Chapter 5
Learning and Pricing with Inventory Constraints

Qi (George) Chen, He Wang, and Zizhuo Wang

5.1 Introduction

In this chapter, we consider learning and pricing problems with inventory constraints: given an initial inventory of one or multiple products and finite selling season, a seller must choose prices dynamically to maximize revenue over the course of the season. Inventory constraints are prevalent in many business settings. For most goods and services, there is limited inventory due to supply constraint, sellers' budget constraint, or limited storage space. Therefore, one must consider the impact of inventory constraints when learning demand functions and setting prices.

Dynamic pricing with inventory constraints has been extensively studied in the revenue management literature, often under the additional assumption that the demand function (i.e., the relationship between demand and price) is known to the seller prior to the selling season. However, when the demand function is unknown, the seller faces a trade-off commonly referred to as the *exploration–exploitation trade-off*. Toward the beginning of the selling season, the seller may offer different prices to try to learn and estimate the demand rate at each price ("exploration" objective). Over time, the seller can use these demand rate estimates to set prices that

Q. (George) Chen
London Business School, London, UK
e-mail: gchen@london.edu

H. Wang (✉)
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA
e-mail: he.wang@isye.gatech.edu

Z. Wang
School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
e-mail: wangzizhuo@cuhk.edu.cn

X. Chen et al. (eds.), *The Elements of Joint Learning and Optimization in Operations Management*, Springer Series in Supply Chain Management 18,
https://doi.org/10.1007/978-3-031-01926-5_5

maximize revenue throughout the remainder of the selling season ("exploitation" objective). With limited inventory, pursuing the exploration objective comes at the cost of not only lowering revenue but also diminishing valuable inventory. Simply put, if inventory is depleted while exploring different prices, there is no inventory left to exploit the knowledge gained.

In this chapter, we will study how one should design learning algorithm in the presence of inventory constraints. Specifically, we will study how one can overcome the additional challenges brought forth by the limited inventory and still design efficient algorithms for learning demand functions with regret guarantees. In what follows, we will first discuss the simplest case in this setting in Sect. 5.2, i.e., the learning and pricing problem of a single product with an inventory constraint. Then, in Sect. 5.3, we discuss the problem of learning and pricing with multiple products under inventory constraints. Finally, in Sect. 5.4, we consider a Bayesian learning setting with inventory constraints. In each of the sections, we describe the model and the challenges and then present the algorithms and analysis for respective problems. In Sect. 5.5, we present concluding remarks and some further readings for this chapter.

5.2 Single Product Case

In this section, we consider the problem of a monopolist selling a single product in a finite selling season T. We assume that the seller has a fixed inventory x at the beginning and no replenishment can be made during the selling season. During the selling season, customers arrive according to a Poisson process with an instantaneous demand rate λ_t at time t.[1] In our model, we assume that λ_t is solely dependent on the price the seller offers at time t. That is, we can write $\lambda_t = \lambda(p(t))$, where $p(t)$ is the price offered at time t. The sales will be terminated at time T, and there is no salvage value for the remaining items.

In our model, we assume that the set of feasible prices is an interval $[\underline{p}, \overline{p}]$ with an additional cut-off price p_∞ such that $\lambda(p_\infty) = 0$. The demand rate function $\lambda(p)$ is assumed to be strictly decreasing in p and has an inverse function $p = \gamma(\lambda)$. We define a revenue rate function $r(\lambda) = \lambda\gamma(\lambda)$, which captures the expected revenue when the price is chosen such that the demand is λ. We further assume $r(\lambda)$ is concave in λ. These assumptions on demand functions are quite standard and are called the *regular* assumptions in the revenue management literature (Gallego and van Ryzin, 1994).

In addition to the above, we make the following assumptions on the demand rate function $\lambda(p)$ and the revenue rate function $r(\lambda)$:

Assumption 1 For some positive constants M, K, m_L, and m_U,

[1] Our analysis and result also work if we discretize the time horizon and assume at each time period t, there is a probability λ_t such that a customer arrives.

1. Boundedness: $|\lambda(p)| \leq M$ for all $p \in [\underline{p}, \overline{p}]$.
2. Lipschitz continuity: $\lambda(p)$ and $r(\lambda(p))$ are Lipschitz continuous with respect to p with factor K. Also, the inverse demand function $p = \gamma(\lambda)$ is Lipschitz continuous in λ with factor K.
3. Strict concavity and differentiability: $r''(\lambda)$ exists and $-m_L \leq r''(\lambda) \leq -m_U < 0$ for all λ in the range of $\lambda(p)$ for $p \in [\underline{p}, \overline{p}]$.

In the following, we use $\Gamma = \Gamma(M, K, m_L, m_U)$ to denote the set of demand functions satisfying the above assumptions with the corresponding coefficients. In our model, the seller does not know the true demand function λ. The only knowledge the seller has is that the demand function belongs to Γ. Note that Γ does not need to have any parametric representation. We note that Assumption 1 is quite mild, and it is satisfied for many commonly used demand function classes including linear, exponential, and logit demand functions.

To evaluate the performance of any pricing algorithm, we adopt the minimax regret objective. We call a pricing policy $\pi = (p(t) : 0 \leq t \leq T)$ admissible if (1) it is a non-anticipating price process that is defined on $[\underline{p}, \overline{p}] \cup \{p_\infty\}$ and (2) it satisfies the inventory constraint, that is, $\int_0^T dN^\pi(s) \leq x$, with probability 1, where $N^\pi(t) = N\left(\int_0^t \lambda(p(s))ds\right)$ denotes the cumulative demand up to time t using policy π.

We denote the set of admissible pricing policies by $\mathcal{P}$. We define the expected revenue generated by a policy π by

$$J^\pi(x, T; \lambda) = E\left[\int_0^T p(s)dN^\pi(s)\right]. \tag{5.1}$$

Given a demand rate function λ, there exists an optimal admissible policy π^* that maximizes (5.1). In our model, since we do not know λ in advance, we seek $\pi \in \mathcal{P}$ that performs as close to π^* as possible.

However, even if the demand function λ is known, computing the expected value of the optimal policy is computationally prohibitive. It involves solving a continuous-time dynamic program exactly. Fortunately, as shown in Gallego and van Ryzin (1994), there exists an upper bound for the expected value of any policy based on the following deterministic optimization problem:

$$\begin{aligned} J^D(x, T; \lambda) = \sup & \int_0^T r(\lambda(p(s)))ds \\ \text{s.t.} & \int_0^T \lambda(p(s))ds \leq x \\ & p(s) \in [\underline{p}, \overline{p}] \cup \{p_\infty\}, \quad \forall s \in [0, T]. \end{aligned} \tag{5.2}$$

Gallego and van Ryzin (1994) showed that $J^D(x, T; \lambda)$ provides an upper bound on the expected revenue generated by any admissible pricing policy π, that is,

$J^{\pi}(x, T; \lambda) \le J^{D}(x, T; \lambda)$, for all $\lambda \in \Gamma$ and $\pi \in \mathcal{P}$. With this upper bound, we define the regret $R^{\pi}(x, T; \lambda)$ for any given demand function $\lambda \in \Gamma$ and policy $\pi \in \mathcal{P}$ by

$$R^{\pi}(x, T; \lambda) = 1 - \frac{J^{\pi}(x, T; \lambda)}{J^{D}(x, T; \lambda)}. \tag{5.3}$$

Clearly, $R^{\pi}(x, T; \lambda)$ is always greater than 0. Furthermore, the smaller $R^{\pi}(x, T; \lambda)$ is, the closer the performance of π is to that of the optimal policy. However, since the decision-maker does not know the true demand function, it is attractive to have a pricing policy π that achieves small regrets across all possible underlying demand functions $\lambda \in \Gamma$. To capture this, we consider the worst-case regret. Specifically, the decision-maker chooses a pricing policy π, and the nature picks the worst possible demand function for that policy and our goal is to minimize the worst-case regret:

$$\inf_{\pi \in \mathcal{P}} \sup_{\lambda \in \Gamma} R^{\pi}(x, T; \lambda). \tag{5.4}$$

Unfortunately, it is hard to evaluate (5.4) for any finite size problem. In order to obtain theoretical guarantee of proposed policies, we adopt a widely used asymptotic performance analysis. Particularly, we consider a regime in which both the size of the initial inventory and the demand rate grow proportionally. Specifically, in a problem with size n, the initial inventory and the demand function are given by

$$x_n = nx \text{ and } \lambda_n(\cdot) = n\lambda(\cdot).$$

Define $J_n^{D}(x, T; \lambda) = J^{D}(nx, T, n\lambda) = nJ^{D}(x, T, \lambda)$ to be the optimal value to the deterministic problem with size n and $J_n^{\pi}(x, T; \lambda) = J^{\pi}(nx, T, n\lambda)$ to be the expected value of a pricing policy π when it is applied to a problem with size n. The regret for the size-n problem $R_n^{\pi}(x, T; \lambda)$ is therefore

$$R_n^{\pi}(x, T; \lambda) = 1 - \frac{J_n^{\pi}(x, T; \lambda)}{J_n^{D}(x, T; \lambda)},$$

and our objective is to study the asymptotic behavior of $R_n^{\pi}(x, T; \lambda)$ as n grows large and design an algorithm with small asymptotic regret.

5.2.1 Dynamic Pricing Algorithm

In this section, we introduce a dynamic pricing algorithm, which achieves near-optimal asymptotic regret for the aforementioned problem. To start with, we first consider the full-information deterministic problem (5.2). As shown in Besbes and Zeevi (2009), the optimal solution to (5.2) is given by

$$p(t) = p^D = \max\{p^u, p^c\} \tag{5.5}$$

where

$$p^u = \arg \max_{p \in [\underline{p}, \overline{p}]} \{r(\lambda(p))\}, \quad p^c = \arg \min_{p \in [\underline{p}, \overline{p}]} \left| \lambda(p) - \frac{x}{T} \right|. \tag{5.6}$$

The following important lemma is proved in Gallego and van Ryzin (1994).

Lemma 1 *Let p^D be the optimal deterministic price when the underlying demand function is λ. Let π^D be the pricing policy that uses the deterministic optimal price p^D throughout the selling season until there is no inventory left. Then, $R_n^{\pi^D}(x, T, \lambda) = O(n^{-1/2})$.*

Lemma 1 states that if one knows p^D in advance, then simply applying this price throughout the entire time horizon can achieve asymptotically optimal performance. Therefore, the idea of our algorithm is to find an estimate of p^D that is close enough to the true one efficiently, using empirical observations on hand. In particular, under Assumption 1, we know that if $p^D = p^u > p^c$, then

$$\left| r(p) - r(p^D) \right| \le \frac{1}{2} m_L (p - p^D)^2 \tag{5.7}$$

for p close to p^D, while if $p^D = p^c \ge p^u$, then

$$\left| r(p) - r(p^D) \right| \le K \left| p - p^D \right| \tag{5.8}$$

for p close to p^D. In the following discussion, without loss of generality, we assume $p^D \in (\underline{p}, \overline{p})$. Note that this can always be achieved by choosing a large interval of $[\underline{p}, \overline{p}]$.

We now state the main result in this section. We use the notation $f(n) = O^*(g(n))$ to denote there is a constant C and k such that $f(n) \le C \cdot g(n) \cdot \log^k n$.

Theorem 1 *Let Assumption 1 hold for $\Gamma = \Gamma(M, K, m_L, m_U)$. Then, there exists an admissible policy π generated by Algorithm 1, such that for all $n \ge 1$,*

$$\sup_{\lambda \in \Gamma} R_n^{\pi}(x, T; \lambda) = O^* \left(n^{-1/2} \right).$$

A corollary of Theorem 1 follows from the relationship between the nonparametric model and the parametric one:

Corollary 1 *Assume that Γ is a parameterized demand function family satisfying Assumption 1. Then, there exists an admissible policy π generated by Algorithm 1, such that for all $n \ge 1$,*

$$\sup_{\lambda \in \Gamma} R_n^{\pi}(x, T; \lambda) = O^* \left(n^{-1/2}\right).$$

Now, we explain the meaning of Theorem 1 and Corollary 1. First, as we will show a matching lower bound in Theorem 2, the result in Theorem 1 is the best asymptotic regret that one can achieve in this setting. Another consequence of our result is that it shows that there is no performance gap between parametric and nonparametric settings in the asymptotic sense, implying that the value of knowing the parametric form of the demand function is marginal in this problem when the best algorithm is adopted. In this sense, our algorithm could save firms' efforts in searching for the right parametric form of the demand functions.

