
Technical Note—Dynamic Pricing and Demand Learning with Limited Price
Experimentation

Author(s): Wang Chi Cheung, David Simchi-Levi and He Wang

Source: *Operations Research*, Vol. 65, No. 6 (November–December 2017), pp. 1722–1731

Published by: INFORMS

Stable URL: <https://www.jstor.org/stable/45111680>

Accessed: 13-02-2025 03:27 UTC

JSTOR is a not-for-profit service that helps scholars, researchers, and students discover, use, and build upon a wide range of content in a trusted digital archive. We use information technology and tools to increase productivity and facilitate new forms of scholarship. For more information about JSTOR, please contact support@jstor.org.

Your use of the JSTOR archive indicates your acceptance of the Terms & Conditions of Use, available at <https://about.jstor.org/terms>



JSTOR


INFORMS is collaborating with JSTOR to digitize, preserve and extend access to *Operations Research*

Technical Note—Dynamic Pricing and Demand Learning with Limited Price Experimentation

Wang Chi Cheung,^a David Simchi-Levi,^b He Wang^c

^aInstitute of High Performance Computing, Agency for Science, Technology and Research, Singapore 138632; ^bDepartment of Civil and Environmental Engineering, MIT Institute for Data, Systems, and Society, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139; ^cSchool of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332

Contact: cheungwc@ihpc.a-star.edu.sg (WCC); dslevi@mit.edu (DS-L); he.wang@isye.gatech.edu,

 <http://orcid.org/0000-0001-7444-2053> (HW)

Received: September 21, 2015

Revised: March 5, 2016; May 2, 2016;
November 18, 2016; February 26, 2017

Accepted: March 28, 2017

Published Online in Articles in Advance:
August 2, 2017

Subject Classifications: marketing: pricing;
production/scheduling: learning

Area of Review: Stochastic Models

<https://doi.org/10.1287/opre.2017.1629>

Copyright: © 2017 INFORMS

Abstract. In a dynamic pricing problem where the demand function is not known a priori, price experimentation can be used as a demand learning tool. Existing literature usually assumes no constraint on price changes, but in practice, sellers often face business constraints that prevent them from conducting extensive experimentation. We consider a dynamic pricing model where the demand function is unknown but belongs to a known finite set. The seller is allowed to make at most m price changes during T periods. The objective is to minimize the worst-case regret—i.e., the expected total revenue loss compared with a clairvoyant who knows the demand distribution in advance. We demonstrate a pricing policy that incurs a regret of $O(\log^{(m)} T)$, or m iterations of the logarithm. Furthermore, we describe an implementation of this pricing policy at Groupon, a large e-commerce marketplace for daily deals. The field study shows significant impact on revenue and bookings.

Funding: This work was supported in part by Groupon and the Accenture-MIT Alliance in Business Analytics.

Supplemental Material: The e-companion is available at <https://doi.org/10.1287/opre.2017.1629>.

Keywords: revenue management • dynamic pricing • learning–earning trade-off • price experimentation

1. Introduction

This paper considers a general learning and pricing problem motivated by the daily deal website Groupon. Groupon is a large e-commerce marketplace where customers can purchase discount deals from local merchants such as restaurants, spas, and housecleaning services. Every day, many new deals are launched on Groupon's website. Because of its business model, Groupon is faced with high levels of demand uncertainty, mainly because there is no previous sale for the newly launched deals that can be used for demand forecasting. This challenge presents an opportunity for Groupon to learn about customer demand using real-time sales data after deals have been launched so as to obtain more accurate demand estimation and adjust prices.

Generally speaking, in revenue management, when the underlying relationship between demand and price is unknown a priori, the seller can use price experimentation for demand learning. In this paper, we consider a dynamic pricing model where the exact demand function is unknown but belongs to a finite set of possible demand functions, or demand hypotheses. The seller faces an exploration–exploitation trade-off between actively adjusting price to gather demand information and optimizing price for revenue maximization.

Dynamic pricing under a finite set of demand hypotheses has previously been considered by Rothschild (1974) and Harrison et al. (2012). But unlike the current paper, both of these papers focus on customized pricing, where price is changed for every arriving customer. For example, the motivation of Harrison et al. (2012) is pricing for financial services such as consumer and auto loans, where sellers can quote a different interest rate for each customer.

However, for many e-commerce companies such as Groupon, charging a different price for each arriving customer is impossible, either because of implementation constraints or for fear of confusing customers and receiving negative customer response. In our collaboration, Groupon stipulated as a rule that the number of price changes has to be as few as possible for each deal, so that the customers would not observe frequent price changes. Motivated by this practical business constraint on price experimentation, the model in this paper includes an explicit constraint on the number of price changes during the sales horizon. We quantify the impact of this constraint on the seller's revenue using *regret*, defined as the gap between the revenue of a clairvoyant who has full information on the demand function and the revenue achieved by a seller facing unknown demand.

