p_j is denoted as a probability mass $f_{i,j,l}$, and the rejection probability is $f_{i,j,0}$, with $\sum_{l=0}^{D_i+1} f_{i,j,l} = 1$. We assume that $f_{i,j,D_i+1} > 0, \forall i, j$, that is, there is a positive probability that customers will rent the product for the maximum time available. The acceptance rate, rental duration, and rewards earned are all initially unknown and require learning. The platform aims to design a learning and earning algorithm to collect as much revenue as possible within a finite time T.

3.2 MDP Formulation

The key elements of a MDP include the state and action space, the state transition probability and the value functions related to a specific policy. We start by defining the state space of our MDP. For each product i, we keep a state $s_i = \{s_{i,0}, s_{i,j,l} : l = 1, ..., D_i, j = 1, ..., M\}$. $s_{i,0}$ denotes the number of products available to be rented, and $s_{i,j,l}$ denotes the number of products rented for l periods at the upfront price p_j at the moment. We require $\sum_{j=1}^M \sum_{l=1}^{D_i} s_{i,j,l} + s_{i,0} \leq C_i^k$ for the k^{th} episode. Combining the state of each product, we get the state $\mathbf{s} = \{s_i : i = 1, 2, ..., N\}$. All the possible states form the state space $\mathcal{S}_k = \{\mathbf{s} : s_{i,0}, s_{i,j,l} \geq 0 \text{ and } \sum_{j=1}^M \sum_{l=1}^{D_i} s_{i,j,l} + s_{i,0} \leq C_i^k\}$.

At each time t, the platform's action a_t is selected from the action space $\mathcal{A} = \{(i,j) : i \in \mathcal{I}, j \in \mathcal{J}\} \cup \{(0,0)\}$. Choosing $(0,0)$ corresponds to offering a null product indexed by 0 with a price of 0, which has no rewards and infinite capacity. The pair (i,j) determines the product that will be offered and the price at which it will be sold. If the product is unavailable, meaning that $s_{i,0} = 0$, then the platform will reject this customer. Otherwise, the customer will decide whether to accept it with probability $1 - f_{i,j,0}$. Assuming the platform can acquire the full state information at each time step, we can observe how long each product has been rented. Denote the random variable $Y_{i,j}$ as the total usage time for a product i with price p_j. The conditional probability of returning the product after exactly l periods given that the product has already been used for $0 \leq l \leq D_i + 1$ periods is:

$$q_{i,j,l} = P(Y_{i,j} = l | Y_{i,j} \geq l) = \frac{f_{i,j,l}}{\sum_{d=l}^{D_i+1} f_{i,j,d}} \tag{1}$$

Note that when $l = 0$, $q_{i,j,0}$ denotes the probability that the customer does not accept the offer, which equals $f_{i,j,0}$, and when $l = D_i + 1$, $q_{i,j,D_i+1} = 1$. The conditional probability that the product is not returned after l periods is, therefore, $1 - q_{i,j,l}$. This conditional probability can be regarded as the hazard rate, providing a nonparametric representation of the usage distribution.

If we denote the state before taking an action (the pre-decision state) at the beginning of the next step as $\tilde{s}'_i$, and assume that the transitions of states for the products are independent, then the transition probability between states can be denoted as:

$$P(\tilde{s}'_i | s_i) = \prod_{j=1}^{M} \prod_{l=1}^{D_i - 1} \binom{s_{i,j,l}}{\tilde{s}'_{i,j,l+1}} (1 - q_{i,j,l+1})^{\tilde{s}'_{i,j,l+1}} q_{i,j,l+1}^{s_{i,j,l} - \tilde{s}'_{i,j,l+1}} \tag{2}$$

This is because the number of products (i,j) that have already been used for $l \geq 1$ periods and will not be returned in the next step follows a binomial distribution with parameters $s_{i,j,l}$ and $1 - q_{i,j,l+1}$. Therefore, the probability of having $\tilde{s}'_{i,j,l+1}$ products that will have been used for $l+1$ periods is $\binom{s_{i,j,l}}{\tilde{s}'_{i,j,l+1}} (1 - q_{i,j,l+1})^{\tilde{s}'_{i,j,l+1}} q_{i,j,l+1}^{s_{i,j,l} - \tilde{s}'_{i,j,l+1}}$. After taking an action $a = (i', j')$, the resulting post-decision state $\mathbf{s}'_i$ satisfies:

$$s'_i = \begin{cases} \tilde{s}'_i + 1_{\{s_{i,0} > 0\}} (\mathbf{e}_{i,j',1} - \mathbf{e}_{i,0}) & \text{if } i = i' \text{ and } Y_{i,j'} > 1 \\ \tilde{s}'_i & \text{o.w.} \end{cases} \tag{3}$$

Recall that $Y_{i,j'}$ is the random variable representing the usage time, which follows the distribution $\{f_{i,j',l} : l = 1, ..., D_i + 1\}$. If we choose to reject the request, then the post-decision state is simply $s'_i = \tilde{s}'_i$. The notation $\mathbf{e}_{i,j',1}$ refers to a unit vector that has its coordinates aligned with the state vector s_i. Specifically, the $(i, j', 1)$ coordinate of the unit vector is equal to 1, while all other coordinates are equal to 0.

We define a policy for the k^{th} episode π_k as a function that maps $\mathcal{S}_k \times [H]$ to the action space $\mathcal{A}$. The optimal policy is denoted as π_k^*. This is the policy that maximizes the expected reward in the k^{th} episode. For each episode k, the total reward collected at state $\mathbf{s}$, action a, and step h is denoted as $R^{h,k}(\mathbf{s}, a)$. The reward from rented products is comprised of two parts as shown below:

$$\{R_{i,l,m}^{h,k} : i \in \mathcal{I}, l \in [2, D_i + 1], m \in [\sum_{j=1}^{M} s_{i,j,l-1}]\} \cup \{R_{i,1}^{h,k} + p_j : a = (i,j)\}. \tag{4}$$

The first part $R_{i,l,m}^{h,k}$ represents the random reward of a product i rented for l periods. These rewards are generated by the products in state $s_{i,j,l-1}$, as only the products that have already been rented for $l-1$ periods and remain being rented can generate this reward. The index m is the m^{th} reward generated by a product in such a state and there are in total $\sum_{j=1}^{M} s_{i,j,l-1}$ such products. The second part $R_{i,1}^{h,k} + p_j$ represents the random reward of a product i rented for exactly 1 period. The only way of generating this reward is by offering the type i product at the j^{th} price given that the customer accepts the product.

The rewards are sub-Gaussian random variables and without loss of generality, we assume they are in the interval $[0, 1]$ with mean $\{r_{i,l}\}_{i \in \mathcal{I}, l \in [D_i+1]}$. The expected reward at time step h at the k^{th} episode is denoted as $r(\mathbf{s}, a) := E[R^{h,k}(\mathbf{s}, a)]$, representing the expectation of summing up all the rewards in (4). For an arbitrary policy π, we define the value function and action-value function at the k^{th} episode as follows:

$$V_h^{\pi,k}(\mathbf{s}) = E_\pi \left[\sum_{t=h}^{H} r(\mathbf{s}_t, a_t) | \mathbf{s}_h = \mathbf{s} \right] \tag{5}$$

$$Q_h^{\pi,k}(\mathbf{s}, a) = E_\pi \left[\sum_{t=h}^{H} r(\mathbf{s}_t, a_t) | \mathbf{s}_h = \mathbf{s}, a_h = a \right] \tag{6}$$

The notation $E_\pi[\cdot]$ here means the expectation of the random trajectory of the state-action sequence.

4 Methodology

In this section, we will first present an approximate solution to the offline problem under the full information setting. This serves as a crucial oracle for the online setting with unknown parameters. Subsequently, we provide a comprehensive description of our online policy with reinforcement learning. We use the terms 'offline' and 'full information setting' interchangeably. Similarly, we use 'online' to refer to the bandit feedback setting, where the reward and usage time distributions are unknown.

4.1 Full Information Setting

We commence with the offline scenario where all parameters are known. In this case, the main challenge arises from the "curse of dimensionality" problem when solving the previously defined MDP. This challenge will be addressed through Approximate Dynamic Programming. Note that this section mainly borrows ideas from Rusmevichientong et al. [2020] and is intended to serve as a benchmark for the upcoming reinforcement learning algorithm.

The optimal value function of our problem is denoted as $V_h^k(\mathbf{s}_h)$ for the k^{th} episode at time step h.

Given the current state $\mathbf{s}_h$, the Bellman equation is

$$V_h^k(\mathbf{s}_h) = \sum_{i=1}^{N}\sum_{l=1}^{D_i} r_{i,l+1}(\sum_{j=1}^{M}(\mathbf{s}_h)_{i,j,l}) + \max_{(i,j)\in\mathcal{A}}\Big\{1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-q_{i,j,0})\big\{(p_j + r_{i,1} + (1-q_{i,j,1})E[V_{h+1}^k(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})] + q_{i,j,1}E[V_{h+1}^k(\tilde{\mathbf{S}}_{h+1}))]\big\} + (1 - 1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-q_{i,j,0}))E[V_{h+1}^k(\tilde{\mathbf{S}}_{h+1})]\Big\} \tag{7}$$

The first line represents the immediate reward received at step h. Note that each of the products in state $s_{i,j,l}$ has already been rented for l periods and remains being rented, so it will generate a reward of $r_{i,l+1}$ in the current step. The second and third line represents the maximum expected reward in the subsequent steps. The notation $\tilde{\mathbf{S}}_{h+1}$ is the pre-decision state, which is a random variable following the distribution in (2). Specifically, the second line is the case where product i is offered and the customer accepts the offer and will use it for more than one step. This means one unit of product in state $s_{i,0}$ will change to $s_{i,j,1}$. The first term $q_{i,j,1}E[V_{h+1}^k(\tilde{\mathbf{S}}_{h+1})]$ in the third line is the case where it is used for exactly one period, so the state in the next step remains unchanged. The second term in the third line is the case where either the customer rejects the offer or the product is unavailable. We can also rewrite (7) to get:

$$V_h^k(\mathbf{s}_h) = \sum_{i=1}^{N}\sum_{l=1}^{D_i} r_{i,l+1}(\sum_{j=1}^{M}(\mathbf{s}_h)_{i,j,l}) + E[V_{h+1}^k(\tilde{\mathbf{S}}_{h+1})] + \max_{(i,j)\in\mathcal{A}}\Big\{1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-q_{i,j,0})\big\{(p_j + r_{i,1} - (1-q_{i,j,1})\{E[V_{h+1}^k(\tilde{\mathbf{S}}_{h+1}) - V_{h+1}^k(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})]\}\Big\} \tag{8}$$

In this adjusted form (8), the term $V_{h+1}^k(\tilde{\mathbf{S}}_{h+1}) - V_{h+1}^k(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})$ can be viewed as the opportunity cost of replacing an inventory item with a sale to a customer at a price of p_j. If we know the true value of this term for each state, then this problem can be easily solved by selecting the best (i,j) pair that maximize the second line of (8).

A major challenge stems from the extensive state space, which scales as $O(C_k^{D_{max}})$ and experiences exponential growth with the maximum usage time, denoted as D_{max}. As a result, directly computing the value function through recursive iterations on the Bellman equation becomes problematic. To address the "curse of dimensionality," inspired by Rusmevichientong et al. [2020], we employ a linear value function for approximation. In the following, we eliminate the index k for brevity as the operations are the same for each episode. Specifically, the linear approximation is defined as

$$\hat{V}_h(\mathbf{s}) = \sum_{i=1}^{N}(w_{i,0}^h s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^h s_{i,j,l}) \tag{9}$$

for all $h = 1, ..., H$ in an episode, with $\hat{V}_{H+1}(\mathbf{s}) = 0, \forall \mathbf{s}$. The weight parameters $w_{i,j,l}^h$ represent the value of renting one unit of resource i at price j for a duration of l periods, while $w_{i,0}^h$ signifies the value of one unit of resource i in the inventory.

The procedure of computing the weight parameters is given in Algorithm 1. At the beginning of each episode, we are given the initial capacity. Since we are considering the offline case in this section, the

Algorithm 1: Offline Scheduling Algorithm

input : Total capacity $C = (C_1, C_2, ..., C_N)^\top$, total horizon H, pairing reward $r_{i,l}$, transition dynamics $q_{i,j,l}$.

Initialize $w_{i,0}^{H+1} = 0, w_{i,j,l}^{H+1} = 0, \forall i \in \mathcal{I}, j \in \mathcal{J}, l \in [D_i]$ and $w_{i,j,D_i+1}^{h} = 0, \forall h \in [H]$;

for $h = H, ..., 1$ **do**

Compute the optimal product and price to offer, take any

$$(\hat{i}_h, \hat{j}_h) \in \text{argmax}_{i \in \mathcal{I} \cup \{0\}, j \in \mathcal{J} \cup \{0\}} (1 - q_{i,j,0})(p_j + r_{i,1} - (1 - q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1})) \tag{10}$$

for $i \in \mathcal{I}$ **do**

Compute the weight parameters for step h

$$w_{i,0}^{h} = w_{i,0}^{h+1} + \max\{\frac{1}{C_i} 1_{\{i=\hat{i}_h\}} (1 - q_{i,\hat{j}_h,0})(p_{\hat{j}_h} + r_{i,1} - (1 - q_{i,\hat{j}_h,1})(w_{i,0}^{h+1} - w_{i,\hat{j}_h,1}^{h+1})), 0\} \tag{11}$$

for $l = 1, ..., D_i$ *and* $j = 1, ..., M$ **do**

$$w_{i,j,l}^{h} = r_{i,l+1} + q_{i,j,l+1} w_{i,0}^{h+1} + (1 - q_{i,j,l+1}) w_{i,j,l+1}^{h+1} \tag{12}$$

end

end

end

output: All the marginal value parameters

mean rewards and transition dynamics are assumed known. We set $w^{H+1} = 0$ and then use backward induction to get the weights for $h = H, ..., 1$.

In each step, (10) calculates the optimal action by utilizing the Bellman equation (8). However, instead of using the opportunity cost directly, it replaces it with linear function approximation $w_{i,0}^{h+1} - w_{i,j,1}^{h+1}$. (11) and (12) are then responsible for updating the weight parameters based on this optimal action. In (11), the second term represents an estimate of the revenue gained by adding one more unit of resource to the inventory. To ensure accuracy, this term is normalized by the total available resource C_i. This normalization step is critical in the development of the competitive ratio outlined in Proposition 4.2. When updating $w_{i,0}^{h}$, we consider only the positive part in (11). This choice reflects the platform's ability to reject a request if none of the allocation options can generate a positive revenue when compared to reserving the product.

As we can see from Algorithm 1, the computational complexity is $O(NMD_{max}T)$, and is irrelevant to the capacity $C = \{C_i : i \in \mathcal{I}\}$. We denote the static policy obtained from this algorithm by $\hat{\pi}_h$, which is only related to the step but not the state. After the weight parameters are computed, the opportunity cost can be represented by $\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1}) = w_{i,0}^{h+1} - w_{i,j,1}^{h+1}$. We are unable to directly employ the static policy $\hat{\pi}_h$ due to its disregard for product availability. Therefore, we develop a greedy algorithm to facilitate real-time recommendations. According to the greedy algorithm, we

choose the product and the price as follows:

$$(i^L, j^L) = \underset{i\in\mathcal{I}\cup\{0\}, j\in\mathcal{J}\cup\{0\}}{\operatorname{argmax}} \{1_{\{s_{i,0}>0\}}(1-q_{i,j,0})[p_j + r_{i,1} - (1-q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1})]\} \tag{13}$$

This greedy policy is denoted as π^L, and $\pi^L(\mathbf{s}, h) = (i^L, j^L)$ if $1_{\{s_{i,0}>0\}}(1-q_{i,j,0})[p_j + r_{i,1} - (1-q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1})] > 0$. Otherwise, we reject the request which is equivalent to taking the action $(0, 0)$.

We proceed by establishing several properties of the linear approximation. We commence with Proposition 4.1, which asserts that the weights exhibit a decreasing trend over time. This observation is a direct consequence of the backward reduction illustrated in (11) and (12).

Proposition 4.1. *(Monotonicity)* $w_{i,0}^h \geq w_{i,0}^{h+1}$ *and* $w_{i,j,l}^h \geq w_{i,j,l}^{h+1}$ *for all* $h \in [H]$.

Furthermore, similar to the proof presented in Rusmevichientong et al. [2020], even in the presence of varying rewards and service prices, we can deduce a performance bound for the linear function approximation. This is demonstrated in Proposition 4.2:

Proposition 4.2. *The linear value approximation* $\hat{V}_1(\mathbf{s}) \geq \frac{1}{2}V_1(\mathbf{s})$ *for all* $\mathbf{s} \in \mathcal{S}$

The following proposition shows that the greedy policy π^L can guarantee that the expected revenue is no less than $\hat{V}_1(\mathbf{s})$. From now on, we denote the expected revenue obtained by the greedy policy as $U_h(\mathbf{s}) := V_h^{\pi^L}(\mathbf{s})$ and $\mathcal{U}_h(\mathbf{s}, a) := Q_h^{\pi^L}(\mathbf{s}, a)$ for brevity, and we have:

Proposition 4.3. *The expected revenue under the greedy policy satisfies* $U_h(\mathbf{s}) \geq \hat{V}_h(\mathbf{s})$ *for all* $h \in [H]$*, implying that* $U_1(\mathbf{s}) \geq \hat{V}_1(\mathbf{s})$.

By combining Proposition 4.2 and 4.3, we can get $U_1(\mathbf{s}) \geq \frac{1}{2}V_1(\mathbf{s})$. This means our greedy policy can achieve at least $1/2$ of the optimal revenue. What's more, we can prove that the value function is bounded by an $O(H)$ term. Denote $p_{max} := \max_{j\in\mathcal{J}} p_j$, $\Lambda_i := \max\{\frac{1}{C_i}(1 + p_{max}), 1\}$, then we have:

Proposition 4.4. *(Bound on the value functions) For each episode with capacity* $C = (C_1, ...C_N)^\top$*,* $\hat{V}_h(\mathbf{s}) \leq H(\sum_{i\in\mathcal{I}} \Lambda_i C_i)$*, and the real optimal value function* $V_h(\mathbf{s}) \leq 2H \sum_{i\in\mathcal{I}} \Lambda_i C_i$.

This bound does not require the knowledge of reward and usage time distributions, and is useful in the design and analysis of the learning algorithm in the next section. In the following, we will refer to the upper bound of the value functions in Proposition 4.4 as

$$B_k(H) := 2H \sum_{i=1}^{N} \Lambda_i C_i^k$$

4.2 Online Case with Reinforcement Learning

In this section, we begin by introducing some preliminary concepts to contextualize our problem within the reinforcement learning framework. Subsequently, we outline the process of estimating unknown parameters. Finally, we present the core algorithm of our paper, which incrementally learns these unknown parameters while making real-time decisions. This is facilitated by an Upper Confidence Bound (UCB)-based reinforcement learning algorithm under bandit feedback.

4.2.1 Preliminaries

We now elucidate how to interpret the value functions in the full information setting from the perspective of the transition operator. This not only provides novel insights into the design of linear function approximation but also offers a concise and convenient expression for the value functions. For brevity, we will first define the following operators that will be useful in the following and we omit the episode index k here:

Definition 4.1. *For any function $f : \mathcal{S} \to \mathbb{R}$, define the transition operator $\mathbb{P}_h$ at step h as:*

$$\mathbb{P}_h f(\mathbf{s}, a) := E[f(\mathbf{s}_{h+1})|\mathbf{s}_h = \mathbf{s}, a_h = a], \tag{14}$$

and the action-value function can then be represented by

$$Q_h^\pi(\mathbf{s}, a) = (r + \mathbb{P}_h V_{h+1}^\pi)(\mathbf{s}, a), \quad V_h^\pi(\mathbf{s}) = Q_h^\pi(\mathbf{s}, \pi_h(\mathbf{s})) \tag{15}$$

Now we return to the linear approximation we designed in the previous section. When employing this linear approximation, we do not utilize the actual transition kernel $P(\cdot|\mathbf{s}, a)$, instead, we employ a modified transition kernel that disregards the chosen product's availability. More precisely, while the pre-action transition kernel remains the same as the actual transition kernel (2), the post-action transition involves substituting the indicator function $1_{s_{i,0}>0}$ with $s_{i,0}/C_i^k$ in (3). This is the crucial element that renders the value function linear with respect to the state. The related modified transition operator for the linear approximation function is denoted as $\tilde{\mathbb{P}}_h$, and we have:

Proposition 4.5.

$$\hat{V}_h(\mathbf{s}) = (r + \tilde{\mathbb{P}}_h \hat{V}_{h+1})(\mathbf{s}, \hat{\pi}_h) \tag{16}$$

where $\hat{\pi}_h$ is the static policy of step h computed by Algorithm 1.

In fact, with the introduction of this modified transition kernel, we not only transform the value function into a linear representation, but we also manage to express the action-value function in a linear form, as shown in Proposition 4.6. Denote $D_{tot} = \sum_{i\in\mathcal{I}}(D_i)$ and $d = (N + MD_{tot} + NM)$, and we have:

Proposition 4.6. *For the linear approximation with corresponding MDP $(\mathcal{S}, \mathcal{A}, H, \tilde{\mathbb{P}}, r)$, there exists a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ and a weight vector $\boldsymbol{\mu}_h \in \mathbb{R}^d$, such that the action-value function $\hat{Q}_h(\mathbf{s}, a)$ can be expressed in a linear form $\boldsymbol{\mu}_h^\top \phi(\mathbf{s}, a)$ for all $\mathbf{s} \in \mathcal{S}, a \in \mathcal{A}$.*

Hence, this modified transition kernel significantly reduces the need for recursive computation of value functions for each state and action. It provides an approximation with a performance guarantee, even though the original problem does not adhere to the linear MDP property, a requirement in many existing studies.