Now, we describe the dynamic pricing algorithm. As mentioned earlier, we aim to learn p^D through price experimentations. Specifically, the algorithm will be able to distinguish whether p^u or p^c is optimal. Meanwhile, it keeps a shrinking interval containing the optimal price with high probability until a certain accuracy is achieved.

Now, we explain the ideas behind the Algorithm 1. In the algorithm, we divide the selling season into a carefully selected set of time periods. In each time period, we test a set of prices within a certain price interval. Based on the empirical observations, we shrink the price interval to a smaller subinterval that still contains the optimal price with high probability and enter the next time period with a smaller price range. We repeat the shrinking procedure until the price interval is small enough so that the desired accuracy is achieved.

Recall that the optimal deterministic price p^D is equal to the maximum of p^u and p^c, where p^u and p^c are solved from (5.6). As shown in (5.7) and (5.8), the local behavior of the revenue rate function is quite different around p^u and p^c: the former one resembles a quadratic function, while the latter one resembles a linear function (this is an important feature due to the inventory constraint). This difference requires us to use different shrinking strategies for the cases when $p^u > p^c$ and $p^c > p^u$. This is why we have two learning steps (Steps 2 and 3) in our algorithm. Specifically, in Step 2, the algorithm works by shrinking the price interval until either a transition condition (5.10) is triggered or the learning phase is terminated. We show that when the transition condition (5.10) is triggered, with high probability, the optimal solution to the deterministic problem is p^c. Otherwise, if we terminate learning before the condition is triggered, we know that p^u is either the optimal solution to the deterministic problem or it is close enough so that using p^u can yield a near-optimal revenue. When (5.10) is triggered, we switch to Step 3, in which we use a new set of shrinking and price testing parameters. Note that in Step 3, we start from the initial price interval rather than the current interval obtained. This is not necessary but solely for the ease of analysis. Both Step 2 and Step 3 (if it is invoked) must terminate in a finite number of iterations.

In the end of the algorithm, a fixed price is used for the remaining selling season (Step 4) until the inventory runs out. In fact, instead of applying a fixed price in

Algorithm 1 Dynamic pricing algorithm (DPA)

```
Step 1. Initialization:
  (a) Consider a sequence of $\tau_i^u, \kappa_i^u$, $i = 1, 2, \ldots, N^u$ and $\tau_i^c, \kappa_i^c$, $i = 1, 2, \ldots, N^c$ ($\tau$ and $\kappa$ represent the length of each learning period and the number of different prices to be tested in each learning period, respectively. Their values along with the values of $N^u$ and $N^c$ are defined in (5.22)–(5.27), (5.17) and (5.21)). Define $\underline{p}_1^u = \underline{p}_1^c = \underline{p}$ and $\overline{p}_1^u = \overline{p}_1^c = \overline{p}$. Define $t_i^u = \sum_{j=1}^i \tau_j^u$, for $i = 0$ to $N^u$ and $t_i^c = \sum_{j=1}^i \tau_j^c$, for $i = 0$ to $N^c$;
Step 2. Learn $p^u$ or determine $p^c > p^u$:
For $i = 1$ to $N^u$ do
  (a) Divide $[\underline{p}_i^u, \overline{p}_i^u]$ into $\kappa_i^u$ equally spaced intervals and let $\{p_{i,j}^u, j = 1, 2, \ldots, \kappa_i^u\}$ be the left endpoints of these intervals;
  (b) Divide the time interval $[t_{i-1}^u, t_i^u]$ into $\kappa_i^u$ equal parts and define
        $$\Delta_i^u = \frac{\tau_i^u}{\kappa_i^u}, \qquad t_{i,j}^u = t_{i-1}^u + j\Delta_i^u, \qquad j = 0, 1, \ldots, \kappa_i^u;$$
  (c) For $j$ from 1 to $\kappa_i^u$, apply $p_{i,j}^u$ from time $t_{i,j-1}^u$ to $t_{i,j}^u$. If inventory runs out, then apply $p_\infty$ until time $T$ and STOP;
  (d) Compute
        $$\hat{d}(p_{i,j}^u) = \frac{\text{total demand over } [t_{i,j-1}^u, t_{i,j}^u)}{\Delta_i^u}, \quad j = 1, \ldots, \kappa_i^u;$$
  (e) Compute
        $$\hat{p}_i^u = \arg\max_{1 \le j \le \kappa_i^u} \{p_{i,j}^u \hat{d}(p_{i,j}^u)\} \quad \text{and} \quad \hat{p}_i^c = \arg\min_{1 \le j \le \kappa_i^u} \left|\hat{d}(p_{i,j}^u) - x/T\right|; \tag{5.9}$$
  (f) If
        $$\hat{p}_i^c > \hat{p}_i^u + 2\sqrt{\log n} \cdot \frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u} \tag{5.10}$$
      then break from Step 2, enter Step 3 and set $i_0 = i$;
      Otherwise, set $\hat{p}_i = \max\{\hat{p}_i^c, \hat{p}_i^u\}$. Define
        $$\underline{p}_{i+1}^u = \hat{p}_i - \frac{\log n}{3} \cdot \frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u} \text{ and } \overline{p}_{i+1}^u = \hat{p}_i + \frac{2\log n}{3} \cdot \frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u}. \tag{5.11}$$
      And define the price range for the next iteration
        $$I_{i+1}^u = [\underline{p}_{i+1}^u, \overline{p}_{i+1}^u].$$
      Here we truncate the interval if it does not lie inside the feasible set $[\underline{p}, \overline{p}]$;
  (g) If $i = N^u$, then enter Step 4(a);
```

(continued)

Algorithm 1 (continued)

```
Step 3. Learn $p^c$ when $p^c > p^u$:
For $i = 1$ to $N^c$ do
    (a) Divide $[\underline{p}_i^c, \overline{p}_i^c]$ into $\kappa_i^c$ equally spaced intervals and let $\{p_{i,j}^c, j = 1, 2, \ldots, \kappa_i^c\}$ be the left endpoints of these intervals;
    (b) Define
        $$\Delta_i^c = \frac{\tau_i^c}{\kappa_i^c}, \qquad t_{i,j}^c = t_{i-1}^c + j\Delta_i^c + t_{i_0}^u, \qquad j = 0, 1, \ldots, \kappa_i^c;$$
    (c) For $j$ from 1 to $\kappa_i^c$, apply $p_{i,j}^c$ from time $t_{i,j-1}^c$ to $t_{i,j}^c$. If inventory runs out, then apply $p_\infty$ until time $T$ and STOP;
    (d) Compute
        $$\hat{d}(p_{i,j}^c) = \frac{\text{total demand over } [t_{i,j-1}^c, t_{i,j}^c)}{\Delta_i^c}, \quad j = 1, \ldots, \kappa_i^c;$$
    (e) Compute
        $$\hat{q}_i = \arg\min_{1 \le j \le \kappa_i^c} \left| \hat{d}(p_{i,j}^c) - x/T \right|. \qquad (5.12)$$
        Define
        $$\underline{p}_{i+1}^c = \hat{q}_i - \frac{\log n}{2} \cdot \frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c} \text{ and } \overline{p}_{i+1}^c = \hat{q}_i + \frac{\log n}{2} \cdot \frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c}. \qquad (5.13)$$
        And define the price range for the next iteration
        $$I_{i+1}^c = [\underline{p}_{i+1}^c, \overline{p}_{i+1}^c].$$
        Here, we truncate the interval if it does not lie inside the feasible set of $[\underline{p}, \overline{p}]$;
    (f) If $i = N^c$, then enter Step 4(b);
Step 4. Apply the learned price:
    (a) Define $\tilde{p} = \hat{p}_{N^u} + 2\sqrt{\log n} \cdot \frac{\overline{p}_{N^u}^u - \underline{p}_{N^u}^u}{\kappa_{N^u}^u}$. Use $\tilde{p}$ for the rest of the selling season until the inventory runs out;
    (b) Define $\tilde{q} = \hat{q}_{N^c}$. Use $\tilde{q}$ for the rest of the selling season until the inventory runs out.
```

Step 4, one may continue learning using our shrinking strategy. However, it will not further improve the asymptotic performance of our algorithm.

In the following, we define τ_i^u, κ_i^u, N^u, τ_i^c, κ_i^c and N^c. Without loss of generality, we assume $T = 1$ and $\overline{p} - \underline{p} = 1$ in the following discussion. We first provide a set of relations we want $(\tau_i^u, \kappa_i^{\overline{u}})$ and (τ_i^c, κ_i^c) to satisfy. Then, we explain the meaning of each relations and derive a set of parameters that satisfy these relations. We use the notation $f \sim g$ to mean that f and g are of the same order in n.

The relations that we want $(\tau_i^u, \kappa_i^u)_{i=1}^{N^u}$ to satisfy are

$$\left(\frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u}\right)^2 \sim \sqrt{\frac{\kappa_i^u}{n\tau_i^u}}, \quad \forall i = 2, \ldots, N^u, \tag{5.14}$$

$$\overline{p}_{i+1}^u - \underline{p}_{i+1}^u \sim \log n \cdot \frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u}, \quad \forall i = 1, \ldots, N^u - 1, \tag{5.15}$$

$$\tau_{i+1}^u \cdot \left(\frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u}\right)^2 \cdot \sqrt{\log n} \sim \tau_1^u, \quad \forall i = 1, \ldots, N^u - 1. \tag{5.16}$$

Also, we define

$$N^u = \min_l \left\{ l : \left(\frac{\overline{p}_l^u - \underline{p}_l^u}{\kappa_l^u}\right)^2 \sqrt{\log n} < \tau_1^u \right\}. \tag{5.17}$$

Next, we state the set of relations we want $(\tau_i^c, \kappa_i^c)_{i=1}^{N^c}$ to satisfy

$$\frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c} \sim \sqrt{\frac{\kappa_i^c}{n\tau_i^c}}, \quad \forall i = 2, \ldots, N^c, \tag{5.18}$$

$$\overline{p}_{i+1}^c - \underline{p}_{i+1}^c \sim \log n \cdot \frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c}, \quad \forall i = 1, \ldots, N^c - 1, \tag{5.19}$$

$$\tau_{i+1}^c \cdot \frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c} \cdot \sqrt{\log n} \sim \tau_1^c, \quad \forall i = 1, \ldots, N^c - 1. \tag{5.20}$$

Also, we define

$$N^c = \min_l \left\{ l : \frac{\overline{p}_l^c - \underline{p}_l^c}{\kappa_l^c} \sqrt{\log n} < \tau_1^c \right\}. \tag{5.21}$$

To understand the above relations, it is useful to examine the source of revenue losses in our algorithm. First, there is an *exploration loss* in each period—the prices tested are not optimal, resulting in suboptimal revenue rate or suboptimal inventory consumption rate. The magnitude of such losses in each period is roughly the deviation of the revenue rate (or the inventory consumption rate) multiplied by the time length of the period. Second, there is a *deterministic loss* due to the limited learning capacity—we only test a grid of prices in each period and may never use the exact optimal price. Third, since the demand follows a stochastic process, the observed demand rate may deviate from the true underlying demand rate, resulting in a *stochastic loss*. Note that these three losses also exist in the learning algorithm proposed in Besbes and Zeevi (2009). However, in dynamic learning, the loss in one period does not simply appear once, it may have impact on all the future periods. The design of our algorithm tries to balance these losses in each step to achieve the maximum efficiency of learning. With these in mind, we explain the meaning of each equation above in the following:

- The first relation (5.14) ((5.18), respectively) balances the deterministic loss induced by only considering the grid points (the grid granularity is $\frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u}$ ($\frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c}$, resp.)) and the stochastic loss induced in the learning period which will be shown to be $\sqrt{\frac{\kappa_i^u}{n\tau_i^u}}$ ($\sqrt{\frac{\kappa_i^c}{n\tau_i^c}}$, respectively). Due to the relation in (5.7) and (5.8), the loss is quadratic in the price granularity in Step 2 and linear in Step 3.
- The second relation (5.15) ((5.19), respectively) makes sure that with high probability, the price intervals I_i^u (I_i^c, respectively) contain the optimal price p^D. This is necessary, since otherwise a constant loss will be incurred in all periods afterward.
- The third relation (5.16) ((5.20), respectively) bounds the exploration loss for each learning period. This is done by considering the multiplication of the revenue rate deviation (also demand rate deviation) and the length of the learning period, which in our case can be upper bounded by $\tau_{i+1}^u \sqrt{\log n} \cdot \left(\frac{\overline{p}_i^u - \underline{p}_i^u}{\kappa_i^u}\right)^2$ ($\tau_{i+1}^c \sqrt{\log n} \cdot \frac{\overline{p}_i^c - \underline{p}_i^c}{\kappa_i^c}$, respectively). We want this loss to be of the same order for each learning period (and all equal to the loss in the first learning period, which is τ_1) to achieve the maximum efficiency of learning.
- Formula (5.17) ((5.21), respectively) determines when the price we obtain is close enough to optimal such that we can apply this price in the remaining selling season. We show that $\sqrt{\log n} \cdot \left(\frac{\overline{p}_l^u - \underline{p}_l^u}{\kappa_l^u}\right)^2$ ($\sqrt{\log n} \cdot \frac{\overline{p}_l^c - \underline{p}_l^c}{\kappa_l^c}$, respectively) is an upper bound of the revenue rate and demand rate deviations of price $\hat{p}_l$. When this is less than τ_1, we can simply apply $\hat{p}_l$ and the loss will not exceed the loss of the first learning period.