Our main finding is a characterization of regret as a function of the number of price changes allowed. When there are T periods in the sales horizon, we propose a pricing policy with at most m price changes, whose regret is bounded by $O(\log^{(m)} T)$, or m iterations of the logarithm.

A natural question is how frequently one needs to change price to achieve a constant regret. Harrison et al. (2012) show that a semimyopic policy can achieve a constant regret, but the policy requires changing price for every time period. To answer this question, we show that a modified version of our algorithm with no more than $O(\log^* T)$ price changes, where $\log^* T$ is the smallest number m such that $\log^{(m)} T \leq 1$, achieves a constant regret.

This characterization of the regret bound shows that the incremental effect of price changes decreases quickly. The first price change reduces regret from $O(T)$ to $O(\log T)$; each additional price change thereafter compounds a logarithm to the order of regret. As a result, the first few price changes generate most of the benefit of dynamic pricing. Interestingly, while the value $\log^* T$ is unbounded when T increases, its growth rate is extremely slow. For example, if the number of time periods T is less than three million, we still have $\log^* T$ no greater than 3.

Motivated by these results, we implemented a pricing strategy at Groupon where each deal can have at most one price change. The result of a field experiment shows significant improvement in both revenue and deal bookings at Groupon.

2. Related Literature

Joint learning and pricing problems have received extensive research attention over the last decade. Recent surveys by Aviv and Vulcano (2012) and den Boer (2015) provide a comprehensive overview of this area. Some papers that consider price experimentation for learning demand curves include Besbes and Zeevi (2009), Boyacı and Özer (2010), Wang et al. (2014), and Besbes and Zeevi (2015). These problems typically focus on the trade-offs between learning and earning, which is closely related to the multiarmed bandit literature (e.g., Kleinberg and Leighton 2003, Mersereau et al. 2009, Rusmevichientong and Tsitsiklis 2010).

Recently, a stream of papers has focused on semimyopic pricing policies using various learning methods. Examples include maximum likelihood estimation (Broder and Rusmevichientong 2012), Bayesian methods (Harrison et al. 2012), maximum quasilielihood estimation (den Boer and Zwart 2014, den Boer 2014), and iterative least-squares estimation (Keskin and Zeevi 2014).

Unlike the current paper, all the literature mentioned above does not assume any constraint on price experimentation. In the dynamic pricing literature with known demand distribution, several papers

consider limited price changes; examples include Feng and Gallego (1995), Bitran and Mondschein (1997), Netessine (2006), and Chen et al. (2015). Caro and Gallien (2012) report that the fashion retailer Zara uses a clearance pricing policy with a predetermined price ladder, which essentially allows for only a limited number of markdown prices. Zbaracki et al. (2004) provide empirical results on the cost of price changes.

To the best of our knowledge, the only work that considers price-changing constraints in an unknown demand setting is Broder (2011). The author assumes that the demand function belongs to a known parametric family (e.g., linear family) but has unknown parameters. He shows that to achieve the optimal regret, a pricing policy needs at least $\Theta(\log T)$ price changes. However, the result only applies to a restricted class of policies where the seller cannot use any knowledge of T .

Our model is different from the model by Broder (2011) in the following aspects. First, we assume a finite number of demand hypotheses, while Broder (2011) assumes a parametric family of demand functions. This is a fundamental difference because the optimal regret in Broder's case is $\Theta(\sqrt{T})$, while in our case the regret can be bounded by a constant. Second, we do not assume a restricted class of policies as in Broder (2011), and our results hold for any pricing policies. Last but not least, unlike Broder (2011), where the number of price changes is an output from the model, we design a pricing algorithm that accepts the number of price changes as an input constraint and achieves the best possible regret bound under that constraint.

3. Problem Formulation

We consider a seller offering a single product with unlimited supply for T periods. The set of allowable prices is denoted by \mathcal{P} . For example, \mathcal{P} can be an interval $[p, \bar{p}]$ or a finite set $\{p_1, \dots, p_k\}$, although no restriction on \mathcal{P} is assumed.

In the t th period ($t = 1, \dots, T$), the seller offers a unit price $P_t \in \mathcal{P}$. Then, she observes the realized customer demand X_t (i.e., the number of units purchased in the t th period). Given $P_t = p$, the distribution of X_t is only determined by price p and is independent of previous prices and demands $\{P_1, X_1, \dots, P_{t-1}, X_{t-1}\}$. We use $D(p)$ to denote a generic random variable for demand given price p . Thus, given $P_t = p$, we have $X_t \sim D(p)$. The corresponding mean demand function $d: \mathcal{P} \rightarrow \mathbb{R}_+$ is defined as $d(p) = \mathbb{E}[D(p)]$.