Revisiting the derivation of the greedy policy within the framework of this operator, the essence of our greedy policy is to maximize an intermediate action-value function $\hat{\mathcal{U}}_h(\mathbf{s}, a)$. This intermediate function aims to estimate the expected future reward based on the linear approximation of the value function and can be viewed as performing a one-step rollout, as in Bertsekas [2021]. To state this formally:

Proposition 4.7. *The greedy action is based on maximizing an intermediate action-value function, which can be expressed as:*

$$\hat{\mathcal{U}}_h(\mathbf{s}, a) = r_h(\mathbf{s}, a) + \mathbb{P}_h \hat{V}_{h+1}(\mathbf{s}, a) \quad \forall \mathbf{s} \in \mathcal{S}, a \in \mathcal{A} \tag{17}$$

Denote $\hat{U}_h(\cdot) = \max_a \hat{\mathcal{U}}_h(\cdot, a)$ *as the related intermediate value function, then* $\hat{U}_h(\cdot) \leq U_h(\cdot)$ *for all* $\mathbf{s} \in \mathcal{S}, h \in [H]$*, where* $U_h(\cdot)$ *is the real value function based on the greedy policy.*

4.2.2 Estimation

Now we show how to estimate the unknown parameters according to the data collected during the process. In the online setting, the platform only knows $\mathcal{S}_k, \mathcal{A}, H$ and T, but does not know the rewards $\{r_{i,l}\}_{l=1}^{D_i+1}$ and the usage time distributions $\{q_{i,j,l}\}_{l=0}^{D_i}$ for all $i = 1, ..., N$, $j = 1, ..., M$. At each step, the platform can observe the current state $\mathbf{s}$, take an action a, and receive full information feedback from all the rewards at that step. However, the platform can only estimate the usage time distributions based on observations of whether the product is returned exactly after l periods. At the beginning of each episode k, we can observe the previous $k-1$ trajectories $\mathcal{F}_{k-1} := \{(\mathbf{s}_1^\tau, a_1^\tau), R^{1,\tau}, ..., (\mathbf{s}_H^\tau, a_H^\tau), R^{H,\tau}\}_{\tau=1}^{k-1}$.

We define the counting parameter $N_{i,j,l}^k$ as the number of times a type i product is rented at price j for l periods before episode k . This parameter is for estimating the transition probability $q_{i,j,l}$. We also define the counting parameter $\tilde{N}_{i,l}^k$ as the number of times a reward with mean $r_{i,l}$ occurs. We denote the event that a customer rejects the offer as $\mathcal{E}_h^k$. The parameters are then defined as:

$$N_{i,j,0}^k = \sum_{\tau=1}^{k-1} \sum_{h=1}^{H} 1_{\{a_h^\tau = (i,j)\}} \quad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{18}$$

$$N_{i,j,1}^k = \sum_{\tau=1}^{k-1} \sum_{h=1}^{H-1} [1_{\{a_h^\tau = (i,j), \neg \mathcal{E}_h^\tau\}}] \quad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{19}$$

$$N_{i,j,l}^k = \sum_{\tau=1}^{k-1} \sum_{h=1}^{H-1} (\mathbf{s}_h^\tau)_{i,j,l-1} \quad \forall i \in \mathcal{I}, j \in \mathcal{J}, l = 2, ..., D_i \tag{20}$$

$$\tilde{N}_{i,1}^k = \sum_{j \in \mathcal{J}} N_{i,j,1}^k \tag{21}$$

$$\tilde{N}_{i,l}^k = \sum_{\tau=1}^{k-1} \sum_{h=1}^{H} \sum_{j=1}^{M} (\mathbf{s}_h^\tau)_{i,j,l-1} \quad \forall i \in \mathcal{I}, l = 2, ..., D_i + 1 \tag{22}$$

The corresponding empirical estimators by taking the mean of all outcomes are as follows, for all

$i \in \mathcal{I}, j \in \mathcal{J}$:

$$\hat{q}^k_{i,j,0} = \frac{1}{N^k_{i,j,0}} \sum_{\tau=1}^{k-1} \sum_{h=1}^{H} 1_{\{a^\tau_h=(i,j), \mathcal{E}^\tau_h\}} \tag{23}$$

$$\hat{q}^k_{i,j,1} = \frac{1}{N^k_{i,j,1}} \sum_{\tau=1}^{k-1} \sum_{h=1}^{H-1} 1_{\{a^\tau_h=(i,j), \neg\mathcal{E}^\tau_h, (\mathbf{s}^\tau_{h+1})_{i,j,1}=0\}} \tag{24}$$

$$\hat{q}^k_{i,j,l} = 1 - \frac{1}{N^k_{i,j,l}} \Big(\sum_{\tau=1}^{k-1} \sum_{h=1}^{H-1} (\mathbf{s}^\tau_{h+1})_{i,j,l} \Big) \quad l = 2, ..., D_i \tag{25}$$

$$\hat{r}^k_{i,1} = \frac{1}{\tilde{N}^k_{i,1}} \sum_{\tau=1}^{k-1} \sum_{h=1}^{H-1} \sum_{j=1}^{M} [1_{\{a^\tau_h=(i,j), \neg\mathcal{E}^\tau_h\}} R^{\tau,h}_{i,1}] \tag{26}$$

$$\hat{r}^k_{i,l} = \frac{1}{\tilde{N}^k_{i,l}} \sum_{\tau=1}^{k-1} \sum_{h=1}^{H} \sum_{m=1}^{\sum_{j=1}^{M} (\mathbf{s}^\tau_h)_{i,j,l-1}} R^{\tau,h}_{i,l,m} \quad l = 2, ..., D_i + 1 \tag{27}$$

We now explain the estimators. $\hat{q}^k_{i,j,l}$ is an estimator of the probability of returning the product after l periods, denoted as $q_{i,j,l}$. Similarly, $\hat{r}^k_{i,l}$ is an estimator of the mean reward $r_{i,l}$. The rejection probability $q_{i,j,0}$ is estimated by counting the times an offer (i, j) is recommended and the times the arriving customer rejects it, denoted as the event $\mathcal{E}^\tau_h$. The estimation of the parameter $q_{i,j,1}$ relies on the frequency with which the offer (i, j) is accepted and the product is returned precisely after a single period, resulting in no product in the subsequent state $s'_{i,j,1}$. The corresponding estimators are shown in (24). For the remaining transition estimators, we count the number of times a product with attribute (i, j) has been rented for $l - 1$ periods for each l. Only those that are not returned can transition from state $s_{i,j,l-1}$ to the next state $s'_{i,j,l}$ with a probability of $1-q_{i,j,l}$, where $l \geq 2$, and the corresponding estimator is shown in (25). Similarly, we take the average of all the rewards collected from a type i product rented for l period as the estimator of $r_{i,l}$, where there are $\sum_{j=1}^{M} s_{i,j,l-1}$ products that can generate reward with mean $r_{i,l}$ when the current state is $\mathbf{s}$. The estimators are shown in (26) and (27).

Next, we proceed to define the confidence radius for the above estimators with parameter δ. For the transition parameters $q_{i,j,l}$, denote its radius at the k^{th} episode as

$$rad_k(q_{i,j,l}) := 2\sqrt{\frac{\log(2D_{max}MNT^2/\delta)}{\max\{1, N^k_{i,j,l}\}}} \tag{28}$$

Similarly, for the reward $r_{i,l}$, denote its radius at the k^{th} episode as

$$rad_k(r_{i,l}) := 2\sqrt{\frac{\log(2D_{max}NT^2/\delta)}{\max\{1, \tilde{N}^k_{i,l}\}}} \tag{29}$$

These radii help to define confidence intervals for the unknown parameters and capture the uncertainty level of the current estimations, which is a key step in designing UCB type algorithms. When a product i is used for only a small number of times, the counting number $N^k_{i,j,l}$ and $\tilde{N}^k_{i,l}$ would also be small, leading to a larger radius that encourages further exploration of the product.

4.2.3 Main Algorithm

Building upon the offline algorithm, we now formally introduce our reinforcement learning algorithm with bandit feedback. For the online scenario, we treat the offline algorithm as an oracle, utilizing the estimated rewards and transition kernels as inputs. To foster exploration, we develop a UCB-based reinforcement learning algorithm that strikes a balance between exploration and exploitation.

To be more specific, suppose we estimate the rewards and transitional probabilities by the empirical estimators mentioned above, denoted as $\hat{r}_h^k$ and $\hat{\mathbb{P}}_h^k$. The estimated modified transition kernel is represented as $\tilde{\mathbb{P}}_h'^k$. The estimated linear action-value function is defined as:

$$\tilde{Q}_h^k(\mathbf{s}, a) = \min\{\hat{r}_h^k(\mathbf{s}, a) + b_h^k(\mathbf{s}, a) + \tilde{\mathbb{P}}_h'^k \tilde{V}_{h+1}(\mathbf{s}, \tilde{\pi}_h(\mathbf{s})), B_k(H)\} \tag{30}$$

The additional term $b_h^k(\mathbf{s}, a)$ serves as a bonus function associated with the action, intended to promote exploration. Its detailed explanation will follow shortly. This bonus function is directly tied to the confidence radii mentioned earlier and serves to bound potential losses in the value function due to uncertainties in unknown parameters. Given that this additional term may be arbitrarily large initially, we constrain the value function by the upper bound $B_k(H)$ proposed in Proposition 4.4. The notation $\tilde{\pi}_h(\cdot)$ represents the policy that always maximizes $\tilde{Q}_h^k(\cdot, a)$, i.e.

$$\tilde{V}_h^k(\cdot) = \tilde{Q}_h^k(\cdot, \tilde{\pi}_h(\cdot)) \tag{31}$$

Similar to the full information case, in the online scenario, the greedy action is also based on maximizing the intermediate action-value function, whose transition probability is based on the estimated operator $\hat{\mathbb{P}}_h^k$ and the future expected reward is given by the linear function approximation $\tilde{V}_h^k(\cdot)$. The intermediate function is denoted as:

$$\tilde{\mathcal{U}}_h^k(\mathbf{s}, a) = \min\{\hat{r}_h^k(\mathbf{s}, a) + b_h^k(\mathbf{s}, a) + \hat{\mathbb{P}}_h^k \tilde{V}_{h+1}^k(\mathbf{s}, a), B_k(H)\} \tag{32}$$

Again, similar to the full information setting, denote $\bar{\mathcal{U}}_h^k(\cdot, \cdot)$ as the online version of the greedy action-value function, which follows the greedy policy $\tilde{\pi}^L(\cdot, h)$ provided by the intermediate function. We have:

$$\bar{\mathcal{U}}_h^k(\mathbf{s}, a) = \min\{\hat{r}_h^k(\mathbf{s}, a) + b_h^k(\mathbf{s}, a) + \hat{\mathbb{P}}_h^k \bar{U}_{h+1}^k(\mathbf{s}, a), B_k(H)\} \tag{33}$$

The value function is denoted as $\bar{U}_h^k(\mathbf{s}) = \bar{\mathcal{U}}_h^k(\mathbf{s}, \tilde{\pi}^L(\mathbf{s}, h))$, where the policy $\tilde{\pi}^L$ is derived by maximizing the intermediate function. Details regarding the online version of resource allocation and the formulation of the bonus function will be provided later. We summarize all the value functions and action value functions mentioned above in Table 1. The relationships among these four rows are as follows: For both the full information and learning setting, the optimal value function represents the best performance as given by the Bellman equation (7). To overcome the curse of dimensionality, a linear function is devised to approximate the optimal one. The intermediate function then determines the policy $\tilde{\pi}^L$ in our algorithm based on the linear function. It conducts a one-step rollout by replacing the opportunity cost in (8) with the subtraction of weight parameters in the linear function. The greedy function reveals the actual expected performance under this policy $\tilde{\pi}^L$. Following the definition of the

Bellman equation, the true value function only needs to substitute the expected future reward approximated by linear function approximation $\tilde{V}_{h+1}$ twith the real expected reward denoted by the function $\bar{U}_{h+1}$.

Table 1: List of Value Functions

Functions	Full Information Setting	Learning Setting
Optimal	$Q_h^*(\mathbf{s},a) = (r + \mathbb{P}_h V_{h+1}^*)(\mathbf{s},a)$ $V_h^*(\mathbf{s}) = \max_a Q_h^*(\mathbf{s},a)$	$Q_h^{*,k}(\mathbf{s},a) = \min\{(\hat{r}_h^k + b_h^k + \hat{\mathbb{P}}_h^k V_{h+1}^{*,k})(\mathbf{s},a), B_k(H)\}$ $V_h^{*,k}(\mathbf{s}) = \max_a Q_h^{*,k}(\mathbf{s},a)$
Linear	$\hat{Q}_h(\mathbf{s},a) = (r + \tilde{\mathbb{P}}_h \hat{V}_{h+1})(\mathbf{s},a)$ $\hat{V}_h(\mathbf{s}) = \max_a \hat{Q}_h(\mathbf{s},a)$	$\tilde{Q}_h^k(\mathbf{s},a) = \min\{(\hat{r}_h^k + b_h^k + \tilde{\mathbb{P}}_h'^k \tilde{V}_{h+1}^k)(\mathbf{s},a), B_k(H)\}$ $\tilde{V}_h^k(\mathbf{s}) = \max_a \tilde{Q}_h^k(\mathbf{s},a)$
Greedy	$\mathcal{U}_h(\mathbf{s},a) = (r + \mathbb{P}_h U_{h+1})(\mathbf{s},a)$ $U_h(\mathbf{s}) = \mathcal{U}_h(\mathbf{s}, \pi^L(\mathbf{s},h))$	$\bar{\mathcal{U}}_h^k(\mathbf{s},a) = \min\{(\hat{r}_h^k + b_h^k + \hat{\mathbb{P}}_h^k \bar{U}_{h+1}^k)(\mathbf{s},a), B_k(H)\}$ $\bar{U}_h^k(\mathbf{s}) = \bar{\mathcal{U}}_h^k(\mathbf{s}, \tilde{\pi}^L(\mathbf{s},h))$
Intermediate	$\hat{\mathcal{U}}_h(\mathbf{s},a) = (r + \mathbb{P}_h \hat{V}_{h+1})(\mathbf{s},a)$ $\hat{U}_h(\mathbf{s}) = \max_a \hat{\mathcal{U}}_h(\mathbf{s},a)$	$\tilde{\mathcal{U}}_h^k(\mathbf{s},a) = \min\{(\hat{r}_h^k + b_h^k + \hat{\mathbb{P}}_h^k \tilde{V}_{h+1}^k)(\mathbf{s},a), B_k(H)\}$ $\tilde{U}_h^k = \max_a \tilde{\mathcal{U}}_h^k(\mathbf{s},a)$

Using the functions outlined in the table, we now formally present our online greedy algorithm in Algorithm 2.

The algorithm is structured into four main parts:

1. In line 1, we initiate the process with a warm-up episode employing a random allocation policy.

2. As depicted in lines 6 to 9, for all subsequent episodes, we update the reward estimates and transition kernel estimate based on the history of past episodes, while observing the initial capacity of each product. Then we initialize all related value functions to 0.

3. As illustrated in lines 10 to 13, we update the weight parameters ($\bar{\mathbf{w}}_h^k$) for the linear value functions by putting estimated parameters into the backward induction process. The main difference between this online backward induction and the offline backward induction is that we now add a bonus function to the rewards and then truncate the function from above, the details are shown in Algorithm 3.

4. As shown in lines 14 to 20, at each step within the episode, we implement the greedy policy by optimizing the intermediate function $\tilde{\mathcal{U}}_h^k$. Throughout this process, we collect data on rewards and the trajectory path, which will aid in updating our estimations at the outset of the next episode.

The sub-algorithm 3 is computed at the beginning of each new episode, as the capacity of each type of product might change and the estimations of the unknown parameters are updated according to the outcomes observed in the last episode. We novelly incorporate the bonus function $b_h^k(\mathbf{s},a)$ into the backward induction procedure by separating it across the weight parameters. It is designed to obtain an upper bound of the value function and encourage exploration. It is important to note that we truncate each weight parameter to $(\Lambda_i(H-h+1))$. This truncation is justified by Lemma A.1, which says that any linear value function with rewards upper bounded by 1 has weight parameters not exceeding $(\Lambda_i(H-h+1))$. By truncating the weight parameters separately, it can be easily verified that the cumulative linear value function does not surpass $B_k(H)$.

Algorithm 2: Online Greedy Policy with Reinforcement Learning

Episode 0 (a warm start with a random policy): Receive the initial capacities $C_1^0, ...C_N^0$ and the initial state $\mathbf{s}_1^0$.
for *step* $h = 1, ..., H$ **do**
  Randomly select an action $a \in \mathcal{A}$ for the customer;
  Observe the rewards and the next state $\mathbf{s}_{h+1}^0$;
end
for *episode* $k = 1, ..., K$ **do**
  Update the estimations $\hat{r}_{i,l}, \hat{q}_{i,j,l}$, and the corresponding confidence radii;
  Receive the initial capacities $C_1^k, C_2^k, ..., C_N^k$ and the initial state $\mathbf{s}_1^k$;
  Initialize $\bar{\mathbf{w}}_{H+1}^k$ and $\{(\bar{\mathbf{w}}_h)_{i,j,D_i+1}^k\}_{i\in\mathcal{I},j\in\mathcal{J},h\in[H]}$ to be 0. Also, set the linear approximation functions and the bonus functions $\{\tilde{V}_h^k(\cdot), \tilde{Q}_h^k(\cdot,\cdot), b_h^k(\cdot,\cdot)\}_{h=1}^{H+1}$ as zero functions;
  for *step* $h = H, H-1, ..., 1$ **do**
    Compute the bonus function $b_h^k(i,j)$ related to each action;
    Compute the parameters $\bar{\mathbf{w}}_h^k$ based on $\bar{\mathbf{w}}_{h+1}^k$ using backward reduction as in Algorithm 3;
  end
  for *step* $h = 1, ..., H$ **do**
    Observe the current state $\mathbf{s}_h^k$;
    Compute $\tilde{\mathcal{U}}_h^k(\mathbf{s}_h^k, \cdot) = \min\{(\hat{r}_h^k + b_h^k + \hat{\mathbb{P}}_h^k \tilde{V}_{h+1}^k)(\mathbf{s}_h^k, \cdot), B_k(H)\}$, where $\tilde{V}_{h+1}^k(\mathbf{s}) = (\bar{\mathbf{w}}_{h+1}^k)^\top \mathbf{s}$;
    Take the greedy action $a_h^k \in arg\max_a \tilde{\mathcal{U}}_h^k(\mathbf{s}_h^k, a)$. (If there are multiple solutions, then randomly select one as the action);
    Observe the rewards $R^{t,k}(\mathbf{s}_h^k, a_h^k)$;
    The whole system changes to the next state $\mathbf{s}_{h+1}^k$;
  end
end

Now, we elaborate on the design of the bonus term $b_h^k(\cdot,\cdot)$. First, we define a larger function $\Gamma_h^k(\cdot,\cdot)$ that characterizes the gap in value functions (under both the real transition kernel and the linear transition kernel) caused by estimation errors. We aim for this function to satisfy the following inequality for any function V_{h+1} bounded by $B_k(H)$.

$$\Gamma_h^k(\mathbf{s},a) \geq \begin{cases} |r(\mathbf{s},a) - \hat{r}_h^k(\mathbf{s},a)| + |(\tilde{\mathbb{P}}_h(\cdot|\mathbf{s},a) - \tilde{\mathbb{P}}_h'^k(\cdot|\mathbf{s},a))^\top V_{h+1}(\cdot)| \\ |r(\mathbf{s},a) - \hat{r}_h^k(\mathbf{s},a)| + |(\mathbb{P}_h(\cdot|\mathbf{s},a) - \hat{\mathbb{P}}_h^k(\cdot|\mathbf{s},a))^\top V_{h+1}(\cdot)| \end{cases} \tag{37}$$

The first line in (37) shows the performance gap under linear function approximation with the modified transition kernel, while the second line shows the gap under the true value function with the original transition kernel. For each line, the first term represents the difference between the actual reward and the estimated reward, while the second term denotes the difference in expected future rewards, calculated using real transition probabilities versus estimated transition probabilities.

According to Lemma D.3, when $a = (i', j')$, the following form satisfies the requirement with high

Algorithm 3: Online Backward Induction Algorithm

input : Total capacity $C = (C_1^k, C_2^k, ..., C_N^k)^\top$, total horizon H, estimated pairing reward $\hat{r}_{i,l}^k$, estimated transition dynamics $\hat{q}_{i,j,l}^k$, weight parameters $\bar{\mathbf{w}}_{h+1}^k$ and the bonus function related to each state component $b_h^k(i,j)$.

Compute the optimal product to offer, take any

$$(i'_h, j'_h) \in \text{argmax}_{i\in\mathcal{I}\cup\{0\}, j\in\mathcal{J}\cup\{0\}}(1-\hat{q}_{i,j,0}^k)(p_j + \hat{r}_{i,1}^k - (1-\hat{q}_{i,j,1}^k)(\bar{w}_{i,0}^{h+1,k} - \bar{w}_{i,j,1}^{h+1,k})) + b_h^k(i,j) \tag{34}$$

for $i \in \mathcal{I}$ **do**

Compute weight parameters for step h

$$\tilde{w}_{i,0}^{h,k} = \bar{w}_{i,0}^{h+1,k} + 1_{\{i=i'_h\}}\frac{1}{C_i^k}[(1-\hat{q}_{i,j'_h,0}^k)(p_{j'_h} + \hat{r}_{i,1}^k - (1-\hat{q}_{i,j'_h,1}^k)(\bar{w}_{i,0}^{h+1,k} - \bar{w}_{i,j'_h,1}^{h+1,k})) + b_h^k(i,j'_h)] \tag{35}$$

Compute $\bar{w}_{i,0}^{h,k} = \min\{\tilde{w}_{i,0}^{h,k}, \Lambda_i(H-h+1)\}$;

for $l = 1, ..., D_i$ *and* $j = 1, ..., M$ **do**

$$\tilde{w}_{i,j,l}^{h,k} = \hat{r}_{i,l+1}^k + \hat{q}_{i,j,l+1}^k \bar{w}_{i,0}^{h+1,k} + (1-\hat{q}_{i,j,l+1}^k)\bar{w}_{i,j,l+1}^{h+1,k} + b_h^k(i,j,l) \tag{36}$$

$\bar{w}_{i,j,l}^{h,k} = \min\{\tilde{w}_{i,j,l}^{h,k}, \Lambda_i(H-h+1)\}$;

end

end

output: weight parameters $\bar{\mathbf{w}}_h^k$.

probability:

$$\begin{aligned}\Gamma_h^k(\mathbf{s},a) = 2\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=2}^{D_i+1} s_{i,j,l-1}[B_k(H)rad_k(q_{i,j,l}) + rad_k(r_{i,l})] \\ +[4B_k(H)(rad_k(q_{i',j',0}) + rad_k(q_{i',j',1})) + rad_k(r_{i',1})]\end{aligned} \tag{38}$$

When $a = (0,0)$, the second line in (38) is 0. Denote the first line as $\Gamma_{1,h}^k(\mathbf{s},a)$, which is the gap brought by rented products due to uncertainty of rewards and transition probability at each rental period, and is only related to states. The second line is denoted as $\Gamma_{2,h}^k(\mathbf{s},a)$. It captures the uncertainty of customers' acceptance rate and the case of returning immediately, which is only related to action.