Now, we solve the relations (5.14)–(5.16) and obtain a set of parameters that satisfy them:

$$\tau_1^u = n^{-\frac{1}{2}} \cdot (\log n)^{3.5} \text{ and } \tau_i^u = n^{-\frac{1}{2} \cdot (\frac{3}{5})^{i-1}} \cdot (\log n)^5, \ \forall i = 2, \dots, N^u, \tag{5.22}$$

$$\kappa_i^u = n^{\frac{1}{10} \cdot (\frac{3}{5})^{i-1}} \cdot \log n, \quad \forall i = 1, 2, \dots, N^u. \tag{5.23}$$

As a by-product, we have

$$\overline{p}_i^u - \underline{p}_i^u = n^{-\frac{1}{4}(1-(\frac{3}{5})^{i-1})}, \quad \forall i = 1, 2, \dots, N^u. \tag{5.24}$$

Similarly, we solve the relations (5.18)–(5.20) and obtain a set of parameters that satisfy them:

$$\tau_1^c = n^{-\frac{1}{2}} \cdot (\log n)^{2.5} \text{ and } \tau_i^c = n^{-\frac{1}{2}\cdot(\frac{2}{3})^{i-1}} \cdot (\log n)^3,\ \forall i = 2, \dots, N^c, \quad (5.25)$$

$$\kappa_i^c = n^{\frac{1}{6}\cdot(\frac{2}{3})^{i-1}} \cdot \log n, \quad \forall i = 1, 2, \dots, N^c \quad (5.26)$$

and

$$\overline{p}_i^c - \underline{p}_i^c = n^{-\frac{1}{2}(1-(\frac{2}{3})^{i-1})}, \quad \forall i = 1, \dots, N^c. \quad (5.27)$$

Note that by (5.24) and (5.27), the price intervals defined in our algorithm indeed shrink in each iteration.

5.2.2 Lower Bound Example

In the last section, we proposed a dynamic pricing algorithm and proved an upper bound of $O^*(n^{-1/2})$ on its regret in Theorem 1. In this section, we show that there exists a class of demand functions satisfying our assumptions such that no pricing policy can achieve an asymptotic regret less than $O^*(n^{-1/2})$. This lower bound example provides a clear evidence that the upper bound is tight. Therefore, our algorithm achieves nearly the best performance among all possible algorithms and closes the performance gap for this problem. Because our algorithm and the lower bound example apply for both parametric and nonparametric settings, it also closes the gap for the problem with a known parametric demand function.

Theorem 2 (Lower Bound Example) *Let $\lambda(p; z) = 1/2 + z - zp$, where z is a parameter taking values in $Z = [1/3, 2/3]$ (we denote this demand function set by Λ). Let $\underline{p} = 1/2$, $\overline{p} = 3/2$, $x = 2$, and $T = 1$. Then, we have the following:*

- *This class of demand function satisfies Assumption 1. Furthermore, for any $z \in [1/3, 2/3]$, the optimal price p^D always equals p^u and $p^D \in [7/8, 5/4]$.*
- *For any admissible pricing policy π and all $n \geq 1$,*

$$\sup_{z \in Z} R_n^\pi(x, T; z) \geq \frac{1}{12(48)^2\sqrt{n}}.$$

We first explain some intuitions behind this example. Note that all the demand functions in Λ cross at one common point, that is, when $p = 1$, $\lambda(p; z) = 1/2$. Such a price is called an *uninformative* price in Broder and Rusmevichientong (2012). When there exists an uninformative price, experimenting at that price will not gain information about the demand function. Therefore, in order to learn the demand function (i.e., the parameter z) and determine the optimal price, one must at least perform some price experiments at prices away from the uninformative price; on the other hand, when the optimal price is indeed the uninformative price, doing price experiments away from the optimal price will incur some revenue losses. This tension is the key reason for such a lower bound for the loss, and mathematically it is reflected in statistical bounds on hypothesis testing. For the proof of Theorem 2, we refer the readers to Wang et al. (2014).

5.3 Multiproduct Setting

In this section, we consider a multiple product and multiple resource generalization of the problem introduced in the previous section. This more general problem, also known as the price-based Network Revenue Management (NRM) problem with learning, considers a setting in which a seller sells to incoming customers n types of products, each of which is made up from a subset of m types of resources, during a finite selling season which consists of T *decision* periods. Denote by $A = [A_{ij}] \in \mathbb{R}_+^{m\times n}$ the *resource consumption matrix*, which indicates that a single unit of product j requires A_{ij} units of resource i. Denote by $C \in \mathbb{R}_+^m$ the vector of initial capacity levels of all resources at the beginning of the selling season which cannot be replenished and have zero salvage value at the end of the selling season. At the beginning of period $t \in [T]$, the seller first decides the price $p_t = (p_{t,1}; \ldots; p_{t,n})$ for his products, where p_t is chosen from a convex and compact set $\mathcal{P} = \otimes_{l=1}^n [\underline{p}_l, \bar{p}_l] \subseteq \mathbb{R}^n$ of feasible price vectors. Let $D_t(p_t) = (D_{t,1}(p_t); \ldots; D_{t,n}(p_t)) \in \mathcal{D} := \{(d_1; \ldots; d_n) \in \{0,1\}^n : \sum_{i=1}^n d_i \leq 1\}$ denote the vector of realized demand in period t under price p_t. For simplicity, we assume that at most one sale for one of the products occurs in each period. We assume that the purchase probability vector for all products under any price p_t, i.e., $\lambda^*(p_t) := \mathbf{E}[D_t(p_t)]$ is unknown to the seller, and this relationship $\lambda^*(.)$, also known as the demand function, needs to be estimated from the data the seller observes during the finite selling season. Define the revenue function $r^*(p) := p \cdot \lambda^*(p)$ to be the one-period expected revenue that the seller can earn under price p. It is typically assumed in the literature that $\lambda^*(.)$ is invertible (see the regularity assumptions below). By abuse of notation, we can then write $r^*(p) = p \cdot \lambda^*(p) = \lambda \cdot p^*(\lambda) = r^*(\lambda)$ to emphasize the dependency of revenue on demand rate instead of on price. We make the following regularity assumptions about $\lambda^*(.)$ and $r^*(.)$ which can be viewed as multidimensional counterparts of Assumption 1.

Regularity Assumptions

R1. $\lambda^*(.)$ is twice continuously differentiable and it has an inverse function $p^*(.)$ which is also twice continuously differentiable.

R2. There exists a set of turnoff prices $p_j^\infty \in \mathbb{R}_+ \cap \{\infty\}$ for $j = 1, \dots, n$ such that for any $p = (p_1; \dots; p_n)$, $p_j = p_j^\infty$ implies that $\lambda_j^*(p) = 0$.
R3. $r^*(.)$ is bounded and strongly concave in λ.

Compared to the single product setting, the NRM setting imposes two challenges: first, the nice solution structure for single product setting breaks down in the presence of multiple types of products and resources, and second, the approach of estimating the deterministic optimal prices and then applying this learned price may not be sufficient to get tight regret bound since ensuring the same estimation error of the deterministic optimal prices in multidimensional setting requires significantly more observations which in turn affects the best achievable regret bound of this approach. The goal of this section is twofold. First, we introduce two settings of NRM where the demand function possesses some additional structural properties, i.e., the parametric setting where demand function comes from a family of functions parameterized by a *finite* number of parameters and the nonparametric setting where demand function is sufficiently smooth. Second, we introduce an adaptive exploitation pricing scheme which help achieve tight regret bound for the two settings. In the remainder of this section, after introducing some additional preliminary results in Sect. 5.3.1, we will first investigate parametric setting in Sect. 5.3.2 and then investigate the nonparametric setting in Sect. 5.3.3.

5.3.1 Preliminaries

Let $D_{1:t} := (D_1, D_2, \dots, D_t)$ denote the history of the demand realized up to and including period t. Let $\mathcal{H}_t$ denote the σ-field generated by $D_{1:t}$. We define a *control* π as a sequence of functions $\pi = (\pi_1, \pi_2, \dots, \pi_T)$, where π_t is a $\mathcal{H}_{t-1}$-measurable real function that maps the history $D_{1:t-1}$ to $\otimes_{j=1}^n [\underline{p}_j, \bar{p}_j] \cup \{p_j^\infty\}$. This class of controls is often referred to as *non-anticipating controls* because the decision in each period depends only on the accumulated observations up to the beginning of the period. Under policy π, the seller sets the price in period t equal to $p_t^\pi = \pi_t(D_{1:t-1})$ almost surely (a.s.). Let Π denote the set of all *admissible controls*:

$$\Pi := \left\{ \pi : \sum_{t=1}^{T} A D_t(p_t^\pi) \preceq C \quad \text{and} \quad p_t^\pi = \pi_t(\mathcal{H}_{t-1}) \;\; a.s. \right\}.$$

Note that even though the seller does not know the underlying demand function, the existence of the turnoff prices $p_1^\infty, \dots, p_n^\infty$ guarantees that this constraint can be satisfied if the seller applies p_j^∞ for product j as soon as the remaining capacity at hand is not sufficient to produce one more unit of product j. Let $\mathbf{P}_t^\pi$ denote the induced probability measure of $D_{1:t} = d_{1:t}$ under an admissible control $\pi \in \Pi$, i.e.,

$$\mathbf{P}_t^{\pi}(d_{1:t}) = \prod_{s=1}^{t} \left[\left(1 - \sum_{j=1}^{n} \lambda_j^*(p_s^{\pi}) \right)^{\left(1-\sum_{j=1}^{n} d_{s,j}\right)} \prod_{j=1}^{n} \lambda_j^*(p_s^{\pi})^{d_{s,j}} \right],$$

where $p_s^{\pi} = \pi_s(d_{1:s-1})$ and $d_s = [d_{s,j}] \in \mathcal{D}$ for all $s = 1, \ldots, t$. (By definition of $\lambda^*(.)$, the term $1 - \sum_{j=1}^{n} \lambda_j^*(p_s^{\pi})$ can be interpreted as the probability of no-purchase in period s under price p_s^{π}.) Denote by $\mathbf{E}^{\pi}$ the expectation with respect to the probability measure $\mathbf{P}_t^{\pi}$. The total expected revenue under $\pi \in \Pi$ is then given by

$$R^{\pi} = \mathbf{E}^{\pi} \left[\sum_{t=1}^{T} p_t^{\pi} \cdot D_t(p_t^{\pi}) \right].$$

The multidimensional version of the deterministic problem in the previous section can be formulated as follows:

$$\text{(P)} \qquad J^D := \max_{p_t, t \in [T]} \left\{ \sum_{t=1}^{T} r^*(p_t) : \sum_{t=1}^{T} A\lambda^*(p_t) \preceq C \right\},$$

$$\text{or equivalently,} \quad (\text{P}_{\lambda}) \qquad J^D := \max_{\lambda_t, t \in [T]} \left\{ \sum_{t=1}^{T} r^*(\lambda_t) : \sum_{t=1}^{T} A\lambda_t \preceq C \right\}.$$

By assumption R3, P_{λ} is a convex program and it can be shown that J^D is an upper bound for the total expected revenue under any admissible control, i.e., $R^{\pi} \leq J^D$ for all $\pi \in \Pi$. This allows us to define the regret of an admissible control $\pi \in \Pi$ as $\rho^{\pi} := J^D - R^{\pi}$. Let λ^D denote the optimal solution of P_{λ}, and let $p^D = p^*(\lambda^D)$ denote the corresponding optimal deterministic price. (Since $r^*(\lambda)$ is strongly concave with respect to λ, by Jensen's inequality, the optimal solution is static, i.e., $\lambda_t = \lambda^D$ for all t.) Let $\mathsf{Ball}(x, r)$ be a closed Euclidean ball centered at x with radius r. We state our fourth regularity assumption below which essentially states that the static price should neither be too low that it attracts too much demand nor too high that it induces no demand:

R4. There exists $\phi > 0$ such that $\mathsf{Ball}(p^D, \phi) \subseteq \mathcal{P}$.