The distribution of $D(p)$ is unknown to the seller. However, the seller knows that the mean demand function $d(p)$ is equal to one of the K given mean demand functions: $d_1(p), \dots, d_K(p)$. We call the collection $\Phi = \{d_1(\cdot), \dots, d_K(\cdot)\}$ the demand hypothesis set. The seller does not necessarily know the distribution associated with each demand hypothesis i ($\forall i = 1, \dots, K$) apart from the mean $d_i(p)$. For each demand

function $d_i \in \Phi$, the expected revenue per period is denoted by $r_i(p) = p d_i(p)$. We also denote the optimal revenue for demand function d_i by $r_i^* = \max_{p \in \mathcal{P}} r_i(p)$ and an optimal price by $p_i^* \in \arg \max_{p \in \mathcal{P}} r_i(p)$.

For all $p \in \mathcal{P}$, given the mean demand function d_i , the probability distribution of $D(p)$ is assumed to be *light-tailed* (sub-exponential) with parameters (σ, b) , where $\sigma, b > 0$. That is, we have $\mathbb{E}_i[e^{\lambda(D(p)-d_i(p))}] \leq \exp(\lambda^2 \sigma^2 / 2)$ for all $|\lambda| < 1/b$. Note that the class of light-tailed distributions includes all sub-Gaussian distributions. Some common light-tailed distributions include normal, Poisson, and Gamma distributions, as well as all distributions with bounded support, such as binomial and uniform distributions.

3.1. Pricing Policies

We say that π is a nonanticipating pricing policy if the price P_t^π offered by π at period t is determined by the time horizon T and the sales history \mathcal{H}_t . History \mathcal{H}_t consists of the realized demand (X_1, \dots, X_{t-1}) , the previous offered prices $P_1^\pi, \dots, P_{t-1}^\pi$, as well as the length of sales horizon T . Importantly, P_t^π should be independent of future demand. We express the dependence of P_t^π on the sales history in the following:

$$P_t^\pi = \pi(X_1, \dots, X_{t-1}; T). \quad (1)$$

Note that we make the dependence on the previous offered prices $P_1^\pi, \dots, P_{t-1}^\pi$ implicit in the expression, as they are determined by the purchase decision X_1, \dots, X_{t-1} and T . Additional explanation is provided in the appendix. The price offered in period 1, $P_1^\pi = \pi(\emptyset; T)$, is only determined only by π and T , in the absence of sales history.

For $i = 1, \dots, K$, let $\mathbb{P}_i^\pi[\cdot]$ and $\mathbb{E}_i^\pi[\cdot]$ be the probability measure and expectation induced by policy π if the underlying demand model is i . In this case, the seller's expected revenue in T periods under policy π is given by

$$\begin{aligned} R_i^\pi(T) &= \mathbb{E}_i^\pi \left[\sum_{t=1}^T P_t X_t \right] = \mathbb{E}_i^\pi \left[\sum_{t=1}^T P_t \mathbb{E}_i^\pi[X_t | P_t] \right] \\ &= \mathbb{E}_i^\pi \left[\sum_{t=1}^T r_i(P_t) \right]. \end{aligned} \quad (2)$$

As motivated earlier, in many revenue management applications, the seller faces a constraint on the number of price changes. In the model, we assume that the seller can make at most m changes to the price over the course of the sales event, where m is a fixed integer. A feasible policy π should therefore satisfy the following condition:

$$\mathbb{P}_i^\pi \left[\sum_{t=2}^T I(P_t \neq P_{t-1}) \leq m \right] = 1, \quad \forall i = 1, \dots, K,$$

where $I(\cdot)$ is the indicator function. We refer to a policy with at most m price changes (w.p. 1) as an m -change policy.

The performance of a pricing policy is measured against the optimal policy in the full information case. If the true demand is d_i , then a clairvoyant with full knowledge of the demand function would offer price p_i^* and obtain expected revenue r_i^* for every period. The *regret* with respect to demand d_i is defined as the gap between the expected revenue achieved by the clairvoyant and the one achieved by policy π —namely,

$$\text{Regret}_i^\pi(T) = T r_i^* - R_i^\pi(T) = \mathbb{E}_i^\pi \left[\sum_{t=1}^T (r_i^* - r_i(P_t)) \right]. \quad (3)$$

Finally, we define the minimax regret for the demand set, $\Phi = \{d_1, \dots, d_K\}$, as

$$\text{Regret}_\Phi^\pi(T) = \max_{i=1, \dots, K} \text{Regret}_i^\pi(T).$$

When there is no ambiguity of which policy we are referring to, we suppress the superscript “ π ” in the notation for clarity; namely, $\mathbb{E}_1 := \mathbb{E}_1^\pi$, $\mathbb{P}_1 := \mathbb{P}_1^\pi$.