The policy can then be selected by maximizing a function that takes the estimation error into account as in many existing papers (Jin et al. [2020], Cai et al. [2020], Zhou et al. [2020]):

$$\pi_h^k(\mathbf{s}) := max_a(\hat{r}_h^k + \Gamma_h^k + \tilde{\mathbb{P}}_h'^k \tilde{V}_h^k)(\mathbf{s},a) \tag{39}$$

Bonus Reduction: While Γ_h^k can effectively compensate for the estimation error, sometimes its magnitude is too large, leading to a slow decrease in the bonus term during the learning process. We

show in C that our bonus term can be finally reduced to:

$$b_h^k(\mathbf{s},a)=\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=2}^{D_i+1}s_{i,j,l-1}[rad_k(q_{i,j,l})|\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,l}^{h+1,k}|+rad_k(r_{i,l})]$$
$$+[2(rad_k(q_{i',j',0})+rad_k(q_{i',j',1}))|(\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,1}^{h+1,k})|+rad_k(r_{i',1})] \tag{40}$$

By separating it across the state dimension (i,j,l), it can be viewed as adding the following weight to the original weight parameters, as exhibited in Algorithm 3.

$$b_h^k(i,j)=rad_k(r_{i,1})+2(rad_k(q_{i,j,0})+rad_k(q_{i,j,1}))|(\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,1}^{h+1,k})| \tag{41}$$
$$b_h^k(i,j,l)=rad_k(r_{i,l+1})+rad_k(q_{i,j,l+1})|(\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,l+1}^{h+1,k})| \tag{42}$$

It merely replaces $2B_k(H)$ notation in Γ_h^k to the absolute value of marginal costs $|(\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,1}^{h+1,k})|$ and $|(\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,l+1}^{h+1,k})|$. These marginal costs do not exceed $2B_k(H)$ as indicated by Lemma A.1, thus $b_h^k(\cdot,\cdot)$ is no larger than $\Gamma_h^k(\cdot,\cdot)$.

In practice, most marginal costs can be much smaller than $B_k(H)$ as this bound is derived from the infinity norm of weight parameters. For instance, as proved in lemma D.4, we have $|(\bar{w}_{i,0}^{h+1,k}-\bar{w}_{i,j,1}^{h+1,k})|\leq\max\{p_{\max}+1,D_{\max}\},\forall(i,j)\in\mathcal{A}$. When D_{max} and p_{max} is much smaller than $H\times C_k$, at least the $b_h^k(i,j)$ part can effectively be reduced. This helps to achieve better numerical performance and speed up the learning process without compromising theoretical results or sacrificing optimality.

5 Regret Analysis

In this section, we provide a non-asymptotic analysis for Algorithm 2 and establish a performance guarantee measured by the revenue loss attributed to the lack of knowledge about the real parameters, commonly referred to as "regret." We will begin by introducing the definition of regret and subsequently present our theoretical results along with a proof sketch.

We assess the performance of Algorithm 2 by comparing the revenue it generates against the revenue earned by the greedy policy under the full information setting. We use this for comparison instead of the optimal revenue obtained by solving the dynamic program in (7) because the latter is hindered by the "curse of dimensionality." This decision is also rooted in the fact that the greedy policy has been demonstrated to be at least half as effective as the optimal policy when complete knowledge of rewards and transition probabilities is available. Thus, it serves as an offline oracle for the learning problem. We refer to this as the "$1/2$-regret," and it is defined as:

$$Reg(T)=\sum_{k=1}^{K}\frac{1}{2}V_1^*(\mathbf{s}_1^k)-V_1^{\pi_k}(\mathbf{s}_1^k), \tag{43}$$

which is the difference between $\frac{1}{2}$ of the optimal revenue, and the expected revenue generated by our proposed learning policy as described in Algorithm 2, denoted as π_k for the k^{th} episode. Furthermore,

since it is proved in Proposition 4.2 that the offline linear value function $\hat{V}_1^k(\mathbf{s}_1^k) \geq \frac{1}{2}V_1^*(\mathbf{s}_1^k)$, we have:

$$\begin{aligned} Reg(T) &\leq \sum_{k=1}^{K} \hat{V}_1^k(\mathbf{s}_1^k) - V_1^{\pi_k}(\mathbf{s}_1^k) \\ &= \sum_{k=1}^{K} \hat{V}_1^k(\mathbf{s}_1^k) - \tilde{V}_1^k(\mathbf{s}_1^k) + \tilde{V}_1^k(\mathbf{s}_1^k) - \bar{U}_1^k(\mathbf{s}_1^k) + \bar{U}_1^k(\mathbf{s}_1^k) - V_1^{\pi_k}(\mathbf{s}_1^k) \\ &\leq \sum_{k=1}^{K} \underbrace{(\hat{V}_1^k(\mathbf{s}_1^k) - \tilde{V}_1^k(\mathbf{s}_1^k))}_{\text{gap of linear value approximations}} + \underbrace{(\bar{U}_1^k(\mathbf{s}_1^k) - V_1^{\pi_k}(\mathbf{s}_1^k))}_{\text{gap of real value function with policy } \pi_k} \end{aligned} \tag{44}$$

In the above formula, $\bar{U}_1^k(\mathbf{s}_1^k)$ is the online greedy value function for episode k and $\tilde{V}_1^k$ is the online linear value function for episode k. Both of them take the estimated parameters as inputs and adds an additional bonus function b_h^k to the rewards generated at the h^{th} step, as defined in the right column of Table 1. Moreover, the notation $V_1^{\pi_k}$ is the realized expected revenue following the online greedy policy π_k in Algorithm 2. It takes the real parameters of rewards and usage time distributions as inputs and represents the actual expected reward we can get by using the policy π_k. Based on Proposition 5.1, we have $\tilde{V}_1^k(\mathbf{s}_1^k) - \bar{U}_1^k(\mathbf{s}_1^k) \leq 0$, which leads to the establishment of the second inequality.

Proposition 5.1. $\bar{U}_1^k(\sum_{i=1}^N C_i^k \mathbf{e}_{i,0}) \geq \tilde{V}_1^k(\sum_{i=1}^N C_i^k \mathbf{e}_{i,0})$ *for all* $k \in [K]$

In fact, the above proposition can be easily derived from Proposition 4.3 by regarding the bonus term as part of the reward function.

From the last line of (44), we can see that the regret can be mainly divided into two terms. The first term represents the performance gap between the offline linear value function and the online linear value function. The second term represents the performance gap between the online greedy value function and the real value function applying the same greedy policy π_k.

Next, we introduce the concept of a "clean event," which assures, with high probability, that the true parameters fall within the confidence intervals centered around the empirical estimations, with radii defined in (28) and (29).

Definition 5.1. *(clean event) We call an event such that* $|r_{i,l} - \hat{r}_{i,l}^k| \leq rad_k(r_{i,l}), \forall l = 1, ..., D_i + 1, |q_{i,j,l} - \hat{q}_{i,j,l}^k| \leq rad_k(q_{i,j,l}), \forall l = 0, ..., D_i$ *for all* $i \in \mathcal{I}, j \in \mathcal{J}$ *as the clean event, denoted as* $\mathcal{E}_{clean}$.

If at least one confidence interval does not hold, we refer to it as the "bad event." Now, we state the main theorem in our paper. Denote $B_{max}(H) = \max_k B_k(H)$ and $B_{aver} = B_{max}(H)/H$, and we can prove in Theorem 5.1 that Algorithm 2 achieves a $\tilde{O}(\sqrt{T})$ regret, where the notation $\tilde{O}$ hides logrithmic terms:

Theorem 5.1. *The* $1/2$*-regret is smaller than*

$$2\delta B_{aver} + 40B_{max}(H)\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2 T}$$

by taking $\delta = C\sqrt{T}$*, where* C *is a constant only related to* D_{max}, M, N*, the algorithm achieves* $\tilde{O}(\sqrt{T})$ *regret.*

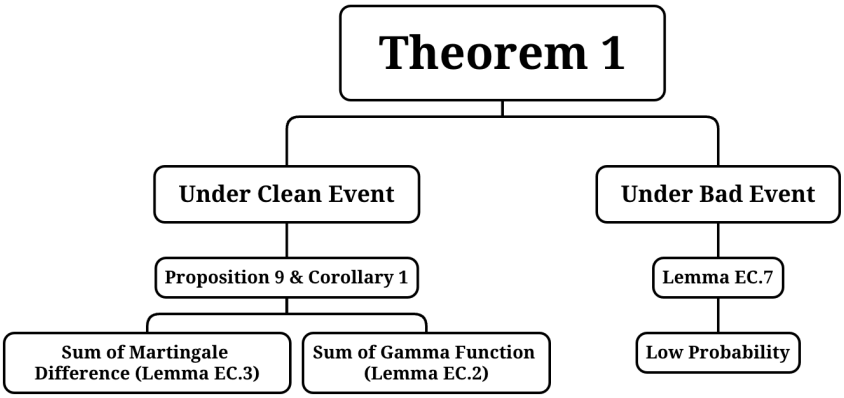


Figure 1: Outline for Proving Theorem 5.1

Proof sketch of Theorem 5.1: Now we provide a brief sketch of the proof of Theorem 5.1, with detailed steps deferred to Appendix B.4. The proof structure is outlined in Figure 1. We divide the analysis based on whether the clean event occurs, as the confidence bounds are applicable only under the clean event condition. Outside the clean event, we directly use $B_{max}(H)$ to bound the performance gap so that (44) is smaller than $2KB_{max}(H) = 2TB_{aver}$. To be more specific, we have:

$$Reg(T) \leq P(\neg\mathcal{E}_{clean})2TB_{aver} + P(\mathcal{E}_{clean})\sum_{k=1}^{K}[(\hat{V}_1^k(\mathbf{s}_1^k) - \tilde{V}_1^k(\mathbf{s}_1^k)) + (\bar{U}_1^k(\mathbf{s}_1^k) - V_1^{\pi_k}(\mathbf{s}_1^k))] \quad (45)$$

It is proved in Lemma D.5 in the appendix that $P(\neg\mathcal{E}_{clean}) \leq \frac{\delta}{T}$, so that the first term is smaller than $2\delta B_{aver}$, which is a constant independent of time. For the second term, according to Corollary 5.2.1, the problem boils down to the sums of a martingale difference sequence and Γ_h^k functions:

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\zeta_{h+1}^k, \quad \sum_{k=1}^{K}\sum_{h=1}^{H}\Gamma_h^k(\mathbf{s}_h^k, a_h^k) \quad (46)$$

which, according to Lemma B.1 and D.1, yields a regret of $O(\log(\frac{T}{\delta})\sqrt{T})$. By summing the cases under the clean event and bad event, we get the a form $C_1\delta + C_2\log(\frac{T}{\delta})\sqrt{T}$, where C_1, C_2 are constants unrelated to T and δ, matching the regret form in Theorem 5.1.

Now we proceed to introduce Proposition 5.2 and its corallary. It shows that, under the clean event, the two performance gaps in (44) can be expressed in an iterative form. This form comprises the current gap brought by the estimation error and the anticipated gap of the value functions in the next step:

Proposition 5.2. *For all $(\mathbf{s}, a, h, k) \in \mathcal{S}_k \times \mathcal{A} \times [H] \times [K]$, under the clean event, we have*

$$\bar{U}_h^k(\mathbf{s}) - V_h^{\pi_k}(\mathbf{s}) \leq E_{\pi_k}[\hat{\mathbb{P}}_h^k(\bar{U}_{h+1}^k - V_{h+1}^{\pi_k})(\mathbf{s}, \pi_k(\mathbf{s}, h)) + 2\Gamma_h^k(\mathbf{s}, \pi_k(\mathbf{s}, h))] \quad (47)$$

Moreover, let $\bar{Q}_h^k(\mathbf{s}, a) := (\hat{r}_h^k + \Gamma_h^k + \tilde{\mathbb{P}}_h'^k\bar{V}_h^k)(\mathbf{s}, a)$ and $\bar{V}_h^k(\cdot) = \max_a \bar{Q}_h^k(\cdot, a)$, we have:

$$\hat{V}_h^k(\mathbf{s}) - \tilde{V}_h^k(\mathbf{s}) \leq \bar{V}_h^k(\mathbf{s}) - \tilde{V}_h^k(\mathbf{s}) \leq \tilde{\mathbb{P}}_h'^k(\bar{V}_{h+1}^k - \tilde{V}_{h+1}^k)(\mathbf{s}, \bar{\pi}_h(\mathbf{s})) + \Gamma_h^k(\mathbf{s}, \bar{\pi}_h(\mathbf{s})) \quad (48)$$

where $\bar{\pi}_h(\cdot)$ is the optimal policy realizing $\bar{V}_h^k(\cdot)$.

We give a brief explanation of this proposition. Recall that Γ_h^k is defined in (38), which upper bounds the performance gap brought by estimation errors. The inequality (47) shows that for any step h in an arbitrary episode, the gap between the online greedy function and the realized expected revenue is bounded by a $O(\Gamma_h^k)$ term plus the expected gap in the next step $h+1$. The expectation $E_{\pi_k}[\cdot]$ on the right is taken with respect to the online greedy policy, which takes the randomness of selecting one action from multiple optimal solutions into account. The inequality (48) shows that the gap between the linear value function in the offline and online setting is bounded in a similar form, with the only difference that $\bar{V}_h^k$ is introduced to upper bound the performance gap. Note that this additional value function is defined only to derive an iterative form and assist the regret analysis and does not influence the decisions made by the platform.

In the analysis, we find the following corollary useful, which further turns the performance gap into the sum of estimation error bounds Γ_h^k and a martingale difference sequence ζ_h^k, paving the way for proving Theorem 5.1.

Corollary 5.2.1. *(Recursive Formula) Let $\delta_h^k = \bar{U}_h^k(\mathbf{s}_h^k) - V_h^{\pi_k}(\mathbf{s}_h^k)$ and $\zeta_{h+1}^k = E[\delta_{h+1}^k | \mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k, h)] - \delta_{h+1}^k$ where the expectation is taken over $\hat{\mathbb{P}}_h^k$ and π_k. Then under the clean event $\mathcal{E}_{clean}$, we have $\forall (k,h) \in [K] \times [H]$:*

$$\delta_h^k \leq \delta_{h+1}^k + \zeta_{h+1}^k + 2E_{\pi_k}[\Gamma_h^k(\mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k, h))] \tag{49}$$

and specifically:

$$\delta_1^k \leq \sum_{h=1}^{H} (\zeta_{h+1}^k + 2E_{\pi_k}[\Gamma_h^k(\mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k, h))]) \tag{50}$$

Similarly, let $\delta_h^{\prime k} = \bar{V}_h^k(\mathbf{s}_h^k) - \tilde{V}_h^k(\mathbf{s}_h^k)$ and $\zeta_{h+1}^{\prime k} = E[\delta_{h+1}^{\prime k} | \mathbf{s}_h^k, \hat{\pi}_h(\mathbf{s}_h^k)] - \delta_{h+1}^{\prime k}$ where the expectation is taken over $\tilde{\mathbb{P}}_h^{\prime k}$. Under the clean event, we have:

$$\delta_1^{\prime k} \leq \sum_{h=1}^{H} (\zeta_{h+1}^{\prime k} + \Gamma_h^k(\mathbf{s}_h^k, \bar{\pi}(\mathbf{s}_h^k))) \tag{51}$$

By summing (50),(51) of all episodes, we get the form in (46), which is critical for proving Theorem 5.1.

6 Numerical Experiment

In this section, we will first introduce the benchmark methods to compare our proposed method with. Then we will conduct numerical experiments on synthetic data.

6.1 Benchmark Methods

We will first develop an offline benchmark which is an upper bound of the optimal reward under the full information setting. This offline benchmark is represented as a linear program, aiming to act as a

baseline for the offline algorithm. We will omit the index k and refer to it as LP-E, which is defined below.

$$\begin{aligned}
\max_{y_{i,j,h},(\mathbf{s}_h)_{i,j,l},(\mathbf{s}_h)_{i,0}} & \sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{M}(1-q_{i,j,0})(p_j+r_{i,1})y_{i,j,h} + \sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=2}^{D_i+1} r_{i,l}(\mathbf{s}_h)_{i,j,l-1} \\
\text{s.t.} \quad & \mathbf{s}_{h+1} = E[\tilde{\mathbf{S}}_{h+1}] - \sum_{i=1}^{M}\sum_{j=1}^{M} y_{i,j,h}(1-q_{i,j,0})(1-q_{i,j,1})(\mathbf{e}_{i,0}-\mathbf{e}_{i,j,1}) \quad h=1,...,H-1 \\
& \mathbf{s}_1 = \sum_{i=1}^{N} C_i \mathbf{e}_{i,0} \\
& \sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j,h} \le 1 \quad h=1,...,H \\
& y_{i,j,h} \ge 0, (\mathbf{s}_h)_{i,j,l} \ge 0, (\mathbf{s}_h)_{i,0} \ge 0, \forall i \in [N], j \in [M], l \in [D_i], h \in [H].
\end{aligned} \tag{52}$$

The decision variable $y_{i,j,h}$ denotes the probability of offering product i at price j in the h^{th} step. The decision variable $\mathbf{s}_h$ signifies the expected state at the h^{th} step. This is computed based on the first constraint, where we replace the stochastic process with a fluid approximation. It is guaranteed to serve as an upper bound, as stated in Proposition 6.1.

Proposition 6.1. *The linear program in (52) is an upper bound of the original problem V_1^* for each episode.*

In the learning setting, to evaluate our algorithm's performance, we use an ϵ-greedy policy as a benchmark. Specifically, with probability ϵ, we select a random allocation policy, while with probability $1-\epsilon$, we choose the greedy policy with estimated parameters. The complete method is shown in Algorithm 4.

Algorithm 4: ϵ-greedy policy

input : K, H, ϵ

for *episode* $k = 1, ..., K$ **do**
- Update the estimators $\hat{r}_{i,l}, \hat{q}_{i,j,l}$, and input them into Algorithm 1 to obtain the weight parameters $\hat{\mathbf{w}}^k$;
- Receive the initial capacities $C_1^k, ..., C_N^k$;
- **for** *step* $h = 1, ..., H$ **do**
  - Observe the current state $\mathbf{s}$;
  - With probability ϵ, randomly choose an action from $\mathcal{A}$;
  - Otherwise, choose the action that maximizes

$$1_{\{s_{i,0}>0\}}(1-\hat{q}_{i,j,0})[p_j + \hat{r}_{i,1} - (1-\hat{q}_{i,j,1})(\hat{w}_{i,0}^{k,h+1} - \hat{w}_{i,j,1}^{k,h+1})]$$

    If the final result is non-positive, then reject the request;
  - Observe the rewards and the system changes to the next state;
- **end**

end

6.2 Numerical Experiment on Synthetic Data

In this section, we perform numerical experiments using synthetic data to assess the effectiveness of our approach. We fix the parameters as $N = 50$, $M = 2$, $H = 200$, and $D_{max} = 51$. This setup involves 50 product types, each with two pricing options—a higher and a lower one. The capacity for each resource type is set at $C_i = 2$. The rental distribution $f_{i,j,l}$ follows a geometric distribution, as illustrated in Figure 2.

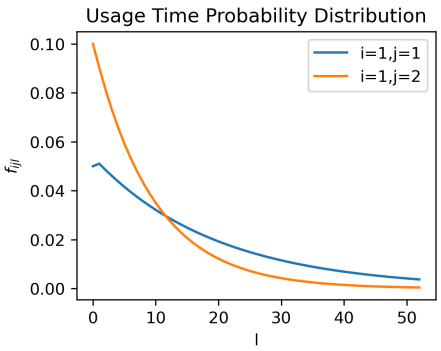


Figure 2: Usage Time Distribution for Product 1 under High (f_{11}) and Low (f_{12}) Prices

We assume that a higher price corresponds to a longer usage time, reflecting the practical scenario where customers may require more time to recover from a higher initial cost.

The expected rewards $r_{i,l}$ are generated in the range of $[0, 4]$ and decrease linearly along the usage time, as illustrated in Figure 3. Products with higher initial rewards experience a steeper decline.

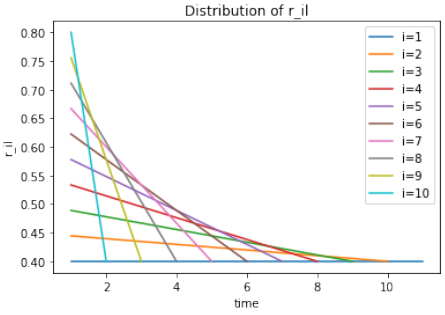


Figure 3: Reward Distribution

The initial value of rewards $r_{i,l}$ and transition probabilities $q_{i,j,l}$ are both set to 0. We compare our online algorithm with the ϵ-greedy method with $\epsilon = 0.001, 0.01, 0.1$. We repeat the experiment 10 times and take the average to evaluate the performance of all algorithms.

We will now illustrate the performance of each algorithm under the specified parameter setting. Before delving into that, it's worth noting that the offline greedy algorithm attains an average performance of 4676, whereas the upper bound reaches 4798. This indicates that the greedy algorithm performs very well in practice, significantly exceeding the theoretical performance guarantee of $1/2$. Figure 4 illustrates the average revenue earned in each episode, comparing against the offline greedy algorithm.

The red line represents our algorithm, while the blue, orange, and green lines depict the ϵ-greedy algorithms with $\epsilon = 0.001$, 0.01, and 0.1, respectively. The brown line corresponds to the offline greedy

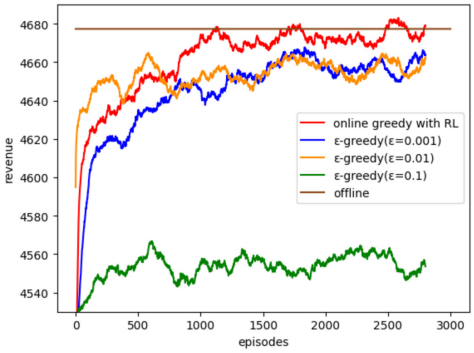
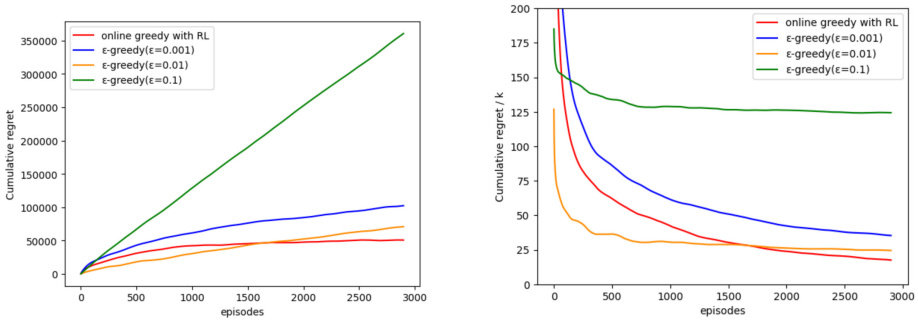


Figure 4: Average Performance

algorithm. We observe that our algorithm exhibits lower revenue in the initial episodes compared with $\epsilon = 0.01$ but surpasses all three ϵ-greedy algorithms over time. This initial dip is attributed to the exploration phase, where the algorithm explores various actions and refines the parameters of each product for more accurate learning. As time progresses, the bonus function diminishes, allowing the algorithm to exploit its acquired knowledge effectively and gradually converge to the performance under full information. For $\epsilon = 0.001$, the algorithm tends to spend more time exploiting current actions, leading to suboptimal solutions due to errors in estimating unknown parameters. Conversely, when $\epsilon = 0.1$, it dedicates too much time to exploring suboptimal actions, resulting in the worst performance. Our UCB-based method strikes a balance between exploration and exploitation, ultimately achieving the best performance.