Finally, we will consider a sequence of problems where the length of the selling season and the initial capacity levels are scaled proportionally by a factor $k > 0$. One can interpret k as the *size* of the problem. One can show that the optimal deterministic solution in the scaled problems remains λ^D. Let $\rho^{\pi}(k)$ denote the regret under an admissible control $\pi \in \Pi$ for the problem with scaling factor k. We use the asymptotic order of $\rho^{\pi}(k)$ as the metric for heuristic performance.

5.3.2 Parametric Case

In the parametric setting, the functional form of the demand is known, but the finite parameters which pin down the function are unknown. Mathematically, let Θ be a compact subset of $\mathbb{R}^q$, where $q \in \mathbb{Z}_{++}$ is the number of unknown parameters. Under the parametric demand case, the seller knows that the underlying demand function $\lambda^*(.)$ equals $\lambda(.;\theta)$ for some $\theta \in \Theta$. Although the function $\lambda(.;\theta)$ is known, the true parameter vector θ^* is unknown and needs to be estimated from the data. The one-period expected revenue function is given by $r(p;\theta) := p \cdot \lambda(p;\theta)$. To leverage the parametric structure of the unknown function, we will focus primarily on *Maximum Likelihood* (ML) estimation which not only has certain desirable theoretical properties but is also widely used in practice. As shown in the statistics literature, to guarantee the regular behavior of ML estimator, certain statistical conditions need to be satisfied. To formalize these conditions in our context, it is convenient to first consider the distribution of a sequence of demand realizations when a sequence of $\tilde{q} \in \mathbb{Z}_{++}$ *fixed* price vectors $\tilde{p} = (\tilde{p}^{(1)}, \tilde{p}^{(2)}, \ldots, \tilde{p}^{(\tilde{q})}) \in \mathcal{P}^{\tilde{q}}$ have been applied. For all $d_{1:\tilde{q}} \in \mathcal{D}^{\tilde{q}}$, we define

$$\mathbf{P}^{\tilde{p},\theta}(d_{1:\tilde{q}}) := \prod_{s=1}^{\tilde{q}} \left[\left(1 - \sum_{j=1}^{n} \lambda_j(\tilde{p}^{(s)};\theta)\right)^{\left(1-\sum_{j=1}^{n} d_{s,j}\right)} \prod_{j=1}^{n} \lambda_j(\tilde{p}^{(s)};\theta)^{d_{s,j}} \right]$$

and denote by $\mathbf{E}_\theta^{\tilde{p}}$ the expectation with respect to $\mathbf{P}^{\tilde{p},\theta}$. In addition to the regularity assumptions R1–R4, we impose additional properties to ensure that the function class $\{\lambda(.;\theta)\}_{\theta\in\Theta}$ is well-behaved.

Parametric Family Assumptions

A1 $\lambda(p;\theta)$ and $\frac{\partial \lambda_j}{\partial p_i}(p;\theta)$ for all $i, j \in [n]$ and $i \neq j$ are continuously differentiable in θ.

A2 R1 and R3 hold for all $\theta \in \Theta$.

A3 There exists $\tilde{p} = (\tilde{p}^{(1)}, \tilde{p}^{(2)}, \ldots, \tilde{p}^{(\tilde{q})}) \in \mathcal{P}^{\tilde{q}}$ such as for all $\theta \in \Theta$,

i. $\mathbf{P}^{\tilde{p},\theta}(.) \neq \mathbf{P}^{\tilde{p},\theta'}(.)$ for all $\theta' \in \Theta$ and $\theta' \neq \theta$.

ii. For all $k \in [\tilde{q}]$ and $j \in [n]$, $\lambda_j(\tilde{p}^{(k)};\theta) > 0$ and $\sum_{j=1}^{n} \lambda_j(\tilde{p}^{(k)};\theta) < 1$.

iii. The minimum eigenvalue of the matrix $\mathcal{I}(\tilde{p},\theta) := [\mathcal{I}_{i,j}(\tilde{p},\theta)] \in \mathbb{R}^{q\times q}$ where

$$\mathcal{I}_{i,j}(\tilde{p},\theta) = \mathbf{E}_\theta^{\tilde{p}}\left[-\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log \mathbf{P}^{\tilde{p},\theta}(D_{1:\tilde{q}})\right]$$

is bounded from below by a positive number.

Note that A1 and A2 are quite natural assumptions satisfied by many demand functions such as linear, multinomial logit, and exponential demand. We call $\tilde{p}$ in A3 *exploration prices*. A3 ensures that there exists a set of price vectors (e.g., $\tilde{p}$), which, when used repeatedly, would allow the seller to use ML estimator to statistically identify the true demand parameter. Note that the symmetric matrix $\mathcal{I}(\tilde{p}, \theta)$ defined in A3-iii is known as the *Fisher information* matrix in the literature, and it captures the amount of information that the seller obtains about the true parameter vector using the exploration prices $\tilde{p}$. A3-iii requires the Fisher matrix to be strongly positive definite; this is needed to guarantee that the seller's information about the underlying parameter vector strictly increases as he observes more demand realizations under $\tilde{p}$. We want to point out that it is easy to find exploration prices for the commonly used demand function families. For example, for linear and exponential demand function families, any $\tilde{q} = n + 1$ price vectors $\tilde{p}^{(1)}, \ldots, \tilde{p}^{(n+1)}$ constitute a set of exploration prices if (a) they are all in the interior of $\mathcal{P}$ and (b) the vectors $(1; \tilde{p}^{(1)}), \ldots, (1; \tilde{p}^{(n+1)}) \in \mathbb{R}^{n+1}$ are linearly independent. For the multinomial logit demand function family, any $\tilde{q} = 2$ price vectors $\tilde{p}^{(1)}, \tilde{p}^{(2)}$ constitute a set of exploration prices if (a) they are both in the interior of $\mathcal{P}$ and (b) $\tilde{p}_i^{(1)} \neq \tilde{p}_i^{(2)}$ for all $i = 1, \ldots, n$.

Next, we develop a heuristic called Parametric Self-adjusting Control (PSC). In PSC, the selling season is divided into an *exploration* stage followed by an *exploitation* stage. The exploration stage lasts for L periods (L is a tuning parameter to be selected by the seller) where the seller alternates among exploration prices to learn the demand function. At the end of the exploration stage, the seller computes his ML estimate of θ^*, denoted by $\hat{\theta}_L$ (in case the maximum of the likelihood function is not unique, take any maximum as the ML estimate), based on all his observations so far, and solves $\mathrm{P}_\lambda(\hat{\theta}_L)$ for its solution $\lambda^D(\hat{\theta}_L)$ as an estimate of the deterministically optimal demand rate $\lambda^D(\theta^*)$. Then, for the remaining $(T - L)$-period exploitation stage, the seller uses price vectors according to a simple adaptive rule which we explain in more detail below. Define $\hat{\Delta}_t(p_t; \hat{\theta}_L) := D_t - \lambda(p_t; \hat{\theta}_L)$, and let C_t denote the remaining capacity at the *end* of period t. The complete PSC procedure is given in Algorithm 2.

In contrast to many proposed heuristics that use the learned deterministic optimal price for exploitation, PSC uses the adaptive price adjustment rule in (5.28) for exploitation. To see the idea behind this design, suppose the estimate of the parameter vector is accurate (Jasin, 2014). In that setting, $\hat{\Delta}_t$ equals the stochastic variability in demand arrivals $\Delta_t := D_t - \lambda(p_t; \theta^*)$, and the pricing rule in (5.28) reduces to adjusting the prices in each period t to achieve a *target demand rate*, i.e., $\lambda^D(\theta^*) - \sum_{s=L+1}^{t-1} \frac{\Delta_s}{T-s}$. The first part of this expression, $\lambda^D(\theta^*)$, is the optimal demand rate if there were no stochastic variability, and we use it as a *base rate*; the second part of the expression, on the other hand, works as a fine adjustment to the base rate in order to mitigate the observed stochastic variability. To see how such adjustment works, consider the case with a single product: if there is more demand than what the seller expects in period s, i.e., $\Delta_s > 0$, then the pricing rule automatically accounts for it by reducing the target demand rate for all remaining

Algorithm 2 Parametric self-adjusting control (PSC)

```
Tuning Parameter: $L$

1. Stage 1 (Exploration)
    a. Determine the exploration prices $\{\tilde{p}^{(1)}, \tilde{p}^{(2)}, \dots, \tilde{p}^{(\tilde{q})}\}$.
    b. For $t = 1$ to $L$, do:
        • If $C_{t-1} \succeq A_j$ for all $j$, apply price $p_t = \tilde{p}^{(\lfloor (t-1)\tilde{q}/L \rfloor + 1)}$ in period $t$.
        • Otherwise, apply price $p_{t',j} = p_j^{\infty}$ for all $j$ and $t' \geq t$; then terminate PSC.
    c. At the end of period $L$:
        • Compute the ML estimate $\hat{\theta}_L$ based on $p_{1:L}$ and $D_{1:L}$
        • Solve $\mathrm{P}_{\lambda}(\hat{\theta}_L)$ for $\lambda^D(\hat{\theta}_L)$.

2. Stage 2 (Exploitation)
   For $t = L + 1$ to $T$, compute:

   $$\hat{p}_t = p\left(\lambda^D(\hat{\theta}_L) - \sum_{s=L+1}^{t-1} \frac{\hat{\Delta}_s(p_s; \hat{\theta}_L)}{T - s}\, ; \hat{\theta}_L\right). \qquad (5.28)$$

    • If $C_{t-1} \succeq A_j$, and $\hat{p}_t \in \mathcal{P}$, apply price $p_t = \hat{p}_t$ in period $t$
    • Otherwise, for product $j = 1$ to $n$, do:
        – If $C_{t-1} \prec A_j$, apply price $p_{t,j} = p_j^{\infty}$.
        – Otherwise, apply price $p_{t,j} = p_{t-1,j}$
```

$(T - s)$-period; moreover, the target demand rate adjustment is made *uniformly* across all $(T - s)$-period so as to minimize unnecessary price variations. Jasin (2014) has shown that the ability to *accurately* mitigate the stochastic variability allows this self-adjusting pricing rule be effective *when the parameter vector is known*. However, as one can imagine, such *precise* adjustment is not possible when the parameter vector is subject to estimation error. Indeed, when $\hat{\theta}_L \neq \theta^*$, the seller can only adjust target demand rate based on an estimate of Δ_s, i.e., $\hat{\Delta}_s$; moreover, the seller can no longer correctly find out the price vector that accurately induces (on average) the target demand rate since the inverse demand function is also subject to estimation error. Can this pricing rule work well when the underlying demand parameter is subject to estimation error? The answer is yes, and the key observation is that these two sources of systematic biases push the price decisions on opposing directions and their impact is thus reduced. To see that, consider a single product case where the seller overestimates demand for all prices, i.e., $\lambda(p; \hat{\theta}_L) > \lambda(p; \theta^*)$ for all p: on the one hand, since the seller would underestimate the stochastic variation that he needs to adjust (i.e., $\hat{\Delta}_s = D_s - \lambda(p_s; \hat{\theta}_L) < D_s - \lambda(p_s; \theta^*) = \Delta_s$), this would push up the target demand rate (which would push down the price) than if there were no estimation error; on the other hand, since $p(\lambda; \hat{\theta}_L) > p(\lambda; \theta^*)$, for a given target demand rate, the presence of estimation error would push the price up. Quite interestingly, these opposing mechanisms are sufficient for PSC to achieve the optimal rate of regret.