3.2. Notations

For two sequences $\{a_n\}$ and $\{b_n\}$ ($n = 1, 2, \dots$), we write $a_n = O(b_n)$ if there exists a constant C such that $a_n \leq C b_n$ for all n ; we write $a_n = \Omega(b_n)$ if there exists a constant c such that $a_n \geq c b_n$ for all n . We use $\log^{(m)} T$ to represent m iterations of the logarithm, $\log(\log(\dots \log(T)))$, where m is the number of price changes. For convenience, we let $\log(x) = 0$ for all $0 \leq x < 1$, so the function $\log^{(m)} T$ is defined for all $T \geq 1$. Similarly, we define the notations $e^{(0)}(T) := T$ and $e^{(\ell)}(T) := \exp(e^{(\ell-1)}(T))$ for $\ell \geq 1$. We also define the short-hand notation $e^{(\ell)} := e^{(\ell)}(1)$. As mentioned earlier, function $\log^* T$ denotes the smallest nonnegative integer m such that $\log^{(m)} T \leq 1$. For any real number x , we denote by $\lceil x \rceil$ the minimum integer greater than or equal to x . For any finite set S , the cardinality of S is denoted by $|S|$. We occasionally use notations $a \vee b = \max\{a, b\}$ and $a \wedge b = \min\{a, b\}$.

4. Main Results: Upper Bounds on Regret

In this section we prove the main results of the paper: an upper bound on the regret as a function of the number of price changes. We design a nonanticipating pricing policy that changes price no more than m times and achieves a regret of $O(\log^{(m)} T)$.

4.1. Upper Bound

We propose a policy mPC (which stands for “ m -price change”) that achieves a regret of $O(\log^{(m)} T)$ with at most m price changes. An important feature of policy mPC is that it applies a *discriminative* price for every period. A price p is *discriminative* if demands $d_1(p), \dots, d_K(p)$ are mutually distinct.

We make the following assumption on the set of demand functions Φ .

Assumption 1. For all $d_i \in \Phi = \{d_1, \dots, d_K\}$, there exists a corresponding revenue-optimal price $p_i^* \in \arg \max_{p \in \mathcal{P}} r_i(p)$ such that p_i^* is a discriminative price for Φ_i ; that is, $d_1(p_i^*), \dots, d_K(p_i^*)$ are distinct. Moreover, such a price p_i^* can be efficiently computed.

Assumption 1 ensures that the seller is able to learn the underlying demand curve while maximizing its revenue for any given demand function $d_i \in \Phi$. In fact, we will show in Section 4.3 that this condition is both sufficient and necessary for achieving a regret bound better than $o(\log T)$.

Algorithm 1 (m -Change policy mPC)

- 1: INPUT:
 - A set of demand functions $\Phi = \{d_1, \dots, d_K\}$.
 - A discriminative price P_0^* .
- 2: (Learning) Set $\tau_0 = 0$.
- 3: **for** $\ell = 0, \dots, m-1$ **do**
- 4: **if** $\log^{(m-\ell)} T = 0$ **then**
- 5: Set $\tau_{\ell+1} = 0$ and $P_{\ell+1}^* = P_\ell^*$.
- 6: **else**
- 7: From period $\tau_\ell + 1$ to $\tau_{\ell+1} := \tau_\ell + \lceil M_\Phi(P_\ell^*) \log^{(m-\ell)} T \rceil$, set the offered price as P_ℓ^* .
- 8: At the end of period $\tau_{\ell+1}$, compute the sample mean \bar{X}^ℓ from period $\tau_\ell + 1$ to $\tau_{\ell+1}$: $\bar{X}^\ell := \frac{\sum_{j=\tau_\ell+1}^{\tau_{\ell+1}} X_j}{\tau_{\ell+1} - \tau_\ell}$, where X_j = number of items sold in period j .
- 9: Choose an index $i_\ell \in \{1, \dots, K\}$ that solves $\min_{i \in \{1, \dots, K\}} |\bar{X}^\ell - d_i(P_\ell^*)|$.
- 10: Set the next offered price as $P_{\ell+1}^* = p_{i_\ell}^*$, where $p_{i_\ell}^*$ is the optimal price for demand d_{i_ℓ} .
- 11: **end if**
- 12: **end for**
- 13: (Earning) From period $\tau_m + 1$ to period $\tau_{m+1} = T$, set the selling price as P_m^* .