(a) Cumulative Regret (b) Time-average Regret

Figure 5: Average and Cumulative Regret against Offline

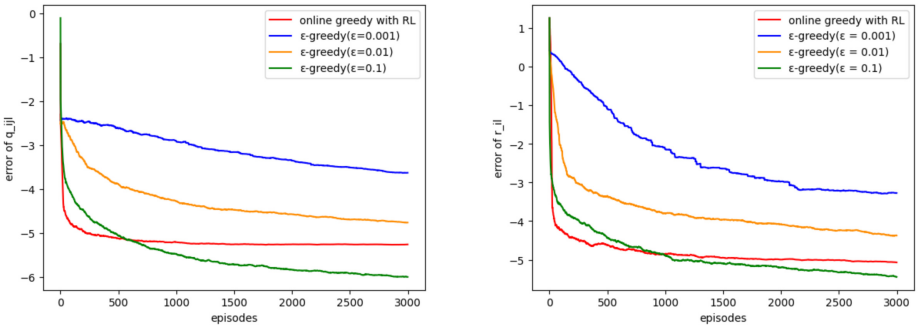
In Figure 5, we present the regret of each algorithm against the offline benchmark. Figure 5a clearly demonstrates that the online greedy algorithm with RL, shown as the red line exhibits a distinct sublinear regret curve, surpassing the performance of all ϵ-greedy algorithms after 1700 episodes. This trend is further supported by Figure 5b, where the averaged regret of our algorithm ultimately proves to be smaller than that of the three benchmarks.

We also compare the estimation errors of different algorithms, as illustrated in Figure 6. The y axis

refers to the log error:

$$\log(\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=0}^{D_i}|q_{i,j,l}-\hat{q}_{i,j,l}^{k}|)$$
$$\log(\sum_{i=1}^{N}\sum_{l=1}^{D_i+1}|r_{i,l}-\hat{r}_{i,l}^{k}|).$$

It shows that our proposed algorithm consistently achieves quicker and lower estimation errors com-



(a) Learning Error of Transition Probabilities (b) Learning Error of Rewards

Figure 6: Learning Error of Unknown Parameters

pared to the ϵ-greedy algorithms with $\epsilon = 0.001$ and 0.01. While letting $\epsilon = 0.1$ achieves lower estimation error after 1000 episodes, it comes at the cost of overall performance.

6.3 Case Study on Real Dataset

In this section, we use real data from TikTok to conduct our case study. We collect 7217 posted contents from 50 Key Opinion Leaders (KOL). For each content, we collect the detailed view counts for each day from the date it was posted for two months. Figure 7 shows the daily viewcount increase of a post randomly sampled from the dataset. The x-axis refers to the days after publishing the post. As we can see, the reward varies significantly over time.

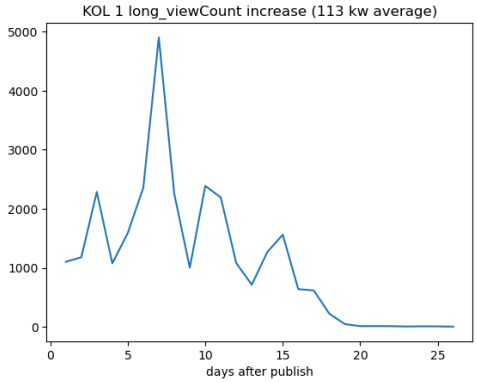
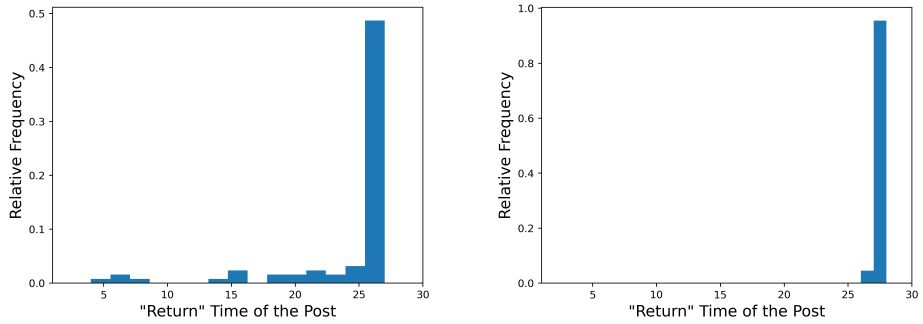


Figure 7: Average Reward Distribution for Posted Contents from KOL 1

We treat different KOLs as different type of resources, which are owned by a company. The advertising requests arrive sequentially. For each advertising request, the company chooses a KOL as well as

a price level to offer. The customer can choose to accept or reject the offer. Once the customer accepts it, the chosen KOL will publish a post for the advertising request, and the company continues to collect advertising revenue at each time period, which is related to the viewcount of this post. Each KOL publishes many posts, and the viewcount data of each post from this KOL is viewed as a piece of record of this KOL being rented. We use the average of all posts from a KOL to estimate the "return" time and reward of this KOL.

A post is "returned" when it is no longer exposed by the recommendation system. As each KOL is allocated only a limited number of exposures in the recommendation system, outdated posts no longer receive exposures due to the forgetting mechanism. Therefore, these posts can be viewed as being spared for upcoming new advertising content. To set the "return" time of a KOL, we analyse the post "return" time distribution for each KOL. Figure 8a and 8b are two typical relative frequency histogram of post "return" time from two KOLs. It can be observed that most, if not all, posts are returned in the last several periods. Therefore, we make a natural simplification, assuming that a post will always get exposure until a fixed predetermined "return" time of its KOL.That means the resource will never be returned until the last period, so we no longer need to estimate the return probability of each period, and only estimate the rejection probability when offering the resources.



(a) Return Distribution of KOL1 (b) Return Distribution of KOL2

Figure 8: Visualization of Return Time

After removing the posts whose "return" time is too short , we take the average daily viewcount increase of all posts from a KOL as the expected reward of each period from this KOL. Without loss of generality, we normalize the reward and set the maximum reward as 20 in our case study. We assume each KOL has 2 units of capacity and set two pricing options—a higher and a lower one with the value of 0.5 and 1. We run K=620 episodes, each with length H=200. The performance results are shown in Figure 9. As we can see, our proposed method surpasses the benchmarks after about 250 episodes. We leave further details regarding the case study in the Appendix.

7 Conclusions

In this research, we formulate the reusable resource allocation and pricing problem as a MDP and concentrate on gradually learning unknown rewards and usage time distributions while making decisions with bandit feedback. Departing from conventional online learning methods, we introduce methodologies from episodic reinforcement learning to address the cold start problem, enabling adaptation to time-varying rewards, non-parametric distributions, and periodic adjustment of resource capacities. Our

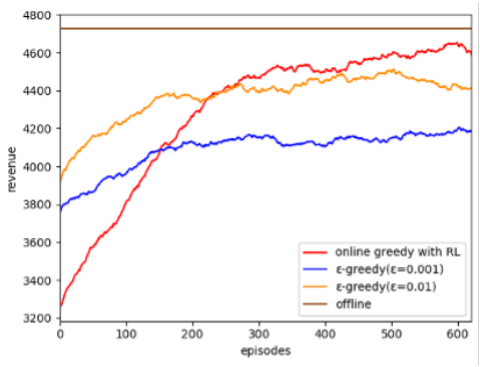


Figure 9: Average Performance of Case Study

proposed algorithm, the online greedy policy, leverages a UCB type bonus term to balance exploration and exploitation. Theoretical analysis demonstrates that our algorithm can achieve $\tilde{O}(\sqrt{T})$ regret, and numerical experiments substantiate its effectiveness.

Discussions:

1. We assume that the platform retrieves the products being used at the end of each episode, which might incur a switching cost. When the switching cost is high, we modify the boundary condition of the Bellman equation by letting $w_{i,j,l}^{H+1}$ be negative instead of 0 accordingly. This parameter represents the retrieving cost of each unit of product i with price j and usage time l at the end. Then, according to Algorithm 1, as the termination time approaches, the opportunity cost is large, and the platform is less likely to offer the products.

2. In our primary model, we assume that rewards are independent of the initial price, and customers are homogeneous, simplifying notations for better comprehension. However, our methodology readily extends to more general scenarios. If rewards are correlated with the price, similar to the usage time parameters $q_{i,j,l}$, we can introduce a new dimension for rewards, denoted as $r_{i,j,l}$, and the confidence intervals would have the same structure as that for $q_{i,j,l}$. In the case of heterogeneous customers with a known arrival rate, only minor adjustments are needed in the backward induction procedure, as discussed in Appendix F.

3. In our scenario, we assume that the price is a decision made by the platform for the sake of generality. However, in certain situations, the price might be exogenously determined through auctions. In such cases, we can treat the price as part of the context in our problem. Similar to handling heterogeneous customers, if the price distribution is known, we only need to make minor adjustments to the backward induction process to align with our algorithm framework. Under cases where price distributions are unknown, we can apply machine learning techniques to predict the prices for the subsequent episode. In this way, our method can still be applied.

Future Directions: Our study has certain limitations and opens up potential avenues for future research. For instance, in advertising, auctions may not adhere to a stationary distribution or exhibit certain patterns easy to be predicted as mentioned in the discussion. Integrating game theory, learning, and earning in this context is an important direction. Moreover, exploring more intricate data properties in data-driven problems, such as high-dimensional learning, nonstationary settings, and privacy concerns, could be subjects for future investigation.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Daniel Adelman. Dynamic bid prices in revenue management. *Operations Research*, 55(4):647–661, 2007.

Alexia Atsidakou, Orestis Papadigenopoulos, Soumya Basu, Constantine Caramanis, and Sanjay Shakkottai. Combinatorial blocking bandits with stochastic delays. In *International Conference on Machine Learning*, pages 404–413. PMLR, 2021.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR, 2017.

Jackie Baek and Will Ma. Bifurcating constraints to improve approximation ratios for network revenue management with reusable resources. *Operations Research*, 70(4):2226–2236, 2022.

Santiago R Balseiro, Will Ma, and Wenxin Zhang. Dynamic pricing for reusable resources: The power of two prices. *arXiv preprint arXiv:2308.13822*, 2023.

Soumya Basu, Rajat Sen, Sujay Sanghavi, and Sanjay Shakkottai. Blocking bandits. *Advances in Neural Information Processing Systems*, 32, 2019.

Soumya Basu, Orestis Papadigenopoulos, Constantine Caramanis, and Sanjay Shakkottai. Contextual blocking bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 271–279. PMLR, 2021.

Dimitri Bertsekas. *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific, 2021.

Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*, pages 1283–1294. PMLR, 2020.

Yiwei Chen, Retsef Levi, and Cong Shi. Revenue management of reusable resources with advanced reservations. *Production and Operations Management*, 26(5):836–859, 2017.

Yiding Feng, Rad Niazadeh, and Amin Saberi. Linear programming based online policies for real-time assortment of reusable resources. *Chicago Booth Research Paper*, (20-25), 2019.

Yiding Feng, Rad Niazadeh, and Amin Saberi. Near-optimal bayesian online assortment of reusable resources. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 964–965, 2022.

Kris Johnson Ferreira, David Simchi-Levi, and He Wang. Online network revenue management using thompson sampling. *Operations research*, 66(6):1586–1602, 2018.

Vineet Goyal, Garud Iyengar, and Rajan Udwani. Asymptotically optimal competitive ratio for online allocation of reusable resources. *arXiv preprint arXiv:2002.02430*, 2020.

Jiafan He, Dongruo Zhou, and Quanquan Gu. Logarithmic regret for reinforcement learning with linear function approximation. In *International Conference on Machine Learning*, pages 4171–4180. PMLR, 2021.

Tianming Huo and Wang Chi Cheung. Online reusable resource assortment planning with customer-dependent usage durations. *Available at SSRN 4504713*, 2023.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Huiwen Jia, Cong Shi, and Siqian Shen. Online learning and pricing for service systems with reusable resources. *Operations Research*, ahead of print, 2022a.

Huiwen Jia, Cong Shi, and Siqian Shen. Online learning and pricing for network revenue management with reusable resources. *Advances in Neural Information Processing Systems*, 35:4830–4842, 2022b.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

Subodha Kumar, Varghese S Jacob, and Chelliah Sriskandarajah. Scheduling advertisements on a web page to maximize revenue. *European journal of operational research*, 173(3):1067–1089, 2006.

Yanzhe Lei and Stefanus Jasin. Real-time dynamic pricing for revenue management with reusable resources, advance reservation, and deterministic service time requirements. *Operations Research*, 68(3):676–685, 2020.

Retsef Levi and Ana Radovanović. Provably near-optimal lp-based policies for revenue management in systems with reusable resources. *Operations Research*, 58(2):503–507, 2010.

Francisco S Melo and M Isabel Ribeiro. Q-learning with linear function approximation. In *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20*, pages 308–322. Springer, 2007.

Xin Pan, Jie Song, Jingtong Zhao, and Van-Anh Truong. Online contextual learning with perishable resources allocation. *Iise Transactions*, 52(12):1343–1357, 2020.

Paat Rusmevichientong, Mika Sumida, and Huseyin Topaloglu. Dynamic assortment optimization for reusable products with random usage durations. *Management Science*, 66(7):2820–2844, 2020.

Paat Rusmevichientong, Mika Sumida, Huseyin Topaloglu, and Yicheng Bai. Revenue management with heterogeneous resources: Unit resource capacities, advance bookings, and itineraries over time intervals. *Operations Research*, 2023.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

Xinchang Xie, Itai Gurvich, and Simge Küçükyavuz. Dynamic allocation of reusable resources: Logarithmic regret in overloaded networks. *preprint available online*, 2022.

Lin Yang and Mengdi Wang. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, pages 6995–7004. PMLR, 2019.

Mohammad Zhalechian, Esmaeil Keyvanshokooh, Cong Shi, and Mark P Van Oyen. Data-driven hospital admission control: A learning approach. *Operations Research*, 2023.

Xilin Zhang and Wang Chi Cheung. Online resource allocation for reusable resources. *arXiv preprint arXiv:2212.02855*, 2022.

Yueyang Zhong, John R Birge, and Amy Ward. Learning the scheduling policy in time-varying multiclass many server queues with abandonment. *Available at SSRN*, 2022.

Dongruo Zhou and Quanquan Gu. Computationally efficient horizon-free reinforcement learning for linear mixture mdps. *Advances in neural information processing systems*, 35:36337–36349, 2022.

Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Conference on Learning Theory*, pages 4532–4576. PMLR, 2021.

Huozhi Zhou, Jinglin Chen, Lav R Varshney, and Ashish Jagmohan. Nonstationary reinforcement learning with linear function approximation. *arXiv preprint arXiv:2010.04244*, 2020.

Appendix A Proofs for Section 4

A.1 Proof of Proposition 4.1

Proof. As we can see from (11), it's obvious that $w^h_{i,0} \geq w^{h+1}_{i,0}, \forall h \in [H]$. We next prove by induction that $w^h_{i,j,l} \geq w^{h+1}_{i,j,l}$. Since

$$w^h_{i,j,D_i} = r_{i,D_i+1} + w^{h+1}_{i,0} \geq r_{i,D_i+1} + w^{h+2}_{i,0} = w^{h+1}_{i,j,D_i}. \tag{53}$$

We have that the statement $w^h_{i,j,l} \geq w^{h+1}_{i,j,l}$ is correct for $l = D_i$ and all $h \in [H]$. Now we prove by induction that the statement is correct for all l by starting from $l = D_i$ to $l = 1$.

Suppose $w^h_{i,j,l} \geq w^{h+1}_{i,j,l}$ is satisfied for all $l \geq m+1$ and $h \in [H]$, then for $l = m$ with an arbitrary h:

$$\begin{aligned} w^h_{i,j,m} &= r_{i,m+1} + q_{i,j,m+1} w^{h+1}_{i,0} + (1 - q_{i,j,m+1}) w^{h+1}_{i,j,m+1} \\ &\geq r_{i,m+1} + q_{i,j,m+1} w^{h+2}_{i,0} + (1 - q_{i,j,m+1}) w^{h+2}_{i,j,m+1} \\ &= w^{h+1}_{i,j,m}, \end{aligned} \tag{54}$$

where the inequality is because $w^{h+1}_{i,0} \geq w^{h+2}_{i,0}$ and $w^{h+1}_{i,j,m+1} \geq w^{h+2}_{i,j,m+1}$ by the induction hypothesis. It is proved that $w^h_{i,j,m} \geq w^{h+1}_{i,j,m}$ for all $h \in [H]$. Therefore the result is proved. □

A.2 Proof of Proposition 4.2

Proof. From Adelman [2007], we know that the following linear program serves as an upper bound of the value function

$$\min \quad \tilde{V}_h(\mathbf{s}_h) \tag{55}$$

$$\begin{aligned} \text{s.t.} \quad \tilde{V}_t(\mathbf{s}) \geq \sum_{i=1}^{N} \sum_{l=1}^{D_i} r_{i,l+1} \Big(\sum_{j=1}^{M} s_{i,j,l}\Big) + E[\tilde{V}_{t+1}(\tilde{\mathbf{S}}_{t+1})] + \\ 1_{\{s_{i,0}>0\}} (1 - q_{i,j,0}) \big\{ (p_j + r_{i,1} - (1 - q_{i,j,1}) \{ E[\tilde{V}_{t+1}(\tilde{\mathbf{S}}_{t+1}) - \tilde{V}_{t+1}(\tilde{\mathbf{S}}_{t+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})] \} \big\} \\ \forall \mathbf{s} \in \mathcal{S}, i \in \mathcal{I}, j \in \mathcal{J}, t = h, h+1, ..., H \end{aligned} \tag{56}$$

$$\tilde{V}_t(\mathbf{s}) \geq \sum_{i=1}^{N} \sum_{l=1}^{D_i} r_{i,l+1} \Big(\sum_{j=1}^{M} s_{i,j,l}\Big) + E[\tilde{V}_{t+1}(\tilde{\mathbf{S}}_{t+1})] \quad \forall \mathbf{s} \in \mathcal{S}, t = h, h+1, ..., H. \tag{57}$$

Note that for $t = H+1$, we follow the convention that $\tilde{V}_{H+1} = 0$. The last constraint is because we can always reject the request if offering the product creates a negative reward. Define the constant $\beta_t = \sum_{i=1}^{N} w^t_{i,0} C_i$. We proceed to prove that $\{\beta_t + \hat{V}_t(\mathbf{s}) : \mathbf{s} \in \mathcal{S}, t = h, ...H\}$ is feasible to the above linear program. Since the set $\mathcal{S}$ is finite as no product can be used for more than D_{max} periods, we have a finite number of constraints.

$$\beta_{t+1} + E[\hat{V}_{t+1}(\tilde{\mathbf{S}}_{t+1})] = \beta_{t+1} + \hat{V}_{t+1}(E[\tilde{\mathbf{S}}_{t+1}]) \tag{58}$$

$$= \beta_{t+1} + \sum_{i=1}^{N}[w_{i,0}^{t+1}(s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} q_{i,j,l+1}s_{i,j,l}) + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{t+1}(s_{i,j,l} - q_{i,j,l+1}s_{i,j,l})] \tag{59}$$

$$= \beta_{t+1} + \sum_{i=1}^{N}[w_{i,0}^{t+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}(q_{i,j,l+1}w_{i,0}^{t+1} + (1 - q_{i,j,l+1})w_{i,j,l}^{t+1})]. \tag{60}$$

The first equality is because $\hat{V}_{t+1}$ can be expressed as a sum of linear functions. We now begin to evaluate the right-hand side of the constraint (56) by plugging in the expression (60).

$$\begin{aligned}\text{R.H.S. of (56)} = &\sum_{i=1}^{N}\sum_{l=1}^{D_i} r_{i,l+1}(\sum_{j=1}^{M} s_{i,j,l}) + \beta_{t+1} + \sum_{i=1}^{N}[w_{i,0}^{t+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}(q_{i,j,l+1}w_{i,0}^{t+1} \\ &+ (1 - q_{i,j,l+1})w_{i,j,l}^{t+1})] + 1_{\{s_{i,0}>0\}}(1 - q_{i,j,0})\{(p_j + r_{i,1} - (1 - q_{i,j,1})(w_{i,0}^{t+1} - w_{i,j,1}^{t+1})\}\end{aligned} \tag{61}$$

$$\begin{aligned}&= \sum_{i=1}^{N} w_{i,0}^{t+1}C_i + \sum_{i=1}^{N}\{w_{i,0}^{t+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{t}s_{i,j,l}\} \\ &+ 1_{\{s_{i,0}>0\}}(1 - q_{i,j,0})\{(p_j + r_{i,1} - (1 - q_{i,j,1})(w_{i,0}^{t+1} - w_{i,j,1}^{t+1})\}.\end{aligned} \tag{62}$$

The second equality is due to the backward induction formula (12). Similarly, we can express the right-hand side of constraint (57) as:

$$\text{R.H.S. of (57)} = \sum_{i=1}^{N} w_{i,0}^{t+1}C_i + \sum_{i=1}^{N}\{w_{i,0}^{t+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{t}s_{i,j,l}\}. \tag{63}$$