Theorem 3 *Suppose that R1–R4 and A1–A3 hold. Set* $L = \lceil \sqrt{kT} \rceil$. *Then, there exists a constant* $M_1 > 0$ *independent of* $k \geq 1$ *such that* $\rho^{PSC}(k) \leq M_1\sqrt{k}$ *for all* $k \geq 1$.

Note that in light of the lower bound example in the previous section, PSC achieves the best achievable regret. The reason PSC achieves this tight bound can be briefly explained as follows. First, it leverages the fact that the demand model is fully determined by a *finite-dimensional* vector θ^*, which can be efficiently estimated by ML estimation. Under ML, roughly speaking, to obtain an estimation error in the order of ϵ, the seller needs to spend roughly $\Theta(\epsilon^{-2})$ periods exploring the demand curve with exploration prices which are not necessarily optimal. Second, the self-adjusting pricing rule in (5.28) helps reduce the impact of estimation error on revenue obtained during exploitation compared to using the learned deterministic price directly. To see that, suppose that the true parameter vector is misestimated by a *small* error ϵ, then one can show that $\lambda^D(\hat{\theta}_L)$ is roughly ϵ away from $\lambda^D(\theta^*)$. If the seller simply uses the learned deterministic optimal price $p^D(\hat{\theta}_L)$ throughout the exploitation stage, then the one-period regret is roughly $r(\lambda^D(\theta^*); \theta^*) - r(\lambda^D(\hat{\theta}_L); \theta^*) \approx \nabla_\lambda r(\lambda^D(\theta^*); \theta^*) \cdot (\lambda^D(\theta^*) - \lambda^D(\hat{\theta}_L)) \approx \Theta(\epsilon)$ (note that a tighter bound cannot be obtained since the gradient at the constrained optimal solution is not necessarily zero). In PSC, as mentioned above, the pricing rule (5.28) introduces opposing mechanisms to mitigate the impact of systematic error ϵ on regret which results in a one-period regret of $\Theta(\epsilon^2)$. Thus, the total regret in both exploration and exploitation is bounded by $\Theta(L) + \Theta(\epsilon^2(kT - L)) = O(\epsilon^{-2} + \epsilon^2 kT)$, which is bounded by $O(\sqrt{kT})$ after optimally tuning ϵ (or equivalently, L).

5.3.3 Nonparametric Case

The setting in Sect. 5.3.2 assumes that the seller has a good prior knowledge of the functional form of the demand function which may not be appropriate in cases such as new product launch where no historically relevant data is available. Blindly assuming a parametric demand model may be inappropriate and could potentially result in significant revenue loss if the parametric form is misspecified, e.g., a seller who uses linear model to fit the data generated by a logit model. An alternative setting, also known as the *nonparametric approach*, is one where the seller has no prior knowledge of the functional form but tries to estimate the demand directly. The challenge of this approach is that, instead of estimating a finite number of parameters, the seller now needs to directly estimate the demand function value at different price vectors to get an idea of the shape of the demand curve; thus, the number of point estimates needed to ensure low estimation error increases exponentially as the number of products increases. To keep the estimation problem tractable, a common assumption made in the statistics literature for nonparametric approaches is to impose smoothness conditions of the underlying demand functions (Gyorfi et al., 2002). To that end, let $\bar{s}$ denote the largest integer such that $|\partial^{a_1,\dots,a_n}\lambda_j^*(p)/\partial p_1^{a_1}\dots\partial p_n^{a_n}|$ is uniformly bounded for all $j \in [n]$ and $0 \le a_1,\dots,a_n \le \bar{s}$. We call $\bar{s}$ the *smoothness index*. We make the following smoothness assumptions:

Nonparametric Function Smoothness Assumptions

N1. $\bar{s} \ge 2$.

N2. $\left|\frac{\partial^{a_1,\dots,a_n}\lambda_j^*(p)}{\partial p_1^{a_1}\dots\partial p_n^{a_n}}\right|$ is uniformly bounded for all $j \in [n]$, $p \in \mathcal{P}$, $0 \le a_1,\dots,a_n \le \bar{s}$.

The above assumptions are fairly mild and are satisfied by most commonly used demand functions, including linear, polynomial with higher degree, logit, and exponential with a bounded domain of feasible prices. The smoothness index $\bar{s}$ indicates the level of difficulty in estimating the corresponding demand function: the larger the value of $\bar{s}$, the smoother the demand function, and the easier it is to estimate its shape because its value cannot have a drastic local change.

The idea of the nonparametric approach to be introduced later in this section is to replace the ML estimator in PSC by a nonparametric estimation procedure. One such approach is to use a linear combination of spline functions to approximate the underlying demand function which we introduce below. Spline functions have been widely used in engineering to approximate complicated functions, and their popularity is primarily due to their flexibility in effectively approximating complex curve shapes (Schumaker, 2007). This flexibility lies in the piecewise nature of spline functions—a spline function is constructed by attaching piecewise polynomial functions with a certain degree, and the coefficients of these polynomials are computed in such a way that a sufficiently high degree of smoothness is ensured in the places where the polynomials are connected. More formally, for all $l \in [n]$, let $\underline{p}_l = x_{l,0} < x_{l,1} \cdots < x_{l,d} < x_{l,d+1} = \bar{p}_l$ be a partition that divides $[\underline{p}_l, \bar{p}_l]$ into $d+1$ subintervals of equal length where $d \in \mathbb{Z}_{++}$. Let $\mathcal{G} := \otimes_{l=1}^n \mathcal{G}_l$ denote a set of grid points, where $\mathcal{G}_l = \{x_{l,i}\}_{i=0}^{d+1}$. We define the function space of *tensor-product polynomial splines of order* $(s;\dots;s) \in \mathbb{R}^n$ with a set of grid points $\mathcal{G}$ as $\mathsf{S}(\mathcal{G}, s) := \otimes_{l=1}^n \mathsf{S}_l(\mathcal{G}_l, s)$, where $\mathsf{S}_l(\mathcal{G}_l, s) := \{f \in \mathsf{C}^{s-2}([\underline{p}_l, \bar{p}_l]) : f$ is a single-variate polynomial of degree $s-1$ on each subinterval $[x_{l,i-1}, x_{l,i})$, for all $i \in [d]$ and $[x_{l,d}, x_{l,d+1}]\}$. One of the key questions that spline approximation theory addresses is the following: given an arbitrary function λ that satisfies N1-N2, find a spline function $g^* \in \mathsf{S}(\mathcal{G}, s)$ that approximates λ well. Among the various approaches, one of the most popular approximations is using the so-called *tensor-product B-Spline basis functions* (Schumaker, 2007). This approach is based on using the linear combinations of a collection of $(s+d)^n$ tensor-product B-Spline basis functions, denoted by $\{N_{i_1,\dots,i_n}(x_1,\dots,x_n)\}_{i_1=1,\dots,i_n=1}^{s+d,\dots,s+d}$, which span the functional space $\mathsf{S}(\mathcal{G}, s)$, to approximate the target function λ. Therefore, the problem of finding g^* is reduced to the problem of computing the coefficients for representing g^*. Schumaker (2007) proposed an explicit formula for computing these coefficients when the value of λ is perfectly observable, and the coefficients depend on $\lambda(.)$ only via its function value evaluated on a finite number of price vectors in $\mathcal{P}$ (i.e., the $(s+d)^n s^n$ price vectors in $\tilde{\mathcal{G}}$ defined in Algorithm 3); the details for the formula are bit technical, but we provide these in Algorithm 3 for completeness. In our problem setting, finding an approximation for $\lambda_j^*(.)$ for all $j \in [n]$ is more challenging since we observe noisy observations of the function value, so we use empirical mean of demand realizations as a surrogate for $\lambda_j^*(p)$ and propose the following *Spline Estimation* algorithm in Algorithm 4 to estimate the demand, which involves observing $\tilde{L}_0 := L_0(s+d)^n s^n$ samples.

Let $\tilde{\lambda}(.)$ denote the spline function computed via Algorithm 4. It can be shown that with high probability, the approximation error of $\tilde{\lambda}(.)$ converges to zero at a slightly slower rate than the ML estimator in the parametric case. While one may be tempted to directly apply the exploitation method in PSC, i.e., the pricing rule in (5.28), the analysis of such approach is quite difficult since, given the nature of B-spline functions and the estimation procedure, $\tilde{\lambda}(.)$ may lose some of the regularity properties that $\lambda^*(.)$ possesses. Thus, we introduce two more functional approximations on $\tilde{\lambda}(.)$ before applying the self-adjusting pricing procedure for exploitation. To that end, we introduce a quadratic program approximation of P

Algorithm 3 Spline approximation

Input function: $\lambda \in C^0(\mathcal{P})$ and λ satisfies N1 and N2
Output function: $g^* \in \mathsf{S}(\mathcal{G}, s)$

1. For $l \in [n]$, $i \in [s+d]$, define $\{y_{l,i}\}_{i=1}^{2s+d}$ as follows

$$y_{l,1} = \cdots = y_{l,s} = x_{l,0},$$
$$y_{l,s+1} = x_{l,1},\ y_{l,s+2} = x_{l,2}, \ldots, y_{l,s+d} = x_{l,d},$$
$$y_{l,s+d+1} = \cdots = y_{l,2s+d} = x_{l,d+1};$$

   moreover, compute the following:

$$\tau_{l,i,j} = y_{l,i} + (y_{l,i+s} - y_{l,i})\tfrac{j-1}{s-1} \quad \text{and} \quad \beta_{l,i,j} = \textstyle\sum_{v=1}^{j} \tfrac{(-1)^{v-1}}{(s-1)!}\phi_{l,i,s}^{(s-v)}(0)\psi_{l,i,j}^{(v-1)}(0), \quad \text{for } j \in [s],$$

   where $\phi_{l,i,s}(t) = \prod_{r=1}^{s-1}(t - y_{l,i+r})$, $\psi_{l,i,j}(t) = \prod_{r=1}^{j-1}(t - \tau_{l,i,r})$, $\psi_{l,i,1}(t) \equiv 1$. Let $\tilde{\mathcal{G}} := \{(\tau_{1,i_1,j_1}; \ldots; \tau_{n,i_n,j_n}) : i_l \in [s+d],\ j_l \in [s] \text{ for all } l \in [n]\}$.
2. Define g^* as follows:

$$g^*(x_1, \ldots, x_n) = \sum_{i_1=1}^{s+d} \cdots \sum_{i_n=1}^{s+d} \gamma_{i_1,\ldots,i_n} N_{i_1,\ldots,i_n}(x_1, \ldots, x_n),$$

$$\text{where} \quad \gamma_{i_1,\ldots,i_n} = \sum_{j_1=1}^{s}\sum_{r_1=1}^{j_1} \cdots \sum_{j_n=1}^{s}\sum_{r_n=1}^{j_n} \frac{\lambda(\tau_{1,i_1,r_1}, \ldots, \tau_{n,i_n,r_n}) \prod_{l=1}^{n} \beta_{l,i_l,j_l}}{\prod_{l=1}^{n} \prod_{s_l=1, s_l \neq r_l}^{j_l} (\tau_{l,i_l,r_l} - \tau_{l,i_l,s_l})}$$

```
Algorithm 4 Spline estimation
Input Parameter: $L_0, n, s$ Tuning Parameter: $d$
1. Estimate $\lambda^*(p)$ at points $p \in \tilde{\mathcal{G}}$. For each $p \in \tilde{\mathcal{G}}$
    a. Apply price $p$ $L_0$ times
    b. Let $\tilde{\lambda}(p)$ be the sample mean of the $L_0$ observations
2. Construct spline approximation
    a. For all $j \in \overline{[1,n]}$ and $i_l \in \overline{[1,s+d]}$, $l \in \overline{[1,n]}$, calculate coefficients $c^j_{i_1,\dots,i_n}$ as:
       $$c^j_{i_1,\dots,i_n} = \sum_{j_1=1}^{s}\sum_{r_1=1}^{j_1}\cdots\sum_{j_n=1}^{s}\sum_{r_n=1}^{j_n} \frac{\tilde{\lambda}_j(\tau_{1,i_1,r_1},\dots,\tau_{n,i_n,r_n})\ \prod_{l=1}^{n}\beta_{l,i_l,j_l}}{\prod_{l=1}^{n}\prod_{s_l=1,s_l\neq r_l}^{j_l}(\tau_{l,i_l,r_l}-\tau_{l,i_l,s_l})}.$$
    b. Construct a tensor-product spline function $\tilde{\lambda}(p) = (\tilde{\lambda}_1(p);\dots;\tilde{\lambda}_n(p))$, where
       $$\tilde{\lambda}_j(p) = \sum_{i_1=1}^{s+d}\cdots\sum_{i_n=1}^{s+d} c^j_{i_1,\dots,i_n} N_{i_1,\dots,i_n}(p).$$
```