Algorithm 1 describes the mPC policy. The policy partitions the finite time horizon $1, \dots, T$ into $m+1$ phases. For each $0 \leq \ell \leq m$, a single price P_ℓ^* is offered through phase ℓ , which starts at period $\tau_\ell + 1$ and ends at $\tau_{\ell+1}$. Phases 0 to $m-1$ are called the *learning phases*, and phase m is referred to as the *earning phase*. Except for a constant factor $M_\Phi(P_\ell^*)$, which is to be defined later, the lengths of phases are iterated-exponentially (tetrationally) increasing, which ensures an optimal balance between exploration and exploitation.

At the end of learning phase ℓ , policy mPC computes the sample mean \bar{X}^ℓ of the sales under price P_ℓ^* (in line 8 of the algorithm). Since price P_ℓ^* is discriminative, the seller gains new information about the underlying demand in this learning phase. She then updates her belief on the true demand distribution to be $d_{i_{\ell+1}}$

(in line 9) and sets the offered price $P_{\ell+1}^*$ to be $p_{i_{\ell+1}}^*$ in the next phase. In going through all the learning phases, the seller progressively refines her estimate on the optimal price, which enables her to establish the choice of optimal price in the earning phase.

The function $M_\Phi(P)$ in line 7 of the mPC algorithm is defined as follows.

Definition 1. Let $p \in \mathcal{P}$ be a discriminative price. We define $M_\Phi(p)$ as

$$M_\Phi(p) := \frac{16\sigma^2}{\min_{i \neq j} (d_i(p) - d_j(p))^2} \vee \frac{8b}{\min_{i \neq j} |d_i(p) - d_j(p)|}, \quad (4)$$

where the minimum is taken over distinct pairs of indices $i, j \in \{1, \dots, K\}$.

Since we assume p to be discriminative, $M_\Phi(p)$ is well defined. The function $M_\Phi(p)$ measures the distinguishability of the demand functions d_1, \dots, d_K under the discriminative price p . We explain the definition of $M_\Phi(p)$ further in the analysis of mPC.

Introduce notations $M_\Phi^* = \max_{i \in \{1, \dots, K\}} M_\Phi(p_i^*)$ and $r^* = \max_{i \in \{1, \dots, K\}} r_i^*$. The following result shows that the regret of mPC is bounded by $O(\log^{(m)} T)$.

Theorem 1. Suppose the demand set Φ satisfies Assumption 1. For all $T \geq 1$, the regret of mPC is bounded by

$$\text{Regret}_\Phi^{\text{mPC}}(T) \leq C_\Phi(P_0^*) \max\{\log^{(m)} T, 1\} + 4(M_\Phi^* + 1)r^*,$$

where $C_\Phi(P_0^*) = \max_{i \in \{1, \dots, K\}} \{M_\Phi(P_0^*)(r_i^* - r_i(P_0^*))\}$.

Proof Idea of Theorem 1. In the proof, we establish that the regret incurred in phase 0 is $O(\log^{(m)} T)$, and the cumulative regret incurred in the remaining phases is $O(1)$. At the beginning of phase 0, which is also the beginning of the sale horizon, the seller has no information on the optimal price. Thus, the regret during phase 0 is proportional to the length of phase 0. However, in each of the subsequent phases, the seller can choose a price based on the previous sale history. By choosing the lengths of the subsequent phases appropriately, we ensure that the total regret in these phases is $O(1)$. The complete proof can be found in Online Appendix EC.1. \square

Remark 1. In phase 0, the discriminative price P_0^* is given as an input. One can further reduce the regret bound by choosing a discriminative price P_0^* , which minimizes the regret during phase 0—namely, $C_\Phi(P_0^*)$.

Remark 2. In line 9 of Algorithm 1, the test to select a demand function d_{i_ℓ} is a simple comparison between the sample mean \bar{X}^ℓ and the mean demand function value $d_{i_\ell}(P_\ell^*)$. Therefore, the algorithm does not require the seller to know the demand distributions for each

demand model. Nevertheless, if the seller does know the demand distribution, line 9 can be replaced by other selection criteria, such as a likelihood ratio test, to improve the efficiency of learning.

4.2. Unbounded but Infrequent Price Experiments

Policy mPC defines m learning phases with iterated-exponentially increasing lengths. This motivates us to consider a modification of mPC, which improves the regret bound to a constant. We call this modified policy uPC (which stands for “unbounded price changes”); see Algorithm 2. Although the number of price changes under this policy is not bounded by any finite number as T increases, it grows extremely slowly with order $O(\log^* T)$, where $\log^* T = \min\{m \in \mathbb{Z}^+: \log^{(m)} T \leq 1\}$. For example, for $T \leq 3,000,000$, we have $\log^* T \leq 3$.