Since during the backward induction in Algorithm 1, we choose an action $\hat{a}_t$ which is either $(\hat{i}_t, \hat{j}_t)$ or $(0, 0)$ depending on which makes the parameter $w_{i,0}^t$ bigger. It is obvious that if we take the maximum of the two terms (62) and (63), the optimal action should be $\hat{a}_t$, and we have that either of the two terms is no more than:

$$\begin{aligned}&\sum_{i=1}^{N} w_{i,0}^{t+1}C_i + \sum_{i=1}^{N}\{w_{i,0}^{t+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{t}s_{i,j,l}\} + 1_{\{s_{\hat{i}_t,0}>0\}} \\ &(1 - q_{\hat{i}_t,\hat{j}_t,0})\{(p_{\hat{j}_t} + r_{\hat{i}_t,1} - (1 - q_{\hat{i}_t,\hat{j}_t,1})(w_{\hat{i}_t,0}^{t+1} - w_{\hat{i}_t,\hat{j}_t,1}^{t+1})\}^+\end{aligned} \tag{64}$$

$$\begin{aligned}&\leq \sum_{i=1}^{N} w_{i,0}^{t+1}C_i + \sum_{i=1}^{N}\{w_{i,0}^{t}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{t}s_{i,j,l}\} + 1_{\{s_{\hat{i}_t,0}>0\}} \\ &(1 - q_{\hat{i}_t,\hat{j}_t,0})\{(p_{\hat{j}_t} + r_{\hat{i}_t,1} - (1 - q_{\hat{i}_t,\hat{j}_t,1})(w_{\hat{i}_t,0}^{t+1} - w_{\hat{i}_t,\hat{j}_t,1}^{t+1})\}^+\end{aligned} \tag{65}$$

$$= \sum_{i=1}^{N} w_{i,0}^{t}C_i + \sum_{i=1}^{N}\{w_{i,0}^{t}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{t}s_{i,j,l}\} \tag{66}$$

$$= \beta_t + \hat{V}_t(\mathbf{s}). \tag{67}$$

The first inequality is because $w_{i,0}^t \geq w_{i,0}^{t+1}$. The first equality is according to the update (11). This proves that the function $\{\beta_t + \hat{V}_t(\mathbf{s}) : \mathbf{s} \in \mathcal{S}, t = h, ...H\}$ is feasible. Since at time $t = 1$, the expression is $\beta_1 + \hat{V}_1(\mathbf{s}) = \sum_{i=1}^N w_{i,0}^1 C_i + \sum_{i=1}^N \{w_{i,0}^1 s_{i,0} + \sum_{l=1}^{D_i} \sum_{j=1}^M w_{i,j,l}^1 s_{i,j,l}\}$, and we have $C_i = s_{i,0}$. Therefore:

$$\beta_1 + \hat{V}_1(\mathbf{s}) = 2\sum_{i=1}^N w_{i,0}^1 C_i + \sum_{i=1}^N \sum_{l=1}^{D_i} \sum_{j=1}^M w_{i,j,l}^1 s_{i,j,l} \tag{68}$$

$$\leq 2\sum_{i=1}^N (w_{i,0}^1 C_i + \sum_{l=1}^{D_i} \sum_{j=1}^M w_{i,j,l}^1 s_{i,j,l}) \tag{69}$$

$$= 2\hat{V}_1(\mathbf{s}). \tag{70}$$

The inequality is due to the non-negativity of the weight parameters. Suppose the optimal solution of the linear program is $\tilde{V}_1^*(\mathbf{s})$. Since $\beta_1 + \hat{V}_1(\mathbf{s})$ is a feasible solution of the linear program, we know that:

$$2\hat{V}_1(\mathbf{s}) \geq \tilde{V}_1^*(\mathbf{s}) \geq V_1(\mathbf{s}). \tag{71}$$

The last inequality is because the solution of the LP program is an upper bound of the original MDP as mentioned at the start. □

A.3 Proof of Proposition 4.3

Proof. We can compute the function U_h by the following recursion:

$$U_h(\mathbf{s}_h) = \sum_{i=1}^N \sum_{l=1}^{D_i} r_{i,l+1}(\sum_{j=1}^M s_{i,j,l}) + E[U_{h+1}(\tilde{\mathbf{S}}_{h+1})] +$$

$$1_{\{(i,j)=\pi^L(\mathbf{s},h)\}} 1_{\{s_{i,0}>0\}} (1 - q_{i,j,0})\{(p_j + r_{i,1} - (1 - q_{i,j,1})\{E[U_{h+1}(\tilde{\mathbf{S}}_{h+1}) - U_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})]\}\}, \tag{72}$$

with the boundary condition $U_{H+1} = 0$, and we use the greedy policy $\pi^L(\mathbf{s}, h)$ instead of the optimal policy in the recursion formula. Next, we will prove by induction that $U_h(\mathbf{s}) \geq \hat{V}_h(\mathbf{s})$. This holds at

$h = H + 1$, and we now prove by induction. Assume that this is correct when $t = h + 1$, and

$$U_h(\mathbf{s}) \geq \sum_{i=1}^{N}\sum_{l=1}^{D_i} r_{i,l+1}(\sum_{j=1}^{M} s_{i,j,l}) + E[\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1})] + 1_{\{(i,j)=\pi^L(\mathbf{s},h)\}}1_{\{s_{i,0}>0\}}$$
$$(1-q_{i,j,0})\{(p_j + r_{i,1} - (1-q_{i,j,1})\{E[\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})]\}\} \tag{73}$$
$$= \sum_{i=1}^{N}\sum_{l=1}^{D_i} r_{i,l+1}(\sum_{j=1}^{M} s_{i,j,l}) + \sum_{i=1}^{N}[w_{i,0}^{h+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}(q_{i,j,l+1}w_{i,0}^{h+1} + (1-q_{i,j,l+1})w_{i,j,l}^{h+1})]$$
$$+ 1_{\{(i,j)=\pi^L(\mathbf{s},h)\}}1_{\{s_{i,0}>0\}}(1-q_{i,j,0})\{(p_j + r_{i,1} - (1-q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1})\} \tag{74}$$
$$\geq \sum_{i=1}^{N}\sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}w_{i,j,l}^{h} + \sum_{i=1}^{N} s_{i,0}w_{i,0}^{h+1}$$
$$+ \sum_{i=1}^{N} 1_{\{(i,j)=\hat{\pi}_h\}}\frac{s_{i,0}}{C_i}(1-q_{i,j,0})(p_j + r_{i,1} - (1-q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1})). \tag{75}$$

To see this, note that by rearranging the terms, the coefficient of every $U_{h+1}(\tilde{\mathbf{S}}_{h+1})$ is:

$$1 - 1_{\{(i,j)=\pi^L(\mathbf{s},h)\}}1_{\{s_{i,0}>0\}}(1-q_{i,j,0})(1-q_{i,j,1})$$

which is non-negative. The coefficient of $U_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})$ is :

$$1_{\{(i,j)=\pi^L(\mathbf{s},h)\}}1_{\{s_{i,0}>0\}}(1-q_{i,j,0})(1-q_{i,j,1})$$

which is also non-negative.

Therefore, replacing them with smaller functions $\hat{V}_{h+1}$ reduces the value. The second inequality is because we have $1_{\{s_{i,0}>0\}} \geq \frac{s_{i,0}}{C_i}$. We know that:

$$\sum_{i=1}^{N}[s_{i,0}w_{i,0}^{h+1} + 1_{\{(i,j)=\hat{\pi}_h\}}\frac{s_{i,0}}{C_i}(1-q_{i,j,0})(p_j + r_{i,1} - (1-q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1}))] \tag{76}$$
$$= \sum_{i=1}^{N} s_{i,0}w_{i,0}^{h+1} + 1_{\{(i,j)=\hat{\pi}_h\}}s_{i,0}(w_{i,0}^{h} - w_{i,0}^{h+1}).$$

Putting this into (75), we have:

$$(75) = \sum_{i=1}^{N}\sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}w_{i,j,l}^{h} + \sum_{i=1}^{N} s_{i,0}w_{i,0}^{h+1} + 1_{\{(i,j)=\hat{\pi}_h\}}s_{i,0}(w_{i,0}^{h} - w_{i,0}^{h+1}) \tag{77}$$
$$= \sum_{i=1}^{N}\sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}w_{i,j,l}^{h} + \sum_{i=1}^{N} s_{i,0}[w_{i,0}^{h} + (1 - 1_{\{(i,j)=\hat{\pi}_h\}})(w_{i,0}^{h+1} - w_{i,0}^{h})] \tag{78}$$
$$\geq \sum_{i=1}^{N}\sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}w_{i,j,l}^{h} + \sum_{i=1}^{N} s_{i,0}w_{i,0}^{h} = \hat{V}_h(\mathbf{s}). \tag{79}$$

Therefore, we prove that $U_h(\mathbf{s}) \geq \hat{V}_h(\mathbf{s}), \forall h \in [H], s \in \mathcal{S}$. By taking $h = 1$, the proof is completed. □

A.4 Proof of Proposition 4.4

First, we will prove the following lemma that gives an upper bound of the weight parameters of the linear function approximation:

Lemma A.1. *(Bound on weight parameters) Denote* $\|w_i^h\|_\infty := \max\{w_{i,0}^h, w_{i,j,l}^h : j = 1, ..., M, l = 1, ..., D_i\}$,$\forall h \in [H]$ *and* $i \in \mathcal{I}$*, then we have* $\|w_i^h\|_\infty \leq (H + 1 - h)\Lambda_i$

Proof. Since for $h = 1, ..., H$, the weights are computed as follows:

$$
\begin{aligned}
w_{i,0}^h &\leq w_{i,0}^{h+1} + \max\{\frac{1}{C_i} 1_{\{i=\hat{i}_h\}}(p_{\hat{j}_h} + r_{i,1} - (1 - q_{i,j,0})(w_{i,0}^{h+1} - w_{i,\hat{j}_h,1}^{h+1})), 0\} \\
&\leq w_{i,0}^{h+1} + \max\{\frac{1}{C_i}(p_{\hat{j}_h} + r_{i,1} - (1 - q_{i,j,0})(w_{i,0}^{h+1} - w_{i,\hat{j}_h,1}^{h+1})), 0\} \\
&\leq \max\{\frac{1}{C_i}(1 + p_{max}) + [1 - \frac{1}{C_i}(1 - q_{i,j,0})]w_{i,0}^{h+1} + \frac{1}{C_i}(1 - q_{i,j,0})w_{i,\hat{j}_h,1}^{h+1}, w_{i,0}^{h+1}\} \\
&\leq \Lambda_i + \max\{w_{i,0}^{h+1}, w_{i,\hat{j}_h,1}^{h+1}\}, \qquad (80) \\
w_{i,j,l}^h &= r_{i,l+1} + q_{i,j,l+1}w_{i,0}^{h+1} + (1 - q_{i,j,l+1})w_{i,j,l+1}^{h+1} \\
&\leq 1 + \max\{w_{i,0}^{h+1}, w_{i,j,l+1}^{h+1}\} \leq \Lambda_i + \max\{w_{i,0}^{h+1}, w_{i,j,l+1}^{h+1}\}, \qquad (81) \\
w_{i,j,D_i}^h &= r_{i,D_i+1} + w_{i,0}^{h+1} \leq \Lambda_i + w_{i,0}^{h+1}. \qquad (82)
\end{aligned}
$$

By putting (80), (81) and (82) together, we get:

$$
\begin{aligned}
\|w_i^h\|_\infty &\leq \Lambda_i + \max\{w_{i,0}^{h+1}, w_{i,j,l}^{h+1} : j = 1, ..., M, l = 1, ..., D_i\} \qquad (83) \\
&= \Lambda_i + \|w_i^{h+1}\|_\infty \qquad (84) \\
&\leq \Lambda_i + \Lambda_i + \|w_i^{h+2}\|_\infty \qquad (85) \\
&\leq \cdots \leq \Lambda_i(H - h + 1) + \|w_i^{H+1}\|_\infty \qquad (86) \\
&= \Lambda_i(H - h + 1). \qquad (87)
\end{aligned}
$$

□

Based on lemma A.1, we can prove Proposition 4.4:

Proof. To prove $\hat{V}_h(\mathbf{s}) \leq H \sum_{i\in\mathcal{I}} \Lambda_i C_i$, note that from Proposition A.1, we know that $\|w_i^h\|_\infty \leq \Lambda_i(H - h + 1)$. Therefore,

$$
\begin{aligned}
\hat{V}_h(\mathbf{s}) &= \sum_{i=1}^{N}(w_{i,0}^h s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^h s_{i,j,l}) \leq \sum_{i=1}^{N}\|w_i^h\|_\infty(s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} s_{i,j,l}) \qquad (88) \\
&\leq \sum_{i=1}^{N}\Lambda_i(H - h + 1)C_i \leq H\sum_{i=1}^{N}\Lambda_i C_i. \qquad (89)
\end{aligned}
$$

□

A.5 Proof of Proposition 4.5

Proof. We prove this by induction. First, note that when $h = H + 1$, we have $\mathbf{w}_h = 0$ and $\hat{V}_h(\cdot) = 0$ and satisfies (16) naturally. Suppose that the equation holds for $t = h + 1$, now we prove that it also holds for $t = h$. Note that we have :

$$\hat{V}_{h+1}(\mathbf{s}) = \sum_{i=1}^{N}(w_{i,0}^{h+1}s_{i,0} + \sum_{l=1}^{D_i}\sum_{j=1}^{M} w_{i,j,l}^{h+1}s_{i,j,l}) = \mathbf{w}_{h+1}^{\top}\mathbf{s}$$

Therefore:

$$\tilde{\mathbb{P}}_h\hat{V}_{h+1}(\mathbf{s}, a) = E[\hat{V}_{h+1}|\mathbf{s}, a] = \mathbf{w}_{h+1}^{\top}E[\mathbf{s}_{h+1}|\mathbf{s}, a]$$

Since for each coordinate of the current state $\mathbf{s}$, it follows a binomial distribution. Therefore, we have $E[(\mathbf{s}_{h+1})_{i,j,l}|\mathbf{s}, a] = (1 - q_{i,j,l})s_{i,j,l-1}$ and

$$E[(\mathbf{s}_{h+1})_{i,0}|\mathbf{s}, a = (i', j')] = s_{i,0} + \sum_{j=1}^{M}\sum_{l=1}^{D_i} q_{i,j,l+1}s_{i,j,l} - \frac{1}{C_i}(1 - q_{i',j',0})(1 - q_{i',j'1})s_{i,0}$$

The products in state $s_{i,j,l}$ generates reward $r_{i,l+1} \cdot s_{i,j,l}$ while the product in state $s_{i',0}$ generates reward $\frac{1}{C_{i'}}(1-q_{i',j',0})(r_{i',l}+p_{j'})$. Therefore, by selecting all the terms related to the coordinate (i, j, l), we get the related parameter for $s_{i,j,l}$ denoted as $\tilde{w}_{i,j,l}^h = r_{i,l+1} + w_{i,0}^{h+1}q_{i,j,l+1} + w_{i,j,l+1}^{h+1}(1 - q_{i,j,l+1})$, which, according to the iteration in Algorithm 1, equals $w_{i,j,l}^h$. Similarly, the related parameter for $s_{i',0}$ is

$$\tilde{w}_{i',0}^h = w_{i',0}^{h+1} + \frac{1}{C_{i'}}(1 - q_{i',j',0})(r_{i',l} + p_{j'} - (1 - q_{i',j',1})(w_{i',0}^{h+1} - w_{i',j',1}^{h+1}))$$

To select the action that maximizes the current value function, it is equivalent to maximizing the weight parameter. The action only influences parameter $\tilde{w}_{i',0}^h$. Therefore, by selecting the term satisfying (10), it is easy to check that the above procedure is the same as Algorithm 1. Therefore, we have $\tilde{\mathbf{w}}_h = \mathbf{w}_h$. That is, the right hand side of (16) equals the value function computed in the previous algorithm, which is the left hand side of (16). The result is proved. □

A.6 Proof of Proposition 4.6

Proof. We start by finding an explicit representation of ϕ. For any state $\mathbf{s} \in \mathcal{S}_k$, we map it into a vector with dimension $N + MD_{tot}$:

$$\psi(\mathbf{s}) := vec((\mathbf{s}_i)_{i=1}^N) = \begin{bmatrix} vec(\mathbf{s}_1) \\ \vdots \\ vec(\mathbf{s}_N) \end{bmatrix}. \tag{90}$$

where $vec(\mathbf{s}_i)$ denotes vectorizing the state $\mathbf{s}_i$ into the order $(s_{i,0}, s_{i,1,1}, ..., s_{i,1,D_i}, .., s_{i,M,1}, ..., s_{i,M,D_i})$, and we map the action into a vector with dimension NM. If the action is $a = (i, j)$, then the map is

denoted as $\psi'(a) = \mathbf{e}_{i,j} := (0, 0, ..., 1, ...0)^\top$, with its $(M(i-1)+j)^{th}$ element equal to 1 and others equal to 0. If the action is to reject the request, then $\psi'(a) = (0, ..., 0)$, a zero vector.

By concatenating the two maps vertically, we get the following feature map:

$$\phi(\mathbf{s}, a) = \begin{bmatrix} \psi(\mathbf{s}) \\ \psi'(a) \end{bmatrix}. \tag{91}$$

The value function can be expressed as $\hat{V}_h(\mathbf{s}) = \mathbf{w}_h^\top \psi(\mathbf{s})$, where the weights $\mathbf{w}_h$ are obtained from Algorithm 1, and arranged in the same order as $\psi(\mathbf{s})$.

Now we proceed to prove that there exists $\boldsymbol{\mu}_h \in \mathbb{R}^d$ such that $\hat{Q}_h(\cdot, \cdot) = \boldsymbol{\mu}_h^\top \phi(\cdot, \cdot)$. Construct a vector γ_1 with dimension $N + MD_{tot}$ as follows, we start by defining vector $\gamma_{1,i}$ for $i \in [N]$:

$$\gamma_{1,i} := vec(0, \underbrace{r_{i,2}, ...r_{1,2}}_{\text{M terms}}, ..., \underbrace{r_{i,D_1+1}, ...r_{i,D_1+1}}_{\text{M terms}})$$

It represents a column vector with length $1 + MD_i$. Then we concatenate them together into one long column vector γ_1:

$$\gamma_1 := vec((\gamma_{1,i})_{i=1}^N) = vec \begin{bmatrix} \gamma_{1,1} \\ \vdots \\ \gamma_{1,N} \end{bmatrix} \tag{92}$$

We define another vector γ_2 with dimension NM, with its $(M(i-1)+j)^{th}$ element equal to $(1 - q_{i,j,0})(r_{i,1} + p_j)$. Then it's easy to see that the reward $r(\mathbf{s}, a) = [\gamma_1, \gamma_2]^\top \phi(\mathbf{s}, a)$.

Furthermore, we define two matrices Γ and Σ. Γ is a block matrix composed of Γ_i with dimension $(1 + MD_i) \times (1 + MD_i)$ as shown in (93)

$$\Gamma_i = \begin{bmatrix} 1 & q_{i,1,2} & q_{i,1,3} & \cdots & q_{i,M,D_i+1} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 - q_{i,1,2} & 0 & \cdots & 0 \\ 0 & 0 & 1 - q_{i,1,3} & 0 & \cdots \\ \vdots & \vdots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & 1 - q_{i,M,D_i} & 0 \end{bmatrix}. \tag{93}$$

The matrix Γ is used to express the expected pre-action state $E[\psi(\tilde{\mathbf{S}}_{h+1}) | \mathbf{s}_h = \mathbf{s}]$, and it's easy to verify that this is equal to $\Gamma\psi(\mathbf{s})$.

Now we proceed to express the change to the post-action state with $\psi'(a)$. Σ is also a block matrix composed of Σ_i with dimension $(1 + MD_i) \times M$ as shown in (94), where only the first two lines are

non-zero:

$$\Sigma_i = \frac{1}{C_i}\begin{bmatrix} -(1-q_{i,1,0})(1-q_{i,1,1}) & -(1-q_{i,2,0})(1-q_{i,2,1}) & \cdots & -(1-q_{i,M,0})(1-q_{i,M,1}) \\ (1-q_{i,1,0})(1-q_{i,1,1}) & (1-q_{i,2,0})(1-q_{i,2,1}) & \cdots & (1-q_{i,M,0})(1-q_{i,M,1}) \\ 0 & 0 & \cdots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{94}$$

This is because when the $\mathbf{e}_{i,j}$ element of $\psi'(a)$ is non-zero, under the modified transition kernel $\tilde{\mathbb{P}}$ and by taking action (i,j), there is a probability of $\frac{1}{C_i}(1-q_{i,j,0})(1-q_{i,j,1})$ that a unit $\mathbf{e}_{i,0}$ in the current state change to $\mathbf{e}_{i,j,1}$. Then we have:

$$\begin{aligned} \hat{Q}_h(\mathbf{s},a) &= r(\mathbf{s},a) + (\tilde{\mathbb{P}}_h \hat{V}_{h+1})(\mathbf{s},a) \\ &= r(\mathbf{s},a) + E[\mathbf{w}_{h+1}^\top \psi(\mathbf{s}_{h+1})|\mathbf{s}_h = \mathbf{s}, a_h = a] \\ &= [\gamma_1,\gamma_2]^\top \phi(\mathbf{s},a) + \mathbf{w}_{h+1}^\top E[\psi(\mathbf{s}_{h+1})|\mathbf{s}_h = \mathbf{s}, a_h = a] \\ &= [\gamma_1,\gamma_2]^\top \phi(\mathbf{s},a) + \mathbf{w}_{h+1}^\top [\Gamma,\Sigma] \begin{bmatrix} \psi(\mathbf{s}) \\ \psi'(a) \end{bmatrix} \\ &= \underbrace{[\gamma_1^\top + \mathbf{w}_{h+1}^\top \Gamma, \gamma_2^\top + \mathbf{w}_{h+1}^\top \Sigma]}_{\boldsymbol{\mu}_h^\top} \phi(\mathbf{s},a). \end{aligned} \tag{95}$$

This proves the result. □

A.7 Proof of Proposition 4.7

Proof. First, we will prove (17) is correct. This is equivalent to proving that:

$$\pi^L(\mathbf{s},h) = argmax_a \hat{U}_h(\mathbf{s},a). \tag{96}$$

Since

$$\begin{aligned} \max_a \hat{U}_h(\mathbf{s},a) &= \max_a (r(\mathbf{s},a) + \mathbb{P}_h \hat{V}_{h+1}(\mathbf{s},a)) \\ &= \max_a (r(\mathbf{s},a) + E[\hat{V}_{h+1}(\mathbf{s}_{h+1})|\mathbf{s}_h = \mathbf{s}, a_h = a]) \\ &= \max_a (r(\mathbf{s},a) + E[\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1})] \\ &\quad - 1_{\{a_h=(i,j)\}} 1_{\{s_{i,0}\geq 0\}} (1-q_{i,j,0})(1-q_{i,j,1})\{E[\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})]\}) \\ &= \max_a (1_{\{a=(i,j), s_{i,0}\geq 0\}} [(1-q_{i,j,0})(r_{i,1}+p_j) - (1-q_{i,j,0})(1-q_{i,j,1})(w_{i,0}^{h+1} - w_{i,j,1}^{h+1})]). \end{aligned} \tag{97}$$

According to the definition of π^L in (13), the action that maximizes $\hat{U}_h(\mathbf{s},a)$ is the action provided by policy π^L. Therefore, (17) is satisfied.