in which we approximate the constraints of P with linear functions and its objective with a quadratic function. First, to linearize the constraints of P, since the capacity constraints form an affine transformation of the demand function, we will simply linearize the demand function. For any $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{n\times n}$, let $B_1,\dots,B_n$ be the columns of B and define $\theta_\iota = (a; B_1;\dots;B_n) \in \mathbb{R}^{n^2+n}$, where the subscript ι stands for *linear demand*. We denote a linear demand function by $\lambda(p;\theta_\iota) = a + B'p$. Next, we explain how we use a quadratic function to approximate the objective of P. For any $E \in \mathbb{R}$, $F \in \mathbb{R}^n$, $G \in \mathbb{R}^{n\times n}$, let $G_1,\dots,G_n$ denote the columns of G and define $\theta_o = (E; F; G_1;\dots;G_n) \in \mathbb{R}^{n^2+n+1}$, where the subscript o stands for *objective*. We denote the resulting quadratic function by $q(p;\theta_o) = E + F'p + \frac{1}{2}p'Gp$. Finally, let $\theta = (\theta_o;\theta_\iota) \in \mathbb{R}^{2n^2+2n+1}$. For any $\theta \in \mathbb{R}^{2n^2+2n+1}$, $\delta \in \mathbb{R}^m$, we can define a quadratic program $\mathbf{QP}(\theta;\delta)$ as follows:

$$(\mathbf{QP}(\theta;\delta)) \qquad \max_{p\in\mathcal{P}} \left\{ q(p;\theta_o):\ A\lambda(p;\theta_\iota) \preceq \tfrac{C}{T} - \delta \right\}.$$

It can be shown that quadratic program will have the same optimal solution as P and will possess some very useful stability properties if the parameters of the quadratic and linear functions are chosen as follows: for linear demand function, let $\theta^*_\iota = (a^*; B^*_1;\dots;B^*_n)$, where $B^* := \nabla\lambda^*(p^D)$ and $a^* := \lambda^D - (B^*)'p^D$; for the quadratic objective function, let $\theta^*_o = (E^*; F^*; G^*_1;\dots;G^*_n)$ where

$$E^* := \frac{1}{2}(p^D)'H^*p^D,\quad F^* := a^* - H^*p^D,\quad G^* := B^* + (B^*)' + H^*,$$

where H^* is an n by n symmetric matrix defined as $H^* := B^* \nabla^2 r^*_\lambda(\lambda^D)(B^*)' - B^* - (B^*)'$. Finally, let $\theta^* := (\theta^*_o; \theta^*_\iota)$. Note that $\mathbf{QP}(\theta^*; \mathbf{0})$ is a very intuitive approximation of P since the function $\lambda(p; \theta^*_\iota) = a^* + (B^*)'p = \lambda^D + (B^*)'(p - p^D)$ can be viewed as a linearization of $\lambda^*(.)$ at p^D. Note also that the gradients of the objective function and the constraints in $\mathbf{QP}(\theta^*; \mathbf{0})$ at p^D coincide with those in P. By Karush–Kuhn–Tucker (KKT) optimality conditions, it can be shown that the optimal solution of $\mathbf{QP}(\theta^*; \mathbf{0})$ is the same as the optimal solution of P.

We are now ready to describe *Nonparametric Self-adjusting Control* (NSC) and discuss its asymptotic performance. NSC consists of an exploration procedure and an exploitation procedure. The exploration procedure uses the Spline Estimation algorithm in Algorithm 4 to construct a spline approximation $\tilde{\lambda}(.)$ of the underlying demand function $\lambda^*(.)$. This function $\tilde{\lambda}(.)$ is then used to construct a linear function $\lambda(.; \hat{\theta}_\iota)$ that closely approximates $\lambda(.; \theta^*_\iota)$ in the neighborhood of p^D and a quadratic program that closely approximates P. During the exploitation phase, we use the optimal solution of the approximate quadratic program as baseline control and automatically adjust the price according to a version of (5.28). Further details will be provided below. Recall that $\tilde{L}_0$ is the duration of the Spline Estimation algorithm. Let C_t denote the remaining capacity at the *end* of period t. Let $\hat{\theta} := (\hat{\theta}_o; \hat{\theta}_\iota)$, where $\hat{\theta}_\iota := (\hat{a}; \hat{B}_1; \ldots; \hat{B}_n)$, $\hat{\theta}_o := (\hat{E}; \hat{F}; \hat{G}_1; \ldots; \hat{G}_n)$ and

$$\hat{B} := \nabla\tilde{\lambda}(\tilde{p}^D),\ \hat{a} := \tilde{\lambda} - \hat{B}'\tilde{p}^D,\ \hat{E} := \tfrac{1}{2}(\tilde{p}^D)'\hat{H}\tilde{p}^D,\ \hat{F} := \hat{a} - \hat{H}\tilde{p}^D,$$

$$\hat{G} := \hat{B} + \hat{B}' + \hat{H}, \quad \text{and}$$

$$\hat{H} = [\hat{H}_{ij}] \text{ where } \hat{H}_{ij} := -\hat{u}'_{ij}\hat{B}^{-1}\tilde{\lambda}^D \text{ and } \hat{u}_{ij} := \left[\frac{\partial^2\tilde{\lambda}_1(\tilde{p}^D)}{\partial p_i \partial p_j}; \ldots; \frac{\partial^2\tilde{\lambda}_n(\tilde{p}^D)}{\partial p_i \partial p_j}\right].$$

(Note that $\tilde{p}^D$ is the deterministic optimal solution of a version of P, where λ^* is replaced by $\tilde{\lambda}$.) The details of NSC is given in Algorithm 5.

The following result states that the performance of NSC is close to the best achievable (asymptotic) performance bound.

Theorem 4 *Suppose that* $s \geq 4$, $L_0 = \lceil (kT)^{(s+n/2)/(2s+n-2)} (\log(kT))^{(2s+n-4)/(2s+n-2)} \rceil$ *and* $d = \lceil (L_0^{1/2}/\log(kT))^{1/(s+n/2)} \rceil$. *There exists a constant* $M_1 > 0$ *independent of* $k > 3$ *such that for all* $s \geq 4$, *we have*

$$\rho^{NSC}(k) \leq M_1 k^{\frac{1}{2}+\epsilon(n,s,\bar{s})} \log k, \quad \textit{where } \epsilon(n,s,\bar{s}) = \frac{1}{2}\left(\frac{2s - 2(s \wedge \bar{s}) + n + 2}{2s + n - 2}\right).$$

Note that since most commonly used demand functions such as polynomial with arbitrary degree, logit, and exponential are infinitely differentiable (i.e., $\bar{s}$ can be arbitrarily large), for any fixed $\epsilon > 0$, we can select integers $s \geq (n+2)/(4\epsilon) - (n-2)/2$ such that the performance under NSC is $O(k^{1/2+\epsilon}\log k)$. Theoretically, this means that the asymptotic performance of NSC is very close to the best achievable performance lower bound of $\Omega(\sqrt{k})$. By comparing the algorithm and the analysis

Algorithm 5 Nonparametric self-adjusting control (NSC)

```
Input Parameters: $n$, $s$ Tuning Parameter: $d$, $L_0$

1. Stage 1 (Exploration Phase 1 - Spline Estimation)
   a. For $t = 1$ to $\tilde{L}_0 \wedge T$
      • If $C_{t-1} \prec A_j$ for some $j = 1, \ldots, n$, set $p_{t,j} = p_j^\infty$ for all $j = 1, \ldots, n$.
      • Otherwise, follow Step 1 in Spline Estimation algorithm.
   b. At the end of period $\tilde{L}_0 \wedge T$, do:
      • If $\tilde{L}_0 \geq T$, terminate NSC.
      • If $\tilde{L}_0 < T$ and $C_{\tilde{L}_0} \prec A_j$ for some $j = 1, \ldots, n$:
        – For all $t > \tilde{L}_0$, set $p_{t,j} = p_j^\infty$ for all $j = 1, \ldots, n$.
        – Terminate NSC.
      • If $\tilde{L}_0 < T$ and $C_{\tilde{L}_0} \succeq A_j$ for all $j = 1, \ldots, n$:
        – Follow Step 2 in Spline Estimation algorithm to get $\tilde{\lambda}(.)$.
        – Go to Stage 2 below.

2. Stage 2 (Exploration Phase 2 - Function Approximation)
   a. Solve $\tilde{\mathbf{P}}$ and obtain the optimizer $\tilde{p}^D$.
   b. Let $\delta := C/T - C_{\tilde{L}_0}/(T - \tilde{L}_0)$.
   c. Compute $\hat{a}$, $\hat{B}$, $\hat{E}$, $\hat{F}$, $\hat{G}$, $\hat{H}$ and $\hat{\theta} = (\hat{\theta}_o; \hat{\theta}_\iota)$.
      • If $\hat{B}$ is invertible, go to Stage 2(d) below.
      • Otherwise, for $t = \tilde{L}_0 + 1$ to $T$:
        – If $C_{t-1} \succeq A_j$ for $j = 1, \ldots, n$, apply $p_t = \tilde{p}^D$.
        – Otherwise, for product $j = 1$ to $n$, do:
          · If $C_{t-1} \prec A_j$, set $p_{t,j} = p_j^\infty$.
          · Otherwise, set $p_{t,j} = p_{t-1,j}$.
   d. Solve $\mathbf{QP}(\hat{\theta}; \delta)$ for its static price $p_\delta^D(\hat{\theta})$.

3. Stage 3 (Exploitation)
   For $t = \tilde{L}_0 + 1$ to $T$:
   • Compute: $\hat{p}_t = p_\delta^D(\hat{\theta}) - \nabla_\lambda p(\lambda_\delta^D(\hat{\theta}); \hat{\theta}_\iota) \cdot \sum_{s=\tilde{L}_0+1}^{t-1} \frac{\tilde{\Delta}_s}{T-s}$, where $\tilde{\Delta}_t := D_t - \lambda(p_t; \hat{\theta}_\iota)$.
   • If $\hat{p}_t \in \mathcal{P}$ and $C_{t-1} \succeq A_j$ for $j = 1, \ldots, n$, apply $p_t = \hat{p}_t$.
   • Otherwise, for product $j = 1$ to $n$, do:
     – If $C_{t-1} \prec A_j$, set $p_{t,j} = p_j^\infty$.
     – Otherwise, set $p_{t,j} = p_{t-1,j}$.
```

of PSC and NSC, the extra ϵ in the exponent of the regret bound of NSC is driven by the slightly slower rate of convergence of the nonparametric approach for estimating demand function. It remains an open question whether there exists a nonparametric approach for the NRM setting with a continuum of feasible price vectors which attains a regret bound of $O(\sqrt{k})$.

5.4 Bayesian Learning Setting

The multi-armed bandit (MAB) problem is often used to model the exploration–exploitation trade-off in the dynamic learning and pricing model *without* inventory constraints (see Chap. 1 for an overview of the MAB problem). In one of the earliest papers on the multi-armed bandit problem, Thompson (1933) proposed a novel randomized Bayesian algorithm, which has since been referred to as the *Thompson sampling* algorithm. The basic idea of Thompson sampling is that at each time period, random numbers are sampled according to the posterior distributions of the reward for each action, and then the action with the highest sampled reward is chosen. In a revenue management setting, each "action" or "arm" is a price, and "reward" refers to the revenue earned by offering that price. Thus, in the original Thompson sampling algorithm—in the absence of inventory constraints—random numbers are sampled according to the posterior distributions of the mean demand rates for each price, and the price with the highest sampled revenue (i.e., price times sampled demand) is offered.