Algorithm 2 (Policy uPC)

```

1: INPUT:
    • A set of demand functions  $\Phi = \{d_1, \dots, d_K\}$ .
    • A discriminative price  $P_0^*$ .
2: Set  $\tau_0 = 0$ .
3: for  $\ell = 0, 1, \dots$  do
4:   From period  $\tau_\ell + 1$  to  $\tau_{\ell+1} := \tau_\ell + \lceil M_\Phi(P_\ell^*)e^{(\ell)} \rceil$ ,
     set the offered price as  $P_\ell^*$ .
5:   if  $T \leq \tau_{\ell+1}$  then stop the algorithm
     at period  $T$ .
6:   else
7:     At the end of period  $\tau_{\ell+1}$ , compute the
       sample mean  $\bar{X}^\ell$  from period  $\tau_\ell + 1$  to  $\tau_{\ell+1}$ :
        $\bar{X}^\ell := \frac{\sum_{j=\tau_\ell+1}^{\tau_{\ell+1}} X_j}{\tau_{\ell+1} - \tau_\ell}$ , where  $X_j$  = number
       of items sold in period  $j$ .
8:     Choose an index  $i_\ell \in \{1, \dots, K\}$ ,
       which solves
       
$$\min_{i \in \{1, \dots, K\}} |\bar{X}^\ell - d_i(P_\ell^*)|.$$

9:     Set the next offered price as  $P_{\ell+1}^* = p_{i_\ell}^*$ ,
       where  $p_{i_\ell}^*$  is an optimal price for demand  $d_{i_\ell}$ .
10:   end if
11: end for.
```

Proposition 1. Suppose Assumption 1 holds. For all $T \geq 1$, the pricing policy uPC has regret

$$\text{Regret}_\Phi^{\text{uPC}}(T) \leq C_\Phi(P_0^*) + 2(M_\Phi^* + 1)r^*,$$

where $C_\Phi(P_0^*) = \max_{i \in \{1, \dots, K\}} \{M_\Phi(P_0^*)(r_i^* - r_i(P_0^*))\}$.

The proof of Proposition 1 is included in Online Appendix EC.2. Furthermore, uPC is an *anytime* policy, meaning that the seller can apply the uPC algorithm without any knowledge of T . Anytime policies can be used for customized pricing. In customized pricing, each customer arrival is modeled as a single time period, so T is the total number of customer arrivals

(see Harrison et al. 2012, Broder and Rusmevichientong 2012). Since uPC is an anytime policy, the seller is not required to know the total number of customers arrivals.

4.3. Discussion on the Discriminative Price Assumption

The $O(\log^{(m)} T)$ regret of mPC and the $O(1)$ regret of uPC hold under the assumption that there exists an optimal discriminative price for each demand function (Assumption 1). In fact, one can show that this assumption is necessary for any nonanticipating policy to achieve a regret better than $o(\log T)$.

Proposition 2. If Assumption 1 is violated, then there exists a price set \mathcal{P} and a demand set Φ such that any nonanticipating pricing policy incurs a regret of $\Omega(\log T)$, even if that policy is allowed to change price for infinitely many times.

The proof of Proposition 2 is included in Online Appendix EC.3. It implies that the best possible regret bound without Assumption 1 is $O(\log T)$.

The remaining question is whether the lower bound in Proposition 2 is tight given that Assumption 1 does not hold. We show next that for any set of K demand functions, there exists a policy kPC (see Algorithm 3) that achieves regret bound of $O(\log T)$ with at most $K - 1$ price changes, regardless of whether Assumption 1 is satisfied. Before introducing the algorithm, we need the following definition.

Definition 2. For any nonempty subset of demand functions $A \subset \{d_1, \dots, d_K\}$, let

$$\tilde{p}_A \in \arg \max_{p \in \mathcal{P}} |\{d_i(p) \mid d_i \in A\}|.$$

In other words, \tilde{p}_A is a price that maximizes the number of distinct values of $d_i(p)$ for all $d_i \in A$.

Furthermore, define

$$\tilde{M}_A(p) := \frac{8\sigma^2}{\min_{(i,j): d_i(p) \neq d_j(p)} (d_i(p) - d_j(p))^2} \vee \frac{4b}{\min_{(i,j): d_i(p) \neq d_j(p)} |d_i(p) - d_j(p)|}, \quad (5)$$

where the minimum is taken over all pairs of demand functions $d_i, d_j \in A$ such that $d_i(p) \neq d_j(p)$.