We now prove that the intermediate value function $\hat{U}_h^k(\cdot)$ is smaller than the explicit greedy value function $U_h^k(\cdot)$. The value function based on the greedy policy is: $U_h(\mathbf{s}) = r(\mathbf{s}, \pi^L(\mathbf{s})) + \mathbb{P}_h U_{h+1}(\mathbf{s}, \pi^L(\mathbf{s}))$. When $h = H$, we have

$$U_H(\mathbf{s}) = r(\mathbf{s}, \pi^L(\mathbf{s})) = \max_a r(\mathbf{s}, a) = \max_a \hat{U}_H(\mathbf{s}, a) = \hat{U}_H(\mathbf{s}).$$

since $\hat{V}_{H+1}(\cdot) = U_{H+1}(\cdot) \equiv 0$. When $h = 1, ..., H-1$, recall that it is proved in Proposition 4.3 that the real value function $U_{h+1}(\cdot) \geq \hat{V}_{h+1}(\cdot)$. Therefore, $\mathbb{P}_h U_{h+1}(\mathbf{s}, a) \geq \mathbb{P}_h \hat{V}_{h+1}(\mathbf{s}, a)$ as $\mathbb{P}_h$ is a monotone operator. Hence,

$$U_h(\mathbf{s}) = \max_a (r(\mathbf{s}, a) + \mathbb{P}_h U_{h+1}(\mathbf{s}, a)) \geq \max_a (r(\mathbf{s}, a) + \mathbb{P}_h \hat{V}_{h+1}(\mathbf{s}, a)) = \hat{U}_h(\mathbf{s}).$$

□

Appendix B Proofs for Section 5

B.1 Proof of Proposition 5.1

Similar to the proof of Proposition 4.3, denote $\tilde{\pi}^L(\mathbf{s}, h)$ the policy we derived from the intermediate function. We have

$$\begin{aligned}
\bar{U}_h(\mathbf{s}_h) &= \sum_{i=1}^{N} \sum_{l=1}^{D_i} \sum_{j=1}^{M} [\hat{r}_{i,l+1} s_{i,j,l}] + E[\bar{U}_{h+1}(\tilde{\mathbf{S}}_{h+1})] \\
&+ 1_{\{(i,j)=\tilde{\pi}^L(\mathbf{s},h)\}} 1_{\{s_{i,0}>0\}} \Big((1 - \hat{q}_{i,j,0}) \{ (p_j + \hat{r}_{i,1} \\
&- (1 - \hat{q}_{i,j,1}) \{ E[\bar{U}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \bar{U}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})] \} \} + b_h(i,j) \Big),
\end{aligned} \tag{98}$$

with the boundary condition $\bar{U}_{H+1} = 0$. We now prove by induction that $\bar{U}_h(\mathbf{s}) \geq \tilde{V}_h(\mathbf{s})$. Note that we omit the index k as the proof applies for each episode. The inequality holds when $h = H + 1$. We proceed to prove the inequality holds when $t = h$ given that it holds when $t = h + 1$ for an arbitrary $h \leq H$. Then by replacing the right hand side of (98), we have

$$\begin{aligned}
\bar{U}_h(\mathbf{s}) &\geq \sum_{i=1}^{N} \sum_{l=1}^{D_i} \sum_{j=1}^{M} [(\hat{r}_{i,l+1}) s_{i,j,l}] + \sum_{i=1}^{N} [\bar{w}_{i,0}^{h+1} s_{i,0} + \sum_{l=1}^{D_i} \sum_{j=1}^{M} s_{i,j,l} (\hat{q}_{i,j,l+1} \bar{w}_{i,0}^{h+1} \\
&(1 - \hat{q}_{i,j,l+1}) \bar{w}_{i,j,l}^{h+1}] + 1_{\{(i,j)=\tilde{\pi}^L(\mathbf{s},h)\}} 1_{\{s_{i,0}>0\}} \Big((1 - \hat{q}_{i,j,0}) \{ (p_j + \hat{r}_{i,1} \\
&- (1 - \hat{q}_{i,j,1}) (\bar{w}_{i,0}^{h+1} - \bar{w}_{i,j,1}^{h+1})) \} + b_h(i,j) \Big) \\
&\geq \sum_{i=1}^{N} \sum_{l=1}^{D_i} \sum_{j=1}^{M} \bar{w}_{i,j,l}^{h} s_{i,j,l} + \sum_{i=1}^{N} \bar{w}_{i,0}^{h+1} s_{i,0} + 1_{\{(i,j)=\pi^h\}} \frac{s_{i,0}}{C_i} \Big((1 - \hat{q}_{i,j,0}) \{ (p_j + \hat{r}_{i,1} \\
&- (1 - \hat{q}_{i,j,1}) (\bar{w}_{i,0}^{h+1} - \bar{w}_{i,j,1}^{h+1})) \} + b_h(i,j) \Big).
\end{aligned} \tag{99}$$

According to the backward induction in Algorithm 1, we know that (99) can be further transformed

to:

$$\sum_{i=1}^{N}\sum_{l=1}^{D_i}\sum_{j=1}^{M}\bar{w}_{i,j,l}^{h}s_{i,j,l}+\sum_{i=1}^{N}s_{i,0}\bar{w}_{i,0}^{h}=\tilde{V}_h(\mathbf{s}). \tag{100}$$

We can then prove that $\bar{U}_h(\mathbf{s})\geq\tilde{V}_h(\mathbf{s}),\forall h\in[H],\mathbf{s}\in\mathcal{S}$, and by taking $h=1$, the proof is done.

B.2 Proof of Proposition 5.2

Proof. By definition, we have:

$$\begin{aligned}
&\bar{\mathcal{U}}_h^k(\mathbf{s}_h^k,a_h^k)-Q_h^{\pi_k}(\mathbf{s}_h^k,a_h^k)\leq(\hat{r}_h-r_h)(\mathbf{s}_h^k,a_h^k)+\hat{\mathbb{P}}_h^k\bar{U}_{h+1}^k(\mathbf{s}_h^k,a_h^k)-\mathbb{P}_hV_{h+1}^{\pi_k}(\mathbf{s}_h^k,a_h^k)+b_h^k(\mathbf{s}_h^k,a_h^k)\\
&=(\hat{r}_h-r_h)(\mathbf{s}_h^k,a_h^k)+(\hat{\mathbb{P}}_h^k-\mathbb{P}_h)V_{h+1}^{\pi_k}(\mathbf{s}_h^k,a_h^k)+\hat{\mathbb{P}}_h^k(\bar{U}_{h+1}^k-V_{h+1}^{\pi_k})(\mathbf{s}_h^k,a_h^k)+b_h^k(\mathbf{s}_h^k,a_h^k)\\
&=((\hat{r}_h-r_h)+(\hat{\mathbb{P}}_h^k-\mathbb{P}_h)(V_{h+1}^{\pi_k}))(\mathbf{s}_h^k,a_h^k)+(\hat{\mathbb{P}}_h^k(\bar{U}_{h+1}^k-V_{h+1}^{\pi_k})+b_h^k)(\mathbf{s}_h^k,a_h^k)\\
&\leq\hat{\mathbb{P}}_h^k(\bar{U}_{h+1}^k-V_{h+1}^{\pi_k})(\mathbf{s}_h^k,a_h^k)+(\Gamma_h^k+b_h^k)(\mathbf{s}_h^k,a_h^k)
\end{aligned}$$

The first inequality is because we truncate the $\bar{\mathcal{U}}_h^k$ function to make it bounded, which is smaller than the untruncated form on the right. The second inequality is satisfied due to lemma D.3, which proves that

$$((\hat{r}_h-r_h)+(\hat{\mathbb{P}}_h^k-\mathbb{P}_h)(V_{h+1}^{\pi_k}))(\mathbf{s},a)\leq\Gamma_h^k(\mathbf{s},a)$$

with $||V_{h+1}^{\pi_k}||_\infty\leq B_k(H)$. Since $(\Gamma_h^k+b_h^k)(\mathbf{s},a)\leq2\Gamma_h^k(\mathbf{s},a)$, we have:

$$\bar{\mathcal{U}}_h^k(\mathbf{s}_h^k,a_h^k)-Q_h^{\pi_k}(\mathbf{s}_h^k,a_h^k)\leq2\Gamma_h^k(\mathbf{s}_h^k,a_h^k)+\hat{\mathbb{P}}_h^k(\bar{U}_{h+1}^k-V_{h+1}^{\pi_k})(\mathbf{s}_h^k,a_h^k)$$

Now that we have established the iterative form of performance gap for the action-value function. We proceed to get the iterative form for the value function. According to the definition of $\bar{U}_h^k(\mathbf{s})$ and $V_h^{\pi_k}(\mathbf{s})$, they take the same policy π_k. Therefore we have

$$\begin{aligned}
\bar{U}_h^k(\mathbf{s}_h^k)-V_h^{\pi_k}(\mathbf{s}_h^k)&=E_{\pi_k}[\bar{\mathcal{U}}_h^k(\mathbf{s}_h^k,\pi_k(\mathbf{s},h))-Q_h^{\pi_k}(\mathbf{s}_h^k,\pi_k(\mathbf{s},h))]\\
&\leq E_{\pi_k}[\hat{\mathbb{P}}_h^k(\bar{U}_{h+1}^k-V_{h+1}^{\pi_k})(\mathbf{s}_h^k,\pi_k(\mathbf{s},h))+2\Gamma_h^k(\mathbf{s}_h^k,\pi_k(\mathbf{s},h))]
\end{aligned}$$

The first part of Proposition 5.2 is derived.

For the linear value function, we first prove that $\hat{V}_h^k(\cdot)\leq\bar{V}_h^k(\cdot)$, which means $\bar{V}_h^k$ is indeed an upper bound of the linear function approximation. To achieve this, we also start from the action value function:

$$\bar{Q}_h^k(\cdot,\cdot)=\min\{(\hat{r}_h^k+\Gamma_h^k+\tilde{\mathbb{P}}_h'^k\bar{V}_{h+1}^k)(\cdot,\cdot),B_k(H)\}$$

we now prove that $\hat{Q}_h^k(\cdot,\cdot)\leq\bar{Q}_h^k(\cdot,\cdot)$ with a classification discussion:

If $\bar{Q}_h^k(\cdot,\cdot)=B_k(H)$, then this inequality holds as no value function exceeds this upper bound, proved by Propositon 4.4.

Otherwise:

$$\begin{aligned}
&\bar{Q}_h^k(\mathbf{s}_h^k, a_h^k) - \hat{Q}_h^k(\mathbf{s}_h^k, a_h^k) = (\hat{r}_h - r_h)(\mathbf{s}_h^k, a_h^k) + \tilde{\mathbb{P}}_h'^k \bar{V}_{h+1}^k(\mathbf{s}_h^k, a_h^k) - \tilde{\mathbb{P}}_h^k \hat{V}_{h+1}^k(\mathbf{s}_h^k, a_h^k) + \Gamma_h^k(\mathbf{s}_h^k, a_h^k) \\
&\geq -\Gamma_h^k(\mathbf{s}_h^k, a_h^k) + \Gamma_h^k(\mathbf{s}_h^k, a_h^k) + \tilde{\mathbb{P}}_h^k(\bar{V}_{h+1}^k - \hat{V}_{h+1}^k)(\mathbf{s}_h^k, a_h^k) \\
&= \tilde{\mathbb{P}}_h^k(\bar{V}_{h+1}^k - \hat{V}_{h+1}^k)(\mathbf{s}_h^k, a_h^k)
\end{aligned} \tag{101}$$

The first inequality is satisfied according to lemma D.3. Now we obtain the relationship of two action value functions at the h^{th} step in an iterative form. We prove by induction that:

$$\hat{Q}_t^k(\cdot, \cdot) \leq \bar{Q}_t^k(\cdot, \cdot) \quad \forall t \in [H] \tag{102}$$

When $h = H$, all the action-value functions are equal to 0 at time $H + 1$, according to (101), the last line equals 0 and we can get $\bar{Q}_H^k(\mathbf{s}_H^k, a_H^k) - \hat{Q}_H^k(\mathbf{s}_H^k, a_H^k) \geq 0$. Therefore, the condition is satisfied when $h = H$. Suppose when $t \geq h + 1$, we have $\hat{Q}_{h+1}^k(\cdot, \cdot) \leq \bar{Q}_{h+1}^k(\cdot, \cdot)$, we now want to prove it is also satisfied when $t = h$.

Note that the last line of (101) denoted the expected gap of value function at $h + 1$. We only need to prove that $(\bar{V}_{h+1}^k - \hat{V}_{h+1}^k)(\cdot) \geq 0$ for every possible state $\mathbf{s}_{h+1}$, so that the expectation is larger than 0. Since by taking the optimal static action for the case with full information setting, under the clean event $\mathcal{E}_{clean}$, we have for arbitrary state $\mathbf{s}$:

$$\hat{V}_{h+1}^k(\mathbf{s}) = \hat{Q}_{h+1}^k(\mathbf{s}, \hat{\pi}_{h+1}) \leq \bar{Q}_{h+1}^k(\mathbf{s}, \hat{\pi}_{h+1}) \leq \bar{Q}_{h+1}^k(\mathbf{s}, \bar{\pi}_{h+1}) = \bar{V}_{h+1}^k(\mathbf{s}) \tag{103}$$

The first inequality is due to the induction hypothesis that (102) is satisfied for $t = h + 1$. The second inequality holds as th policy $\hat{\pi}_{h+1}$ is suboptimal for value function $\bar{Q}_{h+1}^k$. Since for every state $\mathbf{s}$, the inequality holds, it is also holds by taking expectation. Then we can prove that (102) holds for all $t \in [H]$ and $\hat{V}_h^k(\cdot) \leq \bar{V}_h^k(\cdot)$ for all $t \in [H]$ according to (103).

Now we start to prove that the second part of Proposition 5.2 holds. We again start from the action value function:

$$\begin{aligned}
&\bar{Q}_h^k(\mathbf{s}_h^k, a_h^k) - \tilde{Q}_h^k(\mathbf{s}_h^k, a_h^k) \\
&\leq (\hat{r}_h^k + \Gamma_h^k + \tilde{\mathbb{P}}_h'^k \bar{V}_{h+1}^k)(\mathbf{s}_h^k, a_h^k) - (\hat{r}_h^k + b_h^k + \tilde{\mathbb{P}}_h'^k \tilde{V}_{h+1}^k)(\mathbf{s}_h^k, a_h^k) \\
&\leq \Gamma_h^k(\mathbf{s}_h^k, a_h^k) + \tilde{\mathbb{P}}_h'^k(\bar{V}_{h+1}^k - \tilde{V}_{h+1}^k)(\mathbf{s}_h^k, a_h^k)
\end{aligned} \tag{104}$$

where the first inequality is due to the contractive property of making truncations. The second inequality holds as the bonus function is nonnegative. If we take action $a^* = \text{argmax} \bar{Q}_h^k(\mathbf{s}_h^k, a_h^k)$ following the policy $\bar{\pi}_h^k(\cdot)$, which is optimal for $\bar{Q}_h^k$ and suboptimal for $\tilde{Q}_h^k$, then we get:

$$\bar{V}_h^k(\mathbf{s}) - \tilde{V}_h^k(\mathbf{s}) \leq \tilde{\mathbb{P}}_h'^k(\bar{V}_{h+1}^k - \tilde{V}_{h+1}^k)(\mathbf{s}, \bar{\pi}(\mathbf{s})) + \Gamma_h^k(\mathbf{s}, \bar{\pi}(\mathbf{s})) \tag{105}$$

Since

$$\hat{V}_h^k(\mathbf{s}) - \tilde{V}_h^k(\mathbf{s}) \leq \bar{V}_h^k(\mathbf{s}) - \tilde{V}_h^k(\mathbf{s}) \leq \tilde{\mathbb{P}}_h'^k(\bar{V}_{h+1}^k - \tilde{V}_{h+1}^k)(\mathbf{s}, \bar{\pi}(\mathbf{s})) + \Gamma_h^k(\mathbf{s}, \bar{\pi}(\mathbf{s}))$$

The proof is done. □

B.3 Proof of Corollary 5.2.1

Proof. According to Proposition 5.2, we know that by taking the policy π_k, we have:

$$\begin{aligned}
&\bar{U}_h^k(\mathbf{s}_h^k) - V_h^{\pi_k}(\mathbf{s}_h^k) = \delta_h^k = E[\bar{U}_{h+1}^k(\mathbf{S}_{h+1}) - V_{h+1}^{\pi_k}(\mathbf{S}_{h+1})|\mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k)] + 2E_{\pi_k}[\Gamma_h^k(\mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k))] \\
&= 2E_{\pi_k}[\Gamma_h^k(\mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k))] + \zeta_{h+1}^k + (\bar{U}_{h+1}^k(\mathbf{s}_{h+1}^k) - V_h^{\pi_k}(\mathbf{s}_{h+1}^k)) \\
&= 2E_{\pi_k}[\Gamma_h^k(\mathbf{s}_h^k, \pi_k(\mathbf{s}_h^k))] + \zeta_{h+1}^k + \delta_{h+1}^k
\end{aligned}$$

The above is directly obtained according to the definition of the martingale difference sequence ζ_{h+1}^k. The proof of $(\hat{V}_h^k - \tilde{V}_h^k)$ is similar to the above procedure. □

B.4 Proof of Theorem 5.1

First, we propose the following lemma, which is useful in the analysis of our main theorem.

Lemma B.1. *Under the clean event, we have*

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\Gamma_h^k(\mathbf{s}_h^k, a_h^k) \leq 12B_{max}(H)\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2T} \tag{106}$$

which yields a $\tilde{O}(HD_{max}\sqrt{MNT})$ term.

Proof. We first write out the explicit form of the sum consisted of the following four parts:

$$\begin{aligned}
f_1 &= \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=2}^{D_i+1} 2B_k(H)\mathbf{1}_{\{(\mathbf{s}_h^k)_{i,j,l-1}>0\}}\sqrt{\frac{\log(2D_{max}MNT^2/\delta)}{\max\{N_{i,j,l}^k, 1\}}} \\
f_2 &= \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{l=2}^{D_i+1} 2B_k(H)\mathbf{1}_{\{\sum_j(\mathbf{s}_h^k)_{i,j,l-1}>0\}}\sqrt{\frac{\log(2D_{max}NT^2/\delta)}{\max\{\tilde{N}_{i,l}^k, 1\}}} \\
f_3 &= \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{M} 4B_k(H)\mathbf{1}_{\{a_h^k=(i,j)\}}\Big(\sqrt{\frac{\log(2D_{max}MNT^2/\delta)}{\max\{N_{i,j,0}^k, 1\}}} + \sqrt{\frac{\log(2D_{max}MNT^2/\delta)}{\max\{N_{i,j,1}^k, 1\}}}\Big) \\
f_4 &= \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{M} 2B_k(H)\mathbf{1}_{\{a_h^k=(i,j)\}}\sqrt{\frac{\log(2D_{max}NT^2/\delta)}{\max\{\tilde{N}_{i,1}^k, 1\}}}.
\end{aligned}$$

Note that f_1, f_3 are related to the bonus function of the transition probabilities, and f_2, f_4 are related to the bonus function of the rewards.

In our algorithm, the counters are irrelevant to step h, and will only be updated at the end of each episode. Take $N_{i,j,l}^k$ as an example. Denote $N_{i,j,l}^{h,k}$ as the number of times product i is rented at price j for l periods, when currently we are at the h^{th} step of the k^{th} episode. We then have $N_{i,j,l}^{h,k-1} \leq N_{i,j,l}^k$.

Therefore:

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\mathbf{1}_{\{(\mathbf{s}_h^k)_{i,j,l-1}>0\}}\frac{1}{\sqrt{\max\{N_{i,j,l}^k,1\}}}\leq\sum_{k=1}^{K}\sum_{h=1}^{H}\mathbf{1}_{\{(\mathbf{s}_h^k)_{i,j,l-1}>0\}}\frac{1}{\sqrt{\max\{N_{i,j,l}^{h,k-1},1\}}} \tag{107}$$

$$=\sum_{t=1}^{\max\{N_{i,j,l}^{K-1},1\}}\frac{1}{\sqrt{t}}\leq 2\sqrt{\max\{N_{i,j,l}^{K-1},1\}}. \tag{108}$$

The first equality is because the indicator function $\mathbf{1}_{\{(\mathbf{s}_h^k)_{i,j,l-1}>0\}}$ ensures that we add to the sum if and only if the counter increases. Since at each time, each customer can only choose one product to rent, and the states $s_{i,j,l}$ are binary variables taking values in $\{0,1\}$, the counter can only increase by 1 each time if $s_{i,j,l}>0$. Therefore, the sum is equivalent to summing up the function series $\frac{1}{\sqrt{t}}$, with t ranging from 1 to $\max\{N_{i,j,l}^{K-1},1\}$. When $N_{i,j,l}^{K-1}<1$, this term equals to 0. The second inequality is due to the fact that $\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\leq 2\sqrt{T}$.

By combining f_1 and f_3 together, and exchanging the order of summation $(\sum_{k=1}^{K}\sum_{h=1}^{H})$ and $(\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=0}^{D_i})$, according to Fubini's theory for non-negative series, we have:

$$\begin{aligned}
f_1+f_3&\leq 4\max_k\{B_k(H)\}\sqrt{\log(2D_{max}MNT^2/\delta)}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=0}^{D_i}\sum_{t=1}^{\max\{N_{i,j,l}^{K-1},1\}}\frac{1}{\sqrt{t}}\\
&\leq 4\max_k\{B_k(H)\}\sqrt{\log(2D_{max}MNT^2/\delta)}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=0}^{D_i}2\sqrt{\max\{N_{i,j,l}^{K-1},1\}}\\
&\leq 8\max_k\{B_k(H)\}\sqrt{\log(2D_{max}MNT^2/\delta)}\cdot(NMD_{max})\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=0}^{D_i}\max\{N_{i,j,l}^{K-1},1\}}{NMD_{max}}}\\
&\leq 8\max_k\{B_k(H)\}\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2T}.
\end{aligned}$$

The first inequality is because we double the constant parameter of f_1 to match the constant parameter of f_3, and take maximum of the term $B_k(H)$. The second inequality is due to (108). The third inequality is because $\sqrt{x}$ is a concave function, and we apply Jensen's inequality here. The last inequality is because at each time step, we can increase the sum of all counters by at most $D_{max}=\max_i\{D_i+1\}$. Because there are at most D_{max} products remain borrowed at each time, we achieve a $\tilde{O}(HD_{max}\sqrt{MNT})$ term.