In this section, we develop a class of Bayesian learning algorithms for the multiproduct pricing problem with inventory constraints. This class of algorithms extends the powerful machine learning technique known as Thompson sampling to address the challenge of balancing the exploration–exploitation trade-off under the presence of inventory constraints. We focus on a model with discrete price sets and present two algorithms (the algorithm can also be used for continuous price sets, see Ferreira et al. (2018)). The first algorithm adapts Thompson sampling by adding a linear programming (LP) subroutine to incorporate inventory constraints. The second algorithm builds upon our first; specifically, in each period, we modify the LP subroutine to further account for the purchases made to date. Both of the algorithms contain two simple steps in each iteration: sampling from a posterior distribution and solving a linear program. As a result, the algorithms are easy to implement in practice.

5.4.1 Model Setting

We consider a retailer who sells N products, indexed by $i \in [N]$, over a finite selling season. (Below, we denote by $[x]$ the set $\{1, 2, \ldots, x\}$.) These products consume M resources, indexed by $j \in [M]$. Specifically, we assume that one unit of product i consumes a_{ij} units of resource j, where a_{ij} is a fixed constant. The selling season is divided into T periods. There are I_j units of initial inventory for each resource

$j \in [M]$, and there is no replenishment during the selling season. We define $I_j(t)$ as the inventory at the end of period t, and we denote $I_j(0) = I_j$. In each period $t \in [T]$, the following sequence of events occurs:

1. The retailer offers a price for each product from a finite set of admissible price vectors. We denote this set by $\{p_1, p_2, \ldots, p_K\}$, where p_k ($\forall k \in [K]$) is a vector of length N specifying the price of each product. More specifically, we have $p_k = (p_{1k}, \ldots, p_{Nk})$, where p_{ik} is the price of product i, for all $i \in [N]$. Following the tradition in dynamic pricing literature, we also assume that there is a "shut-off" price p_∞ such that the demand for any product under this price is zero with probability one. We denote by $P(t) = (P_1(t), \ldots, P_N(t))$ the prices chosen by the retailer in this period, and require that $P(t) \in \{p_1, p_2, \ldots, p_K, p_\infty\}$.
2. Customers then observe the prices chosen by the retailer and make purchase decisions. We denote by $D(t) = (D_1(t), \ldots, D_N(t))$ the demand of each product at period t. We assume that given $P(t) = p_k$, the demand $D(t)$ is sampled from a probability distribution on $\mathbb{R}_+^N$ with joint cumulative distribution function (CDF) $F(x_1, \ldots, x_N; p_k, \theta)$, indexed by a parameter (or a vector of parameters) θ that takes values in the parameter space $\Theta \subset \mathbb{R}^l$. The distribution is assumed to be subexponential; note that many commonly used demand distributions such as normal, Poisson, exponential and all bounded distributions belong to the family of subexponential distributions. We also assume that $D(t)$ is independent of the history $\mathcal{H}_{t-1} = (P(1), D(1), \ldots, P(t-1), D(t-1))$ given $P(t)$.

   Depending on whether there is sufficient inventory, one of the following events happens:

   (a) If there is enough inventory to satisfy all demand, the retailer receives an amount of revenue equal to $\sum_{i=1}^N D_i(t)P_i(t)$, and the inventory level of each resource $j \in [M]$ diminishes by the amount of each resource used such that $I_j(t) = I_j(t-1) - \sum_{i=1}^N D_i(t)a_{ij}$.

   (b) If there is not enough inventory to satisfy all demand, the demand is partially satisfied and the rest of demand is lost. Let $\tilde{D}_i(t)$ be the demand satisfied for product i. We require $\tilde{D}_i(t)$ to satisfy three conditions: $0 \leq \tilde{D}_i(t) \leq D_i(t), \forall i \in [N]$; the inventory level for each resource at the end of this period is nonnegative: $I_j(t) = I_j(t-1) - \sum_{i=1}^N \tilde{D}_i(t)a_{ij} \geq 0, \forall j \in [M]$; there exists at least one resource $j' \in [M]$ whose inventory level is zero at the end of this period, i.e. $I_{j'}(t) = 0$. Besides these natural conditions, we do not require any additional assumption on how demand is specifically fulfilled. The retailer then receives an amount of revenue equal to $\sum_{i=1}^N \tilde{D}_i(t)P_i(t)$ in this period.

We assume that the demand parameter θ is fixed but *unknown* to the retailer at the beginning of the season, and the retailer must learn the true value of θ from demand data. That is, in each period $t \in [T]$, the price vector $P(t)$ can only be chosen based on the observed history $\mathcal{H}_{t-1}$, but cannot depend on the unknown value θ or any event in the future. The retailer's objective is to maximize expected revenue over the course of the selling season given the prior distribution on θ.

We use a parametric Bayesian approach in our model, where the retailer has a *known* prior distribution of $\theta \in \Theta$ at the beginning of the selling season. However, our model allows the retailer to choose an arbitrary prior. In particular, the retailer can assume an arbitrary parametric form of the demand CDF, given by $F(x_1, \ldots, x_N; p_k, \theta)$. This joint CDF parametrized by θ can parsimoniously model the correlation of demand among products. For example, the retailer may specify products' joint demand distribution based on some discrete choice model, where θ is the unknown parameter in the multinomial logit function. Another benefit of the Bayesian approach is that the retailer may choose a prior distribution over θ such that demand is correlated for different prices, enabling the retailer to learn demand for all prices, not just the offered price. e selling season as inventory is depleted; this latter idea is incorporated into the second algorithm that we will present later.

5.4.2 *Thompson Sampling with Fixed Inventory Constraints*

We now present the first version of the Thompson sampling-based pricing algorithm. For each resource $j \in [M]$, we define a fixed constant $c_j := I_j/T$. Given any demand parameter $\rho \in \Theta$, we define the mean demand under ρ as the expectation associated with CDF $F(x_1, \ldots, x_N; p_k, \rho)$ for each product $i \in [N]$ and price vector $k \in [K]$. We denote by $d = \{d_{ik}\}_{i \in [N], k \in [K]}$ the mean demand under the *true* model parameter θ.

The Thompson sampling with Fixed Inventory Constraints (TS-fixed) algorithm is shown in Algorithm 6. Here, "TS" stands for Thompson sampling, while "fixed" refers to the fact that we use fixed constants c_j for all time periods as opposed to updating c_j over the selling season as inventory is depleted; this latter idea is incorporated into the second algorithm that we will present later.

Steps 1 and 4 are based on the Thompson sampling algorithm for the classical multi-armed bandit setting, whereas Steps 2 and 3 are added to incorporate inventory constraints. In Step 1 of the algorithm, we randomly sample parameter $\theta(t)$ according to the posterior distribution of unknown demand parameter θ. This step is motivated by the original Thompson sampling algorithm for the classical multi-armed bandit problem. The key idea of the Thompson sampling algorithm is to use random sampling from the posterior distribution to balance the exploration–exploitation trade-off. The algorithm differs from the ordinary Thompson sampling in Steps 2 and 3. In Step 2, the retailer solves a linear program, $\mathsf{LP}(d(t))$, which identifies the optimal mixed price strategy that maximizes expected revenue given the sampled parameters. The first constraint specifies that the average resource consumption in this time period cannot exceed c_j, the average inventory available per period. The second constraint specifies that the sum of probabilities of choosing a price vector cannot exceed one. In Step 3, the retailer randomly offers one of the K price vectors (or p_∞) according to probabilities specified by the optimal solution of $\mathsf{LP}(d(t))$. Finally, in Step 4, the algorithm updates the posterior distribution of θ given $\mathcal{H}_t$. Such Bayesian updating is a simple and powerful tool to update belief

Algorithm 6 Thompson sampling with fixed inventory constraints (TS-fixed)

Repeat the following steps for all periods $t = 1, \dots, T$:

1. *Sample Demand*: Sample a random parameter $\theta(t) \in \Theta$ according to the posterior distribution of θ given history $\mathcal{H}_{t-1}$. Let the mean demand under $\theta(t)$ be $d(t) = \{d_{ik}(t)\}_{i\in[N],k\in[K]}$.
2. *Optimize Prices given Sampled Demand*: Solve the following linear program, denoted by $\mathsf{LP}(d(t))$:

$$\begin{aligned}\mathsf{LP}(d(t)):\quad & \max_x \sum_{k=1}^{K}(\sum_{i=1}^{N} p_{ik}d_{ik}(t))x_k \\ \text{subject to } & \sum_{k=1}^{K}(\sum_{i=1}^{N} a_{ij}d_{ik}(t))x_k \le c_j,\ \forall j \in [M] \\ & \sum_{k=1}^{K} x_k \le 1 \\ & x_k \ge 0,\ k \in [K].\end{aligned}$$

   Let $x(t) = (x_1(t), \dots, x_K(t))$ be the optimal solution to $\mathsf{LP}(d(t))$.
3. *Offer Price*: Offer price vector $P(t) = p_k$ with probability $x_k(t)$, and choose $P(t) = p_\infty$ with probability $1 - \sum_{k=1}^{K} x_k(t)$.
4. *Update Estimate of Parameter*: Observe demand $D(t)$. Update the history $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{P(t), D(t)\}$ and the posterior distribution of θ given $\mathcal{H}_t$.

probabilities as more information—customer purchase decisions in our case—becomes available. By employing Bayesian updating in Step 4, we are ensured that as any price vector p_k is offered more and more times, the sampled mean demand associated with p_k for each product i becomes more and more centered around the true mean demand, d_{ik}.

We note that the LP defined in Step 2 is closely related to the LP used by Gallego and Van Ryzin (1997), where they consider a network revenue management problem in the case of known demand. Essentially, their pricing algorithm is a special case of Algorithm 6 where they solve $\mathsf{LP}(d)$, i.e., $\mathsf{LP}(d(t))$ with $d(t) = d$, in every time period.

Next, we illustrate the application of our TS-fixed algorithm by providing one concrete example. For simplicity, in this example, we assume that the prior distribution of demand for different prices is independent; however, the definition of TS-fixed is quite general and allows the prior distribution to be *arbitrarily correlated* for different prices. As mentioned earlier, this enables the retailer to learn the mean demand not only for the offered price but also for prices that are not offered.

Example (Bernoulli Demand with Independent Uniform Prior) We assume that for all prices, the demand for each product is Bernoulli distributed. In this case, the unknown parameter θ is just the mean demand of each product. We use a beta posterior distribution for each θ because it is conjugate to the Bernoulli distribution. We assume that the prior distribution of mean demand d_{ik} is uniform in [0, 1] (which is equivalent to a Beta(1, 1) distribution) and is independent for all $i \in [N]$ and $k \in [K]$. In this example, the posterior distribution is very simple to calculate. Let $N_k(t-1)$ be the number of time periods that the retailer has offered price p_k in the first $t-1$ periods, and let $W_{ik}(t-1)$ be the number of periods that product i is purchased under price p_k during these periods. In Step 1 of TS-fixed, the posterior distribution of d_{ik} is $\text{Beta}(W_{ik}(t-1)+1, N_k(t-1)-W_{ik}(t-1)+1)$, so we sample $d_{ik}(t)$ independently from a $\text{Beta}(W_{ik}(t-1)+1, N_k(t-1)-W_{ik}(t-1)+1)$ distribution for each price k and each product i. In Steps 2 and 3, $\mathsf{LP}(d(t))$ is solved and a price vector $p_{k'}$ is chosen; then, the customer demand $D_i(t)$ is revealed to the retailer. In Step 4, we then update $N_{k'}(t) \leftarrow N_{k'}(t-1)+1$, $W_{ik'}(t) \leftarrow W_{ik'}(t-1)+D_i(t)$ for all $i \in [N]$. The posterior distributions associated with the $K-1$ unchosen price vectors ($k \neq k'$) are not changed.

5.4.3 *Thompson Sampling with Inventory Constraint Updating*

Now, we propose the second Thompson sampling-based algorithm. Recall that in TS-fixed, we use fixed inventory constants c_j in every period. Alternatively, we can update c_j over the selling season as inventory is depleted, thereby incorporating real-time inventory information into the algorithm.

In particular, we recall that $I_j(t)$ is the inventory level of resource j at the end of period t. Define $c_j(t) = I_j(t-1)/(T-t+1)$ as the average inventory for resource j available from period t to period T. We then replace constants c_j with $c_j(t)$ in $\mathsf{LP}(d(t))$ in step 2 of TS-fixed, which gives us the Thompson sampling with Inventory Constraint Updating algorithm (TS-update for short) shown in Algorithm 7. The term “update” refers to the fact that in every iteration, the algorithm updates inventory constants $c_j(t)$ in $\mathsf{LP}(d(t))$ to incorporate real-time inventory information.