Note that if $|A| \geq 2$, for any pair of demand functions d_i and d_j in A , we can always find a price p such that $d_i(p) \neq d_j(p)$, because otherwise, the two demand functions are identical. So the value $\tilde{M}_A(\tilde{p}_A)$ in line 4 of Algorithm 3 is well defined for any $|A| \geq 2$.

Algorithm 3 (Policy kPC).

```

1: INPUT: A set of demand functions  $\Phi = \{d_1, \dots, d_K\}$ .
2: (Learning) Set  $A \leftarrow \Phi$ . Set  $\ell = 0$ ,  $\tau_0 = 0$ .
3: while  $|A| \neq 1$  do:
```

- 4: Set the price as $P_\ell^* = \tilde{p}_A$ from period $\tau_\ell + 1$ to $\tau_{\ell+1} := \tau_\ell + \lceil \tilde{M}_A(P_\ell^*) \log T \rceil$.
- 5: At the end of period $\tau_{\ell+1}$, compute the sample mean \bar{X}^ℓ from period $\tau_\ell + 1$ to $\tau_{\ell+1}$:

$$\bar{X}^\ell := \frac{\sum_{j=\tau_\ell+1}^{\tau_{\ell+1}} X_j}{\tau_{\ell+1} - \tau_\ell}, \text{ where } X_j = \text{number of items sold in period } j.$$
- 6: Update A : keep all d_i in set A if it is a minimizer of $\min_{d_i \in A} |\bar{X}^\ell - d_i(P_\ell^*)|$. Eliminate other demand functions from A . If there are two minimizers d_i and d_j such that $d_i(P_\ell^*) < \bar{X}^\ell < d_j(P_\ell^*)$, remove d_j and only keep d_i .
- 7: Set $\ell \leftarrow \ell + 1$.
- 8: **end while**
- 9: (Earning) Suppose $A = \{d_i\}$. From period $\tau_\ell + 1$ to period $\tau_{\ell+1} = T$, set the selling price as $P_\ell^* = p_i^*$.

Proposition 3. For all $T \geq 1$, the regret of kPC is bounded by

$$\text{Regret}_\Phi^{\text{kPC}}(T) \leq (K - 1)(\tilde{M}_\Phi r^* \log T + 3r^*),$$

where $\tilde{M}_\Phi = \max_{A \subset \{1, \dots, K\}} \tilde{M}_A(\tilde{p}_A)$.

Proof Idea of Proposition 3. In each of the learning phases, the definition of the algorithm (line 6) guarantees that at least one demand function is eliminated. The number of iterations in the while loop is at most $K - 1$, and thus the regret of the learning phases is $O((K - 1) \log T)$. We then show that with high probability, the single demand function remained in the earning phase is the true demand function. The complete proof is in Online Appendix EC.4. \square

5. Field Experiment at Groupon

We collaborated with Groupon, a large e-commerce marketplace for daily deals, to implement the pricing algorithm presented in Section 4. Groupon offers discount deals from local merchants to subscribed customers. By the second quarter of 2015, Groupon served more than 500 cities worldwide, had nearly 49 million active customers, and featured more than 510,000 active deals.

To illustrate Groupon's business model, consider the Japanese restaurant deal taken from Groupon's website and depicted in Figure 1. The deal can be purchased through Groupon at \$17 and redeemed at the Japanese restaurant for \$30. The amount paid by a customer (\$17) is called "booking." The booking is then split between Groupon and the local merchant. For example, an agreement may allow the local merchant to receive \$12 and Groupon to keep \$5 as its net revenue. In most cases, a deal is only available for a limited time, ranging from several weeks to several months.

Prior to our collaboration, Groupon applied a fixed price strategy for each deal. Our initial analysis suggested that Groupon could benefit from the dynamic pricing algorithm proposed in Section 4 for the following reasons.

Figure 1. (Color online) Screenshot of a Restaurant Deal on the Groupon Website



- A majority of Groupon's deals are offered on its website for the first time, and there are not enough historical data to predict demand before new deals are launched. Thus, there is an opportunity for Groupon to learn from real-time sales data after deals are launched so as to improve its demand forecast and pricing strategies.

- Deals are offered for a limited time, so there is a time trade-off between price experimentation and revenue maximization, which is a trade-off addressed by our pricing algorithm.

- Groupon managers prefer using as few price changes as possible for a number of reasons. First, they are concerned that frequent price changes may confuse customers. Second, it is easy to communicate and explain a simple dynamic pricing algorithm with minimal price changes to merchants.