Similarly, we have:

$$\begin{aligned}
f_2+f_4&\leq 2\max_k\{B_k(H)\}\sqrt{\log(2D_{max}NT^2/\delta)}\sum_{i=1}^{N}\sum_{l=1}^{D_i+1}2\sqrt{\max\{\tilde{N}_{i,l},1\}}\\
&\leq 4\max_k\{B_k(H)\}\sqrt{\log(2D_{max}NT^2/\delta)}\sqrt{ND_{max}^2T}.
\end{aligned}$$

This is a $\tilde{O}(HD_{max}\sqrt{NT})$ term, which is smaller because it is independent of the dimension M. Therefore, the sum of the four terms is smaller than:

$$12\max_k\{B_k(H)\}\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2T}.$$

□

- **Proof of Theorem 5.1**:

Proof. Since all the value functions are non-negative and are bounded by $B_{max}(H)$, We have that the regret (44) is smaller than:

$$\begin{aligned} &P(\neg\mathcal{E}_{clean})\sum_{k=1}^{K}2B_k(H) + P(\mathcal{E}_{clean})\sum_{k=1}^{K}[\hat{V}_1^k(\mathbf{s}_1^k) - \tilde{V}_1^k(\mathbf{s}_1^k) + \bar{U}_1^k(\mathbf{s}_1^k) - V_1^{\pi_k}(\mathbf{s}_1^k)] \\ &\leq 2\delta B_{aver} + (1-\frac{\delta}{T})\sum_{k=1}^{K}\sum_{h=1}^{H}2\zeta_{h+1}^k + 2E_{\pi^k,\hat{\mathbb{P}}}[\Gamma_h^k(\mathbf{s}_h^k,a_h^k)] + E_{\tilde{\mathbb{P}}'}[\Gamma_h^k(\mathbf{s}_h^k,a_h^k)] \end{aligned} \tag{109}$$

The notation ζ_{h+1}^k is a martingale difference sequence with $|\zeta_h^k| \leq 2B_k(H)$. Therefore, by applying Azuma-Hoeffding inequality in lemma D.1, we know that:

$$P(\sum_{k=1}^{K}\sum_{h=1}^{H}\zeta_h^k \geq t) \leq \exp\Big(-\frac{t^2}{2B_k(H)^2T}\Big) \leq \exp\Big(-\frac{t^2}{2\max_k B_k(H)^2T}\Big) \tag{110}$$

By taking $t = \sqrt{2B_{max}(H)^2T\log(\frac{2T}{\delta})}$, we know with probability is $1-\frac{\delta}{2T}$, this yields a $O(\sqrt{T}\log(T))$ term and with probability $\frac{\delta}{2T}$, this yields a constant term B_{aver}.

Now we proceed to bound the main part consisted of the bonus functions. According to Lemma B.1, we know that for arbitrary sequences, the sum of Γ_h^k functions safisfies (106). Therefore, by taking expectation, we still have that the sum is no larger than $12B_{max}(H)\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2T}$. Finally, we have that (109) is smaller than

$$\begin{aligned} &2\delta B_{aver} + 2B_{max}(H)\sqrt{2T\log(\frac{2T}{\delta})} + 36B_{max}(H)\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2T} \\ &\leq 2\delta B_{aver} + 40B_{max}(H)\sqrt{\log(2D_{max}MNT^2/\delta)}\sqrt{NMD_{max}^2T} \end{aligned}$$

□

Appendix C Bonus Reduction

To further minimize the bonus function, it is worth noting that, according to the "Optimism in the face of uncertainty" (OPFU) rule, the optimal action is to select the option that maximizes the best-case reward within the uncertainty set. This can be expressed as:

$$\begin{aligned} &\max_{(i,j)\in\mathcal{A}}\max_{\tilde{r}\in H_r,\tilde{q}\in H_q} 1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-\tilde{q}_{i,j,0})\big\{(p_j + \tilde{r}_{i,1} + (1-\tilde{q}_{i,j,1})E[\hat{V}_{h+1}^k(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})] \\ &+ \tilde{q}_{i,j,1}E[\hat{V}_{h+1}^k(\tilde{\mathbf{S}}_{h+1})]\big\} + (1 - 1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-\tilde{q}_{i,j,0}))E[\hat{V}_{h+1}^k(\tilde{\mathbf{S}}_{h+1})] + \sum_{i=1}^{N}\sum_{l=1}^{D_i}\tilde{r}_{i,l+1}(\sum_{j=1}^{M}(\mathbf{s}_h)_{i,j,l}) \end{aligned} \tag{111}$$

where $H_r := \{\tilde{r}\in[0,1] : |\tilde{r}-\hat{r}| \leq rad(r)\}$ and $H_q := \{\tilde{q}\in[0,1] : |\tilde{q}-\hat{q}| \leq rad(q)\}$, representing the confidence interval for reward and transition parameters. The value function $\hat{V}$ is obtained by performing backward induction in Algorithm 1.

This value function is contained in the functional class $\mathcal{V}$:

$$\mathcal{V} := \{\hat{V}_h(\cdot) = \mathbf{w}_h^\top \psi(\cdot) : \mathbf{w} \text{ obtained by putting } \tilde{r}, \tilde{q} \text{ into Algorithm 1 with } \tilde{r} \in H_r, \tilde{q} \in H_q\} \quad (112)$$

which contains all linear value functions with parameters within the confidence region. We prove in Lemma D.4 that the maximum value function in this class is bounded by taking empirical estimators $\hat{r}, \hat{q}$ with bonus function $b_h^k(\mathbf{s}, a)$ into the online backward induction procedure in Algorithm 3. This helps to generate $\bar{V}$, an upper bound of the value function. We then put this into (111) and generate an upper bound of the action-value function. Now the problem becomes simpler:

$$\max_{a=(i,j)\in\mathcal{A}} 1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-\hat{q}_{i,j,0})\big\{(p_j + \hat{r}_{i,1} + (1-\hat{q}_{i,j,1})E[\bar{V}_{h+1}^k(\hat{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1})] + \hat{q}_{i,j,1}E[\bar{V}_{h+1}^k(\tilde{\mathbf{S}}_{h+1}))]\big\}$$
$$+ (1 - 1_{\{(\mathbf{s}_h)_{i,0}>0\}}(1-\hat{q}_{i,j,0}))E[\bar{V}_{h+1}^k(\tilde{\mathbf{S}}_{h+1})] + \sum_{i=1}^{N}\sum_{l=1}^{D_i}\hat{r}_{i,l+1}(\sum_{j=1}^{M}(\mathbf{s}_h)_{i,j,l}) + b_h^k(\mathbf{s}, a)$$

Following the OPFU rule, we can use this upper bound action-value function to solve the outer maximization problem and get the optimal action. According to Lemma D.4, we show that when $(\max\{p_{max}+1, D_{max}\}) < 2B_k(H)$, the bonus term $b_h^k(\mathbf{s}, a)$ is strictly smaller than $\Gamma_h^k(\mathbf{s}, a)$. Thus, the bonus term can be finally reduced from $\Gamma_h^k(\mathbf{s}, a)$ to $b_h^k(\mathbf{s}, a)$. The use of this bonus function helps the algorithm exploit and converge earlier without compromising overall performance.

Appendix D Auxilliary Lemmas

Lemma D.1. *(Azuma-Hoeffding) Wainwright [2019] Let $\{(D_k, \mathcal{F}_k)\}_{k=1}^{\infty}$ be a martingale difference sequence for which there are constants $\{(a_k, b_k\}_{k=1}^n$ such that $D_k \in [a_k, b_k]$ almost surely for all $k = 1, ..., n$, then $\forall t \geq 0$:*

$$P(|\sum_{k=1}^{n} D_k| \geq t) \leq 2e^{-\frac{2t^2}{\sum_{k=1}^n (b_k - a_k)^2}} \quad (113)$$

Lemma D.2. *(self-normalizing concentration inequality) Abbasi-Yadkori et al. [2011] Let $\{\mathcal{F}_q\}_{q=1}^T$ be a filtration. Let $\{\xi_q\}_{q=1}^T$ be a real-valued stochastic process such that ξ_q is $\mathcal{F}_q$ measurable and conditionally R sub-Gaussian, i.e., for all $\lambda \geq 0$, it holds that $E[e^{\lambda\xi_q}|\mathcal{F}_{q-1}] \leq e^{\lambda^2 R^2/2}$. Let $\{Y_q\}_{q=1}^T$ be a non-negative real-valued stochastic process such that Y_q is $\mathcal{F}_{q-1}$- measurable. For any $\delta' \in (0,1)$, it holds that*

$$P\Big(\frac{\sum_{q=1}^t \xi_q Y_q}{\max\{1, \sum_{q=1}^t Y_q^2\}} \leq 2R\sqrt{\frac{log(T/\delta')}{\max\{1, \sum_{q=1}^t Y_q^2\}}}, \forall t \in [T]\Big) \geq 1 - \delta'$$

Lemma D.3. *For the bonus function $\Gamma_h^k(\cdot,\cdot)$ defined in (37), under the clean event $\mathcal{E}_{clean}$, we have for any function $V_{h+1}(\cdot) \leq B_k(H)$:*

$$|\hat{r}_h^k - r_h| + |(\hat{\mathbb{P}}_h^k - \mathbb{P}_h)V_{h+1}(\cdot,\cdot)| \leq \Gamma_h^k(\cdot,\cdot), \quad (114)$$

and

$$|\hat{r}_h^k - r_h| + |(\tilde{\mathbb{P}}_h'^k - \tilde{\mathbb{P}}_h)V_{h+1}^k(\cdot,\cdot)| \leq \Gamma_h^k(\cdot,\cdot). \tag{115}$$

Proof. The expected reward $\hat{\mathbb{P}}_h^k V_{h+1}(\cdot,\cdot)$ can be viewed as the inner product of the transition kernel $\hat{P}_h^k(\cdot \mid \mathbf{s}_t = \mathbf{s}, a_t = a)$ and the value function $V_{h+1}(\cdot)$, and can be written as:

$$\left\langle \hat{P}_h^k(\cdot \mid \mathbf{s}_t = \mathbf{s}, a_t = a), V_{h+1}(\cdot) \right\rangle.$$

According to Hölder's inequality, the second term in (114) is smaller than:

$$||\hat{P}_h^k(\cdot|\mathbf{s}_t = \mathbf{s}, a_t = a) - P_h(\cdot|\mathbf{s}_t = \mathbf{s}, a_t = a)||_1 ||V_{h+1}||_\infty \tag{116}$$

$$\leq ||\hat{P}_h^k(\cdot|\mathbf{s}_t = \mathbf{s}, a_t = a) - P_h(\cdot|\mathbf{s}_t = \mathbf{s}, a_t = a)||_1 B_k(H). \tag{117}$$

Since the transition of each product is independent, the transition kernel can be represented by multiplying the transition probability of each product indexed by $1, ..., C_k$. Suppose the next state is $\mathbf{s}'$, and the next state of each product i is $x_i' \in \mathcal{X}_i$, then it is obvious that $|\mathcal{S}_k| \leq |\prod_{i=1}^{C_k} \mathcal{X}_i|$, and each state $\mathbf{s}'$ can be mapped into multiple states in the latter space. Suppose we take one mapping and decompose the whole state into the state of each product x_i. For the sake of brevity, we next denote the conditional probability $P_h(x_{i,t+1} = x_i'|x_{i,t} = x_i, a_t = a)$ as $p_i(x_i')$, and $\hat{P}_h^k(x_{i,t+1} = x_i'|x_{i,t} = x_i, a_t = a)$ as $\hat{p}_i^k(x_i')$. We have that $||\hat{P}_h^k(\cdot|\mathbf{s}_t = \mathbf{s}, a_t = a) - P_h(\cdot|\mathbf{s}_t = \mathbf{s}, a_t = a)||_1$ is smaller than :

$$\sum_{\mathbf{s}' \in \mathcal{S}_k} |\prod_{i=1}^{C_k} p_i(x_i') - \prod_{i=1}^{C_k} \hat{p}_i^k(x_i')| \tag{118}$$

$$\leq \sum_{\mathbf{x}' \in \prod_{i=1}^{C_k} \mathcal{X}_i} |\prod_{i=1}^{C_k} p_i(x_i') - \prod_{i=1}^{C_k} \hat{p}_i^k(x_i')| \tag{119}$$

$$= \sum_{\mathbf{x}' \in \prod_{i=1}^{C_k} \mathcal{X}_i} |p_1(x_1') - \hat{p}_1^k(x_1') \prod_{i=2}^{C_k} p_i(x_i') + \hat{p}_1^k(x_1')(p_2(x_2') - \hat{p}_2^k(x_2')) \prod_{i=3}^{C_k} p_i(x_i') + ...$$

$$+ \prod_{i=1}^{C_k - 1} \hat{p}_i^k(x_i')(p_{C_k}(x_{C_k}') - \hat{p}_{C_k}^k(x_{C_k}')| \tag{120}$$

$$\leq \sum_{\mathbf{x}' \in \prod_{i=1}^{C_k} \mathcal{X}_i} |p_1(x_1') - \hat{p}_1^k(x_1')| \prod_{i=2}^{C_k} p_i(x_i') + |p_2(x_2') - \hat{p}_2^k(x_2')| \hat{p}_1^k(x_1') \prod_{i=3}^{C_k} p_i(x_i') + ...$$

$$\prod_{i=1}^{C_k - 1} \hat{p}_i^k(x_i')|(p_{C_k}(x_{C_k}') - \hat{p}_{C_k}^k(x_{C_k}')| \tag{121}$$

$$= \sum_{x_1' \in \mathcal{X}_1} |p_1(x_1') - \hat{p}_1^k(x_1')| + \sum_{x_2' \in \mathcal{X}_2} |p_2(x_2') - \hat{p}_2^k(x_2')| + ... + \sum_{x_{C_k}' \in \mathcal{X}_{C_k}} |p_{C_k}(x_{C_k}') - \hat{p}_{C_k}^k(x_{C_k}')|.$$

(122)

The second inequality is due to the triangle inequality, and the last equality is due to the law of total

probability. Note that all the variables x_i' are Bernoulli random variables that can only take two states, and the parameter is related to the current state of the i^{th} product. We can obtain that for products in state $s_{i,j,l}$, the gap is smaller than $2|q_{i,j,l+1} - \hat{q}^k_{i,j,l+1}|$, which, under the clean event $\mathcal{E}_{clean}$, is smaller than $2rad_k(q_{i,j,l+1})$ for each product. By summing all the products in state $s_{i,j,l}$, and combining (116), we know this part generates a bonus of $2s_{i,j,l}B_k(H)rad_k(q_{i,j,l+1})$. By summing each component of the state, this part is bounded by $\Gamma^k_{1,h}$.

The special case is when we take action $a = (i, j)$, and for a product in state $s_{i,0}$, it will transition to $s_{i,j,1}$ with probability $(1 - q_{i,j,0})(1 - q_{i,j,1})$. For this case:

$$
\begin{aligned}
&|(1 - q_{i,j,0})(1 - q_{i,j,1}) - (1 - \hat{q}^k_{i,j,0})(1 - \hat{q}^k_{i,j,1})| \\
&= |(\hat{q}^k_{i,j,0} - q_{i,j,0}) + (\hat{q}^k_{i,j,1} - q_{i,j,1}) + (q_{i,j,0}q_{i,j,1} - \hat{q}^k_{i,j,0}\hat{q}^k_{i,j,1})| \\
&\leq rad_k(q_{i,j,0}) + rad_k(q_{i,j,1}) + |(q_{i,j,0} - \hat{q}^k_{i,j,0})q_{i,j,1} + \hat{q}^k_{i,j,0}(q_{i,j,1} - \hat{q}^k_{i,j,1})| \\
&\leq rad_k(q_{i,j,0}) + rad_k(q_{i,j,1}) + |(q_{i,j,0} - \hat{q}^k_{i,j,0})| + |(q_{i,j,1} - \hat{q}^k_{i,j,1})| \\
&\leq 2(rad_k(q_{i,j,0}) + rad_k(q_{i,j,1})).
\end{aligned}
$$

We can further separate the error gap caused by products already been rented and the current action as follows:

$$
\begin{aligned}
&|\sum_{\mathbf{s}'\in\mathcal{S}}(P^k_h(\mathbf{s}'|\mathbf{s}, a) - \hat{P}_h(\mathbf{s}'|\mathbf{s}, a))V(\mathbf{s}')| = \\
&|\sum_{\tilde{\mathbf{S}}\in\mathcal{S}}(P_h(\tilde{\mathbf{S}}|\mathbf{s}) - \hat{P}^k_h(\tilde{\mathbf{S}}|\mathbf{s}))[V(\tilde{\mathbf{S}}) + \sum_{\mathbf{s}'\in\mathcal{S}}(P_h(\mathbf{s}'|\tilde{\mathbf{S}}, a) - \hat{P}^k_h(\mathbf{s}'|\tilde{\mathbf{S}}, a))(V(\mathbf{s}') - V(\tilde{\mathbf{S}})]| \\
&\leq ||P_h(\cdot|\mathbf{s}) - \hat{P}^k_h(\cdot|\mathbf{s})||_1||V||_\infty + E_{\tilde{\mathbf{S}}}[||P_h(\cdot|\tilde{\mathbf{S}}, a) - \hat{P}^k_h(\cdot|\tilde{\mathbf{S}}, a)||_1||(V(\cdot) - V(\tilde{\mathbf{S}})||_\infty] \\
&\leq ||P_h(\cdot|\mathbf{s}) - \hat{P}^k_h(\cdot|\mathbf{s})||_1||V||_\infty + 4(rad_k(q_{i,j,0}) + rad_k(q_{i,j,1}))B_k(H) = \Gamma^k_h(\mathbf{s}, a)
\end{aligned}
$$

where the first equality holds because (8) is equivalent to (7) and the event of returning rented product and customer accepting new product are independent. Under the value function expression with opportunity cost, the transition probability can be decomposed to $P(\tilde{\mathbf{S}}|\mathbf{s})$ and $P(\mathbf{s}'|\tilde{\mathbf{S}}, a)$. The second line is derived by directly writing out the expectation in a summation form in (8). The first inequality holds due to triangle inequality and Hölder inequality. The second inequality holds as when $\mathbf{s}' = \tilde{\mathbf{S}}$, $V(\mathbf{s}') - V(\tilde{\mathbf{S}}) = 0 \leq 2B_k(H)$ and when $\mathbf{s}' = \tilde{\mathbf{S}} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1}$, we still have that $|V(\mathbf{s}') - V(\tilde{\mathbf{S}})| \leq 2B_k(H)$. So it also holds by taking expection. The result is proved. □

Lemma D.4. *For all value function $\hat{V}$ in the function class $\mathcal{V}$ defined in (112), we have $|(\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1}))| \leq \max\{p_{max} + 1, D_{max}\}$. It suffices to take $b^k_h(\mathbf{s}, a)$ as the bonus function, which is smaller than $\Gamma^k_h(\mathbf{s}, a)$ when $\max\{p_{max} + 1, D_{max}\} < 2B_k(H)$*

Proof. (1) First, we prove that the opportunity cost for linear function approximation is bounded by $\max\{p_{max} + 1, D_{max}\}$. For the modified kernel, $\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1}) = w^{h+1}_{i,0} - w^{h+1}_{i,j,1}$. From the iteration in (11), we have

$$
w^h_{i,0} = w^{h+1}_{i,0} + \max\{\frac{1}{C_i}1_{\{i=\hat{i}_h\}}(1 - q_{i,\hat{j}_h,0})(p_{\hat{j}_h} + r_{i,1} - (1 - q_{i,\hat{j}_h,1})(w^{h+1}_{i,0} - w^{h+1}_{i,\hat{j}_h,1})), 0\}
$$

By treating the null action as $\hat{i}_h = 0, \hat{j}_h = 0$, we can omit the maximum operator. Since $w^{h+1}_{i,j,1} \leq w^h_{i,j,1}$ according to Proposition 4.1, by subtracting $w^h_{i,j,1}$ on the left and $w^{h+1}_{i,j,1}$ on the right, we obtain:

$$w^h_{i,0} - w^h_{i,j,1} \leq w^{h+1}_{i,0} - w^{h+1}_{i,j,1} + \frac{1}{C_i} 1_{\{i=\hat{i}_h\}} (1 - q_{i,\hat{j}_h,0})(p_{\hat{j}_h} + r_{\hat{i}_h,1} - (1 - q_{\hat{i}_h,\hat{j}_h,1})(w^{h+1}_{\hat{i}_h,0} - w^{h+1}_{\hat{i}_h,\hat{j}_h,1})) \tag{123}$$

$$\leq (1 - \frac{1}{C_i} 1_{\{i=\hat{i}_h\}} (1 - q_{i,\hat{j}_h,0})(1 - q_{\hat{i}_h,\hat{j}_h,1})) \max_{i,j} (w^{h+1}_{i,0} - w^{h+1}_{i,j,1}) + \frac{1}{C_i} 1_{\{i=\hat{i}_h\}} (1 - q_{i,\hat{j}_h,0})(p_{max} + 1) \tag{124}$$

for every (i, j) pair. For the second inequality, note that when $(i, j) \neq (\hat{i}_h, \hat{j}_h)$, we have $w^h_{i,0} - w^h_{i,j,1} \leq w^{h+1}_{i,0} - w^{h+1}_{i,j,1} \leq \max_{i,j}(w^{h+1}_{i,0} - w^{h+1}_{i,j,1})$ and the inequality holds naturally. Otherwise, we have:

$$\begin{aligned}
&w^{h+1}_{\hat{i}_h,0} - w^{h+1}_{\hat{i}_h,\hat{j}_h,1} + \frac{1}{C_i} 1_{\{i=\hat{i}_h\}} (1 - q_{i,\hat{j}_h,0})(p_{\hat{j}_h} + r_{\hat{i}_h,1} - (1 - q_{\hat{i}_h,\hat{j}_h,1})(w^{h+1}_{\hat{i}_h,0} - w^{h+1}_{\hat{i}_h,\hat{j}_h,1})) \\
&= (1 - \frac{1}{C_{\hat{i}_h}} (1 - q_{i,\hat{j}_h,0})(1 - q_{\hat{i}_h,\hat{j}_h,1}))(w^{h+1}_{\hat{i}_h,0} - w^{h+1}_{\hat{i}_h,\hat{j}_h,1}) + \frac{1}{C_{\hat{i}_h}} (1 - q_{\hat{i}_h,\hat{j}_h,0})(p_{\hat{j}_h} + r_{\hat{i}_h,1}) \\
&\leq (1 - \frac{1}{C_{\hat{i}_h}} (1 - q_{i,\hat{j}_h,0})(1 - q_{\hat{i}_h,\hat{j}_h,1})) \max_{i,j} (w^{h+1}_{\hat{i}_h,0} - w^{h+1}_{\hat{i}_h,\hat{j}_h,1}) + \frac{1}{C_{\hat{i}_h}} (1 - q_{\hat{i}_h,\hat{j}_h,0})(p_{max} + 1)
\end{aligned}$$

Since the inequality (124) applies for arbitrary (i, j), we can take maximum over the left hand side of (123) to get an iteration form. For brevity, denote $f_{i,h} = \frac{1}{C_i} 1_{\{i=\hat{i}_h\}} (1 - q_{i,\hat{j}_h,0})$, and by repeating this process iteratively, we get:

$$\begin{aligned}
\max_{(i,j)} (w^h_{i,0} - w^h_{i,j,1}) \leq [&f_{i,h} + (1 - f_{i,h}(1 - q_{i,\hat{j}_h,1})) f_{i,h+1} + ... \\
&+ \prod_{t=h}^{H-1} (1 - f_{i,t}(1 - q_{i,\hat{j}_t,1})) \cdot f_{i,H}](p_{max} + 1) \leq p_{max} + 1
\end{aligned} \tag{125}$$

Following a similar iteration process, by subtracting $w^h_{i,0}$ from both sides in (12), we get to bound $w^h_{i,0} - w^h_{i,j,1}$ from below:

$$\begin{aligned}
&w^h_{i,j,1} - w^h_{i,0} \leq r_{i,2} + (1 - q_{i,j,2})(w^{h+1}_{i,j,2} - w^{h+1}_{i,0}) \\
&= r_{i,2} + (1 - q_{i,j,2}) r_{i,3} + (1 - q_{i,j,2})(1 - q_{i,j,3})(w^{h+1}_{i,j,3} - w^{h+1}_{i,0}) ... \\
&\leq 1 + (1 - q_{i,j,2}) + (1 - q_{i,j,2})(1 - q_{i,j,3}) ... + \prod_{l=2}^{D_i+1} (1 - q_{i,j,l}) \leq D_i + 1 \leq D_{max}
\end{aligned}$$

With the upper and lower bound of $w^h_{i,0} - w^h_{i,j,1}$, we know that $|(\hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1}) - \hat{V}_{h+1}(\tilde{\mathbf{S}}_{h+1} - \mathbf{e}_{i,0} + \mathbf{e}_{i,j,1}))| \leq \max\{p_{max} + 1, D_{max}\}$ holds.