In the revenue management literature, the idea of using updated inventory rates like $c_j(t)$ has been previously studied in various settings (Jasin and Kumar, 2012; Jasin, 2014). TS-update is an algorithm that incorporates real-time inventory updating when the retailer faces an exploration–exploitation trade-off with its pricing decisions. Although intuitively incorporating updated inventory information into the pricing algorithm should improve the performance of the algorithm, Cooper (2002) provides a counterexample where the expected revenue is reduced after the updated inventory information is included. Therefore, it is not immediately clear if TS-update would achieve higher revenue than TS-fixed. We will rigorously analyze the performance of both TS-fixed and TS-update in the next section; our numerical simulation shows that in fact there are situations where TS-update outperforms TS-fixed and vice versa.

Algorithm 7 Thompson sampling with inventory constraint updating (TS-update)

```
Repeat the following steps for all periods $t = 1, \dots, T$:
1. Sample Demand: Sample a random parameter $\theta(t) \in \Theta$ according to the posterior distribution
   of $\theta$ given history $\mathcal{H}_{t-1}$. Let the mean demand under $\theta(t)$ be $d(t) = \{d_{ik}(t)\}_{i \in [N], k \in [K]}$.
2. Optimize Prices given Sampled Demand: Solve the following linear program, denoted by
   $\mathsf{LP}(d(t), c(t))$:
```

$$
\begin{aligned}
\mathsf{LP}(d(t), c(t)) : \quad & \max_x \sum_{k=1}^{K} \Big(\sum_{i=1}^{N} p_{ik} d_{ik}(t)\Big) x_k \\
\text{subject to } & \sum_{k=1}^{K} \Big(\sum_{i=1}^{N} a_{ij} d_{ik}(t)\Big) x_k \le c_j(t), \ \forall j \in [M] \\
& \sum_{k=1}^{K} x_k \le 1 \\
& x_k \ge 0, \ k \in [K].
\end{aligned}
$$

```
   Let $x(t) = (x_1(t), \dots, x_K(t))$ be the optimal solution to $\mathsf{LP}(d(t), c(t))$.
3. Offer Price: Offer price vector $P(t) = p_k$ with probability $x_k(t)$, and choose $P(t) = p_\infty$ with
   probability $1 - \sum_{k=1}^{K} x_k(t)$.
4. Update Estimate of Parameter: Observe demand $D(t)$. Update the history $\mathcal{H}_t = \mathcal{H}_{t-1} \cup$
   $\{P(t), D(t)\}$ and the posterior distribution of $\theta$ given $\mathcal{H}_t$.
```

5.4.4 Performance Analysis

To evaluate the proposed Bayesian learning algorithms, we compare the retailer's revenue with a benchmark where the true demand distribution is known a priori. We define the retailer's *regret* over the selling horizon as

$$\text{Regret}(T, \theta) = \text{E}[\text{Rev}^*(T) \mid \theta] - \text{E}[\text{Rev}(T) \mid \theta],$$

where $\text{Rev}^*(T)$ is the revenue achieved by the optimal policy if the demand parameter θ is known a priori, and $\text{Rev}(T)$ is the revenue achieved by an algorithm that may not know θ. The conditional expectation is taken on random demand realizations given θ and possibly on some external randomization used by the algorithm (e.g., random samples in Thompson sampling). In words, the regret is a nonnegative quantity measuring the retailer's revenue loss due to not knowing the latent demand parameter.

We also define the *Bayesian regret* (also known as *Bayes risk*) by

$$\text{BayesRegret}(T) = E[\text{Regret}(T, \theta)],$$

where the expectation is taken over the prior distribution of θ.

We now prove regret bounds for TS-fixed and TS-update under the realistic assumption of bounded demand. Specifically, in the following analysis, we further assume that for each product $i \in [N]$, the demand $D_i(t)$ is bounded by $D_i(t) \in [0, \bar{d}_i]$ under any price vector p_k, $\forall k \in [K]$. However, the result can be generalized when the demand is unbounded and follows a sub-Gaussian distribution. We also define the constants

$$p_{\max} := \max_{k \in [K]} \sum_{i=1}^{N} p_{ik} \bar{d}_i, \quad p^j_{\max} := \max_{i \in [N]: a_{ij} \neq 0, k \in [K]} \frac{p_{ik}}{a_{ij}}, \ \forall j \in [M],$$

where $p_{\max}$ is the maximum revenue that can possibly be achieved in one period, and $p^j_{\max}$ is the maximum revenue that can possibly be achieved by adding one unit of resource j, $\forall j \in [M]$.

Theorem 5 *The Bayesian regret of* TS-fixed *is bounded by*

$$\text{BayesRegret}(T) \leq \left(18 p_{\max} + 37 \sum_{i=1}^{N} \sum_{j=1}^{M} p^j_{\max} a_{ij} \bar{d}_i \right) \sqrt{TK \log K}.$$

Theorem 6 *The Bayesian regret of* TS-update *is bounded by*

$$\text{BayesRegret}(T) \leq \left(18 p_{\max} + 40 \sum_{i=1}^{N} \sum_{j=1}^{M} p^j_{\max} a_{ij} \bar{d}_i \right) \sqrt{TK \log K} + p_{\max} M.$$

The results above state that the Bayesian regrets of both TS-fixed and TS-update are bounded by $O(\sqrt{TK \log K})$, where K is the number of price vectors that the retailer is allowed to use and T is the number of time periods. Moreover, the regret bounds are *prior-free* as they do not depend on the prior distribution of parameter θ; the constants in the bounds can be computed explicitly without knowing the demand distribution.

It has been shown that for a multi-armed bandit problem with reward in [0, 1]—a special case of our model with no inventory constraints—no algorithm can achieve a prior-free Bayesian regret smaller than $\Omega(\sqrt{KT})$ (see Theorem 3.5, Bubeck and Cesa-Bianchi 2012). In that sense, the above regret bounds are optimal with respect to T and cannot be improved by any other algorithm by more than $\sqrt{\log K}$.

Note that the regret bound of TS-update is slightly worse than the regret bound of TS-fixed. Although intuition would suggest that updating inventory information in TS-update will lead to better performance than TS-fixed, this intuition is somewhat surprisingly not always true—we can find counterexamples where updating inventory information actually deteriorates the performance for any given horizon length T.

The detailed proofs of Theorems 5 and 6 are omitted. We briefly summarize the intuition behind the proofs. For both Theorems 5 and 6, we first assume an "ideal" scenario where the retailer is able to collect revenue even after inventory runs out. We show that if prices are given according to the solutions of TS-fixed or TS-update, the expected revenue achieved by the retailer is within $O(\sqrt{T})$ compared to the optimal revenue $Rev^*(T)$. However, this argument overestimates the expected revenue. In order to compute the actual revenue given constrained inventory, we should account for the amount of revenue that is associated with lost sales. For Theorem 5 (TS-fixed), we prove that the amount associated with lost sales is no more than $O(\sqrt{T})$. For Theorem 6 (TS-update), we show that the amount associated with lost sales is no more than $O(1)$.

5.5 Remarks and Further Reading

The content of Sect. 5.2 is based on Wang (2012) and Wang et al. (2014). For the proofs of the main results, the readers are referred to Wang et al. (2014). In Wang et al. (2014), there are also implementation suggestions for the proposed algorithms. Note that in practical implementation, the algorithm can be made more efficient by relaxing some requirements stated in the Algorithm 1. Extensive numerical experiments and comparison with other algorithms can be found in Wang (2012) and Wang et al. (2014). Later, Lei et al. (2014) improve the result of Theorem 1 to remove the logarithmic factor in the worst-case regret using a bisection type of method. For details of the algorithm and the analysis, we refer the readers to Lei et al. (2014).

Section 5.3 is adapted from Chen et al. (2019) and Chen et al. (2021), which contain full proofs of the theorems presented and additional numerical studies. Chen et al. (2021) further considers a well-separated condition of demand functions and derive a much sharper $O(\log^2 k)$ regret than the $O(\sqrt{k})$ regret in the general demand case.

Section 5.4 is primarily based on Ferreira et al. (2018). The definition of Bayesian regret used in this section is a standard metric for the performance of online Bayesian algorithms, see Russo and Van Roy (2014). Ferreira et al. (2018) also developed the Thompson sampling algorithms for the linear demand case and the *bandits with knapsack* problem, see Badanidiyuru et al. (2013).

Other methods have been proposed in the literature to address learning and pricing problems in the constrained inventory setting. One approach is to separate the selling season (T periods) into a disjoint exploration phase (say, from period 1 to τ) and exploitation phase (from period $\tau + 1$ to T) (Besbes and Zeevi, 2009, 2012). One drawback of this strategy is that it does not use purchasing data after period τ to continuously refine demand estimates. Furthermore, when there is very limited inventory, this approach is susceptible to running out of inventory during the exploration phase before any demand learning can be exploited. Another approach is to use multi-armed bandit methods such as the upper confidence bound (UCB) algorithm (Auer et al., 2002) to make pricing decisions in each period. The UCB algorithm creates a confidence interval for unknown demand using purchase data and then selects a price that maximizes revenue among all parameter values in the confidence set. We refer the readers to Badanidiyuru et al. (2013) and Agrawal and Devanur (2014) for UCB algorithms with constrained inventory.

Acknowledgments This chapter is partially based on material copyrighted by INFORMS and is republished with permission.

References

Agrawal, S., & Devanur, N. R. (2014). Bandits with concave rewards and convex knapsacks. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation* (pp. 989–1006)

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

Badanidiyuru, A., Kleinberg, R., & Slivkins, A. (2013). Bandits with knapsacks. In *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)* (pp. 207–216).

Besbes, O., & Zeevi, A. (2009). Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, *57*(6), 1407–1420.

Besbes, O., & Zeevi, A. (2012). Blind network revenue management. *Operations Research*, *60*(6), 1537–1550.

Broder, J., & Rusmevichientong, P. (2012). Dynamic pricing under a general parametric choice model. *Operations Research*, *60*(4), 965–980.

Bubeck, S., & Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, *5*(1), 1–122.

Chen, Q., Jasin, S., & Duenyas, I. (2019). Nonparametric self-adjusting control for joint learning and optimization of multiproduct pricing with finite resource capacity. *Mathematics of Operations Research*, *44*(2), 601–631.

Chen, Q., Jasin, S., & Duenyas, I. (2021). Joint learning and optimization of multi-product pricing with finite resource capacity and unknown demand parameters. *Operations Research*, *69*(2), 560–573.

Cooper, W. L. (2002). Asymptotic behavior of an allocation policy for revenue management. *Operations Research*, *50*(4), 720–727.

Ferreira, K. J., Simchi-Levi, D., & Wang, H. (2018). Online network revenue management using Thompson sampling. *Operations Research*, *66*(6), 1586–1602.

Gallego, G., & van Ryzin, G. (1994). Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, *40*(8), 999–1029.

Gallego, G., & Van Ryzin, G. (1997). A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, *45*(1), 24–41.

Gyorfi, L., Kohler, M., Krzyzak, A., & Walk, H. (2002). *A distribution-free theory of nonparametric regression*. Springer.

Jasin, S. (2014). Reoptimization and self-adjusting price control for network revenue management. *Operations Research*, *62*(5), 1168–1178.

Jasin, S., & Kumar, S. (2012). A re-solving heuristic with bounded revenue loss for network revenue management with customer choice. *Mathematics of Operations Research*, *37*(2), 313–345.

Lei, Y. M., Jasin, S., & Sinha, A. (2014). Near-optimal bisection search for nonparametric dynamic pricing with inventory constraint, in *Working Paper*.

Russo, D., & Van Roy, B. (2014). Learning to optimize via posterior sampling. *Mathematics of Operations Research*, *39*(4), 1221–1243.

Schumaker, L. (2007). *Spline functions: Basic theory* (3rd ed.). Cambridge University Press.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3/4), 285–294.

Wang, Z. (2012). *Dynamic learning mechanism in revenue management problems*. PhD thesis, Stanford University, Palo Alto.

Wang, Z., Deng, S., & Ye, Y. (2014). Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. *Operations Research*, *62*(2), 318–331.