(2) Now we prove that it suffices to take $b^k_h(\mathbf{s}, a)$ as the bonus function. To prove this, we are actually requiring to get an upper bound of the value function class mentioned in (112):

$$\mathcal{V} := \{\hat{V}_h(\cdot) = \mathbf{w}_h^\top \psi(\cdot) : \mathbf{w} \text{ obtained by putting } \tilde{r}, \tilde{q} \text{ into Algorithm 1 with } \tilde{r} \in H_r, \tilde{q} \in H_q\}$$

where H_r, H_q are the confidence interval of reward and rental time distribution as mentioned in C. Denote the set of all the weight vectors of this value function class as $\mathcal{W}$. At each step, we hope this

bonus function to compensate for the loss of value function. It suffices to compute an upper bound of each element of the real weight vector $\mathbf{w}$. We prove this by induction. We start from $h = H + 1$, in this case, we have $\bar{w}^h_{i,j,l} = w^h_{i,j,l} = 0$ and $\bar{w}^h_{i,0} = w^h_{i,0} = 0$ for all (i, j, l) pairs. This satisfies the requirement that every element of $\bar{w}_h$ is no smaller than all possible $w_h \in \mathcal{W}$. Now assume that $\bar{w}_t$ is no smaller than all possible $w_t \in \mathcal{W}$ for $t = h + 1$, we now prove that this holds when $t = h$.

Recall that $\bar{w}^h_{i,j,l} = \min\{\Lambda_i(H-h+1), \hat{r}^k_{i,l+1} + \hat{q}^k_{i,j,l+1}\bar{w}^{h+1,k}_{i,0} + (1-\hat{q}^k_{i,j,l+1})\bar{w}^{h+1,k}_{i,j,l+1} + b^k_h(i,j,l)\}$. When $\bar{w}^h_{i,j,l} = \Lambda_i(H-h+1)$, it reaches the universal upper bound of the weight parameter, and the inequality holds naturally. Otherwise, we need to show that:

$$\max_{\tilde{r}\in H_r, \tilde{q}\in H_q} \max_{\mathbf{w}\in\mathcal{W}} (\tilde{r}_{i,l+1} + \tilde{q}_{i,j,l+1} w^{h+1}_{i,0} + (1-\tilde{q}_{i,j,l+1}) w^{h+1}_{i,j,l+1}) \leq \bar{w}^h_{i,j,l} \tag{126}$$

The inner maximum value on the left hand side is smaller than $(\tilde{r}_{i,l+1} + \tilde{q}_{i,j,l+1}\bar{w}^{h+1}_{i,0} + (1-\tilde{q}_{i,j,l+1})\bar{w}^{h+1}_{i,j,l+1})$, $\forall \tilde{r}, \tilde{q} \in H_q, H_r$. Therefore, we only need to prove that $\max_{\tilde{r}\in H_r, \tilde{q}\in H_q}(\tilde{r}_{i,l+1} + \tilde{q}_{i,j,l+1}\bar{w}^{h+1}_{i,0} + (1 - \tilde{q}_{i,j,l+1})\bar{w}^{h+1}_{i,j,l+1}) \leq \bar{w}^h_{i,j,l}$.

Since $\tilde{r}_{i,l+1} \leq \hat{r}_{i,l+1} + rad(r_{i,l+1})$ and

$$\begin{aligned}
&\tilde{q}_{i,j,l+1}\bar{w}^{h+1}_{i,0} + (1-\tilde{q}_{i,j,l+1})\bar{w}^{h+1}_{i,j,l+1} = \tilde{q}_{i,j,l+1}(\bar{w}^{h+1}_{i,0} - \bar{w}^{h+1}_{i,j,l+1}) + \bar{w}^{h+1}_{i,j,l+1} \\
&\leq \hat{q}_{i,j,l+1}(\bar{w}^{h+1}_{i,0} - \bar{w}^{h+1}_{i,j,l+1}) + \bar{w}^{h+1}_{i,j,l+1} + |rad(q_{i,j,l+1})(\bar{w}^{h+1}_{i,0} - \bar{w}^{h+1}_{i,j,l+1})|
\end{aligned}$$

Summing these together, we can prove that the left hand side of (126) is smaller than

$$\begin{aligned}
&\hat{r}_{i,l+1} + rad(r_{i,l+1}) + \hat{q}_{i,j,l+1}(\bar{w}^{h+1}_{i,0} - \bar{w}^{h+1}_{i,j,l+1}) + \bar{w}^{h+1}_{i,j,l+1} + |rad(q_{i,j,l+1})(\bar{w}^{h+1}_{i,0} - \bar{w}^{h+1}_{i,j,l+1})| \\
&= \hat{r}_{i,l+1} + \hat{q}_{i,j,l+1}(\bar{w}^{h+1}_{i,0} - \bar{w}^{h+1}_{i,j,l+1}) + \bar{w}^{h+1}_{i,j,l+1} + b_h(i,j,l) = \bar{w}^h_{i,j,l}
\end{aligned}$$

as indicated by the backward induction update formula in (36) Similarly, we can prove that:

$$\max_{\tilde{r}\in H_r, \tilde{q}\in H_q} \max_{\mathbf{w}\in\mathcal{W}} \Big\{ w^{h+1,k}_{i,0} + 1_{\{i=i'_h\}} \frac{1}{C^k_i} [(1-\tilde{q}^k_{i,j'_h,0})(p_{j'_h} + \tilde{r}^k_{i,1} - (1-\tilde{q}^k_{i,j'_h,1})(w^{h+1,k}_{i,0} - w^{h+1,k}_{i,j'_h,1}))] \Big\} \leq \bar{w}^h_{i,0} \tag{127}$$

First, it is easy to verify that the left hand side is an increasing function of $\mathbf{w}^{h+1}$, and is smaller than $\bar{w}^{h+1,k}_{i,0} + 1_{\{i=i'_h\}} \frac{1}{C^k_i} [(1-\tilde{q}^k_{i,j'_h,0})(p_{j'_h} + \tilde{r}^k_{i,1} - (1-\tilde{q}^k_{i,j'_h,1})(\bar{w}^{h+1,k}_{i,0} - \bar{w}^{h+1,k}_{i,j'_h,1}))]$, $\forall \tilde{r}, \tilde{q} \in H_q, H_r$.

Following the similar step, we can show that the this term is smaller than:

$$\begin{aligned}
&\bar{w}^{h+1,k}_{i,0} + 1_{\{i=i'_h\}} \frac{1}{C^k_i} [(1-\hat{q}^k_{i,j'_h,0})(p_{j'_h} + \hat{r}^k_{i,1} + rad(r_{i,1}) - (1-\hat{q}^k_{i,j'_h,1})(\bar{w}^{h+1,k}_{i,0} - \bar{w}^{h+1,k}_{i,j'_h,1})) + \\
&2(rad_k(q_{i',j',0}) + rad_k(q_{i',j',1}))|(\bar{w}^{h+1,k}_{i,0} - \bar{w}^{h+1,k}_{i,j,1})|] = \bar{w}^h_{i,0}
\end{aligned}$$

as indicated by the update formula in (35). Therefore, by induction, we can prove that $\bar{\mathbf{w}}$ is larger than any $\mathbf{w} \in \mathcal{W}$ elementwise. This proves that the linear function approximation $\tilde{V}(\cdot)$ is an upper bound of all the value functions in $\mathcal{V}$, and it suffices to take $b_h(\mathbf{s}, a)$ as the bonus function. □

Lemma D.5. *Define the event that* $\{|r_{i,l} - \hat{r}^k_{i,l}| \leq rad_k(r_{i,l}), \forall i \in \mathcal{I}, l \in [D_i + 1], k \in [K]\}$ *as* $\mathcal{E}_r$*, then* $P(\mathcal{E}_r) \geq 1 - \frac{\delta}{2T}$*. Similarly, define the event that* $\{|q_{i,j,l} - \hat{q}^k_{i,j,l}| \leq rad_k(q_{i,j,l}) \forall i \in \mathcal{I}, j \in \mathcal{J}, l = 0, ..., D_i\}$ *as* $\mathcal{E}_q$*, then* $P(\mathcal{E}_q) \geq 1 - \frac{\delta}{2T}$*. Furthermore, the probability that the clean event happens is at*

least $1-\frac{\delta}{T}$.

Proof. For an arbitrary $i \in \mathcal{I}, l = 2, ..., D_i + 1$, it holds that

$$P(|r_{i,l} - \hat{r}^k_{i,l}| \geq rad_k(r_{i,l}), \forall k) \tag{128}$$

$$= P\Big(\Big|\frac{1}{\max\{1, \tilde{N}^k_{i,l}\}} \sum_{\tau=1}^{(k-1)} \sum_{h=1}^{H} [R^{\tau,h}_{i,l,m} - r_{i,l}] \cdot \mathbf{1}_{\{\mathbf{s}^\tau_h = \mathbf{s} \wedge m = \sum_{j=1}^M s_{i,j,l-1} > 0\}}\Big| \geq 2\sqrt{\frac{\log(2D_{max}NT^2/\delta)}{\max\{1, \tilde{N}^k_{i,l}\}}}, \forall k\Big). \tag{129}$$

We now show how we apply Lemma D.2 to our problem. We set $q = 1, ..., KH$, and let the filtration $\mathcal{F}_q = \{(\mathbf{s}_t, a_t, R_t\}_{t=1}^q \cup \{(\mathbf{s}_{q+1}, a_{q+1})\}$, and $q = (\tau - 1)H + h$. We let $\xi_q = R^{\tau,h}_{i,l,m} - r_{i,l}$, which is bounded in $[-1, 1]$, and therefore 1-sub-Gaussian. Let $Y_q = \mathbf{1}_{\{\mathbf{s}^\tau_h = \mathbf{s} \wedge m = \sum_{j=1}^M s_{i,j,l-1} > 0\}}$, and it's easy to verify that Y_q is $\mathcal{F}_{q-1}$ measurable. We also have $Y_q^2 = Y_q$, as this is a $\{0, 1\}$ variable. Therefore, by taking $\delta' = \frac{\delta}{2D_{max}NT}$, (129) $\leq \frac{\delta}{2D_{max}NT}$.

Furthermore, for the case $l = 1$, the filtration is changed to $\mathcal{F}'_q = \{(\mathbf{s}_t, a_t, R_t\}_{t=1}^q \cup \{(\mathbf{s}_{q+1}, a_{q+1}, \mathbf{1}_{\{\mathcal{E}^{\tau'}_{h'}\}})\}$, where $q+1 = (\tau'-1)H + h'$. That is, we must observe whether the customer rejects the offer. Note that R_{q+1} actually contains the information of $\mathcal{E}^{\tau'}_{h'}$, which is reflected by whether or not there is an upfront payment. It's easy to verify that $Y_q = \mathbf{1}_{\{a^\tau_h = (i,j) \wedge \neg \mathcal{E}^\tau_h\}}$ is adapted to $\mathcal{F}_{q-1}$, and the other parts remain unchanged.

By taking the union bound, we have:

$$P(\mathcal{E}_r) \geq 1 - \sum_{i=1}^{N} \sum_{l=1}^{D_{max}} \frac{\delta}{2D_{max}NT} = 1 - \frac{\delta}{2T}. \tag{130}$$

Similarly, for the transition probability, it holds that for an arbitrary $i \in \mathcal{I}, j \in \mathcal{J}, l = 2, ..., D_i$.

$$P(|q_{i,j,l} - \hat{q}^k_{i,j,l}| \geq rad_k(q_{i,j,l}), \forall k) \tag{131}$$

$$= P\Big(\Big|\frac{1}{\max\{1, N^k_{i,j,l}\}} \sum_{\tau=1}^{(k-1)} \sum_{h=1}^{H-1} [1 - s'_{i,j,l} - q_{i,j,l} s_{i,j,l-1}] \cdot \mathbf{1}_{\{\mathbf{s}^\tau_h = \mathbf{s} \wedge \mathbf{s}^\tau_{h+1} = \mathbf{s}'\}}\Big|$$

$$\geq 2\sqrt{\frac{\log(2D_{max}MNT^2/\delta)}{\max\{1, N^k_{i,j,l}\}}}, \forall k\Big) \tag{132}$$

$$\leq \frac{\delta}{2D_{max}MNT}, \tag{133}$$

where the corresponding filtration is $\mathcal{F}_q = \{\mathbf{s}_t, a_t\}_{t=1}^{q+1}$, $\xi_q = (1 - s'_{i,j,l} - q_{i,j,l} s_{i,j,l})$ is a 1-sub-Gaussian, and $Y_q = \mathbf{1}_{\{\mathbf{s}^\tau_h = \mathbf{s} \wedge \mathbf{s}^\tau_{h+1} = \mathbf{s}'\}}$ is adapted to $\mathcal{F}_{q-1}$.

Furthermore, for the case $l = 0, l = 1$, we define the filtration $\mathcal{F}'_q = \{\mathbf{s}_t, a_t, \mathcal{E}_t\}_{t=1}^{q+1}$. For $q_{i,j,0}$, we have $\xi_q = \mathbf{1}_{\{\mathcal{E}^\tau_h\}} - q_{i,j,0}$, and $Y_q = \mathbf{1}_{\{a^\tau_h = (i,j)\}}$. For $q_{i,j,1}$, we have $\xi_q = \mathbf{1}_{\{(\mathbf{s}^\tau_{h+1})_{i,j,1} = 0\}} - q_{i,j,0}$, and $Y_q = \mathbf{1}_{\{a^\tau_h = (i,j), \neg \mathcal{E}^\tau_h\}}$, and Lemma D.2 applies. By taking the union bound, we have:

$$P(\mathcal{E}_q) \geq 1 - \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{l=0}^{D_{max}-1} \frac{\delta}{2D_{max}MNT} = 1 - \frac{\delta}{2T}. \tag{134}$$

Therefore, we conclude that the probability that the clean event happens is at least $1 - 2\frac{\delta}{2T} = 1 - \frac{\delta}{T}$ □

Appendix E Further Details of Case Study

Figure 10 shows the normalized reward distribution, where the x axis refers to the time period, and each line refers to the r_{il} distribution of a KOL. The box plot for the posts' return time distribution is shown in

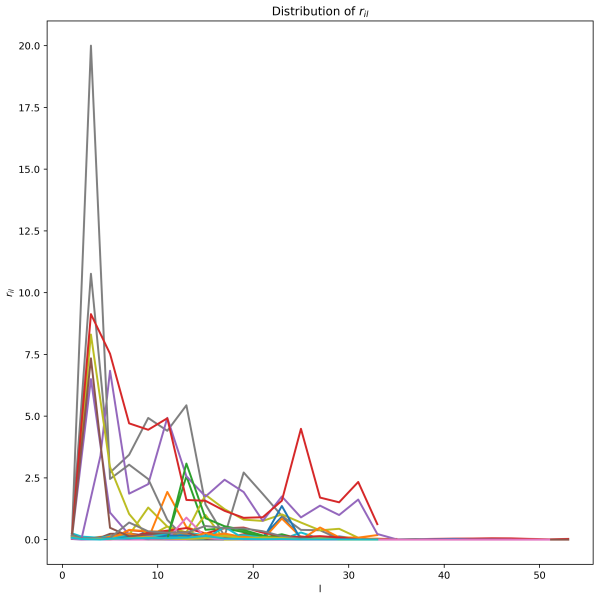


Figure 10: Normalized Reward Distribution

Figure 11. Clearly , the median, the third quartile and the maximum value are quite close for all KOLs, which is consistent with the relative frequency histogram in the main text. Therefore, it is reasonable to take the third quartile as the maximum rental time.

Moreover, Figure 12 verifies that our method indeed achieves sublinear regret, as it slightly bends lower compared to the linear line we get by regression.

Appendix F Extensions

In this section, we give a detailed algorithm on how to extend the algorithm to heterogeneous customer setting with known arrival rate. Assume there are G types of customers indexed by $g = 1, 2, ..., G$, with arrival rate λ_g. For each product i, the state space now extends to $s_i = \{s_{i,0}, s^g_{i,j,l} : l = 1, ..., D_i; j = 1, ..., M; g = 1, ..., G\}$. Morever, the reward of renting a product i to type g customer with price j for l periods becomes $r^g_{i,l}$ and the hazard rate of returning the product becomes $q^g_{i,j,l}$. First, the offline linear function approximation is denoted as:

$$\hat{V}_h(\mathbf{s}) = \sum_{i=1}^{N} (w^h_{i,0} s_{i,0} + \sum_{j=1}^{M} \sum_{g=1}^{G} \sum_{l=1}^{D_i} w^{h,g}_{i,j,l} s^g_{i,j,l})$$

where the weight parameter $w^{h,g}_{i,j,l}$ refers to the unit value of a product i rented to customer type g and at price j for l periods. For the backward induction in Algorithm 1, simply change the iteration form (11)

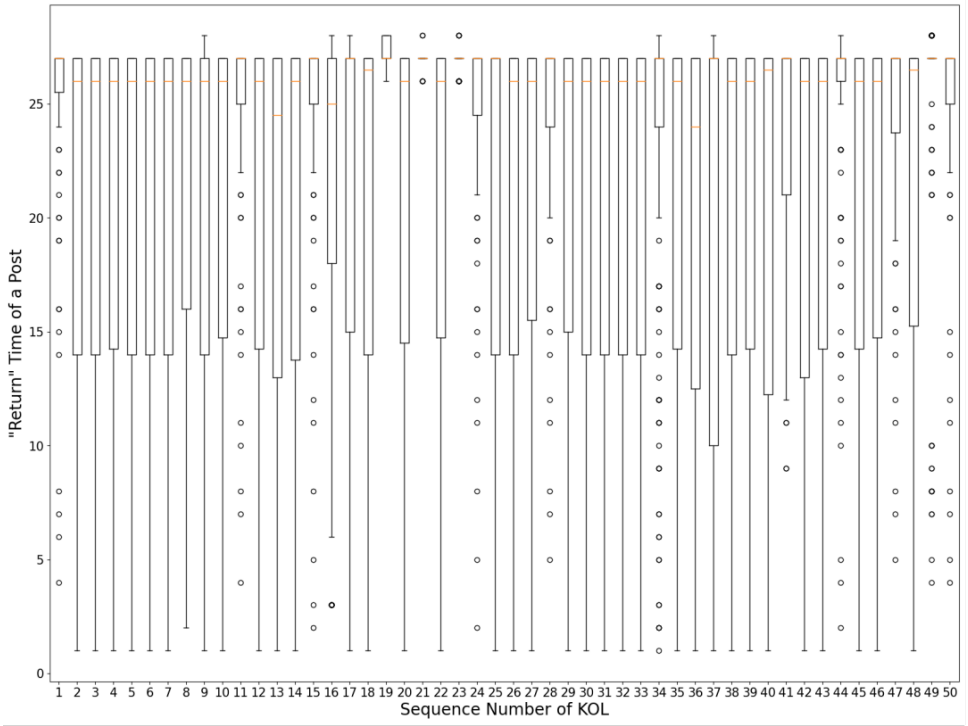


Figure 11: Box Plot of Return Time

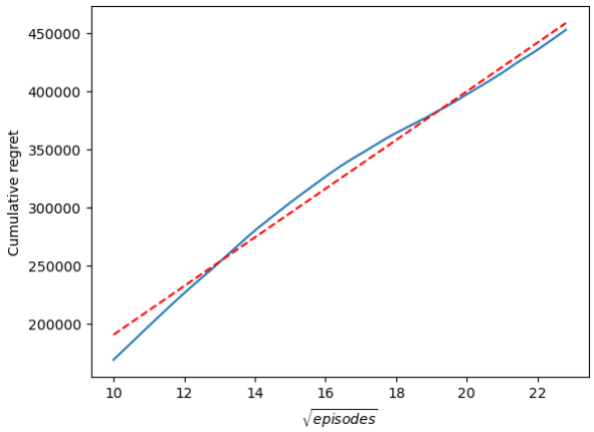


Figure 12: Verification of Sublinear Regret

to:

$$w_{i,0}^{h} = w_{i,0}^{h+1} + \max\{\frac{1}{C_i} E_g\big[1_{\{i=\hat{i}_h\}}(1 - q_{i,\hat{j}_h,0}^{g})(p_{\hat{j}_h} + r_{i,1}^{g} - (1 - q_{i,\hat{j}_h,1}^{g})(w_{i,0}^{h+1} - w_{i,\hat{j}_h,1}^{h+1,g}))\big], 0\} \tag{135}$$

where the expectation is taken with respect to g. Similarly, the iteration form in (12) changes to:

$$w_{i,j,l}^{h,g} = r_{i,l+1}^{g} + q_{i,j,l+1}^{g} w_{i,0}^{h+1} + (1 - q_{i,j,l+1}^{g}) w_{i,j,l+1}^{h+1,g} \tag{136}$$

For the online setting, if the arrival rate is known, then we only need to change the backward induction in Algorithm 3 similar as (135),(136). Otherwise, we also need to establish an empirical estimation

and confidence radius for λ_g.