- Since Groupon mainly offers coupons instead of physical products, we ignored the inventory constraint in this implementation. Technically, each deal has a cap that specifies the maximum quantity that can be sold, but historical data show that only a small fraction of deals have reached their caps. Moreover, once a deal has reached its cap, Groupon can renegotiate with the merchant to increase the cap. So the unlimited inventory assumption is a reasonable approximation of reality.

The pricing algorithm proposed in Section 4 requires a set of possible demand functions as an input. In the rest of this section, we propose a method to generate demand function sets based on clustering. We then describe implementation details and state additional business constraints specified by Groupon. Finally, we present the implementation results and the analysis.

5.1. Generating the Demand Function Set

Recall that we assume a finite demand function set, Φ , in the model assumption. In reality, it is unlikely that the true demand function will belong to the finite set

that we estimated. However, our goal is to find a set Φ , such that the true demand function can be well approximated by at least one function in the set. To this end, we propose the following three-step process to generate a finite set of linear demand functions.¹

—*Step 1.* We first collect data on historical deals that have been tested for dynamic pricing. Given a new deal, we select a subset of historical deals with similar features (e.g., deal category, price range, discount rate). Since deals in this subset were tested for dynamic pricing, they have been offered under at least two different prices, so we can fit a linear demand function for *each* historical deal.

—*Step 2.* The linear demand functions are then mapped into points on a two-dimensional plane according to the following rule: the y -coordinate of the plane represents the negative slope of linear functions, and the x -coordinate represents the mean demand valued at the initial price of the new deal. For example, suppose we fit a demand function $d_i(p) = a - bp$ for a historical deal, and the new deal has an initial price P_1 ; then this demand function is mapped to the point $(a - bP_1, b)$. Using this mapping, each deal in the subset can be represented by a point on the plane, shown as an “x” in Figure 2. Moreover, the mapping is bijective, so there is a one-to-one correspondence between any point on the plane and a linear demand function.

—*Step 3.* We apply K -means clustering to group the points into K clusters. For example, Figure 2 shows the clustering result for $K = 3$. Note that the centroids of the clusters are points on the plane, so according to the bijective mapping defined in Step 2, they also represent linear demand functions. In particular, if a centroid is located at (x, y) , it corresponds to the linear function $d(p) = x + (P_1 - p)y$. Collectively, the centroids of K clusters represent K linear demand functions, which form the demand function set.

Typically, Step 1 would produce hundreds to thousands of historical deals. If we omit Step 3 and simply use linear functions generated in Step 2 as input

to our dynamic pricing algorithm, it would be difficult for the learning algorithm to correctly identify the true demand function among thousands of functions within a short period of time. Therefore, we use clustering in Step 3 to limit the number of demand functions to be learned.

To minimize the demand prediction error, we need to select an appropriate number K to balance the bias–variance trade-off. When K is large, we have a large set of demand functions that can better approximate the true demand function (small bias), but learning about the correct demand function is hard (large variance), since the constant M_Φ^* in the regret bound of Theorem 1 is large. When K is small, the demand function set has a large approximation error relative to the true demand function (large bias), but it is easier to identify the best function in this set (small variance).

To determine the best value of K , we apply cross-validation: the historical deals are randomly split into training and testing sets. Each deal in the testing set had been offered under two prices (p_1, p_2) and is treated as a new deal with initial price (p_1) . We then generate demand functions from the training set following the three-step process described above. After that, we select one function among the K functions whose value at p_1 is the closest to the actual demand of the new deal at p_1 , since this is the function that would have been chosen by our learning algorithm. Next, we compare the realized demand under price p_2 to the mean demand predicted by the selected function at p_2 . The difference between these two values can be interpreted as the prediction error of our learning algorithm. We repeat this process for different values of K and choose a K that minimizes prediction error.

In Figure 3, we plot the mean squared error (MSE) of demand prediction for different values of K . The dark curve is the MSE of purchase quantity per impression (i.e., per customer visit), and the gray curve is the MSE of booking per impression. The figure shows that the prediction error is large for small values of K , and then it decreases as K increases. This implies that for small values of K , none of the demand functions in

Figure 2. Applying K -Means Clustering to Generate K Linear Demand Functions

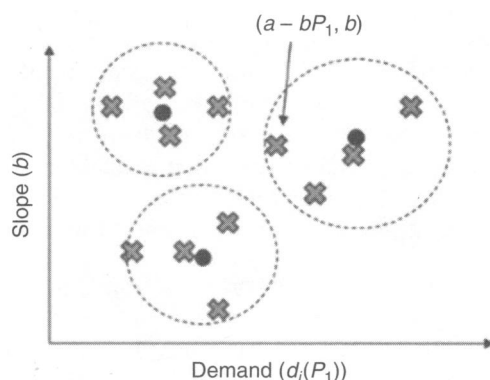
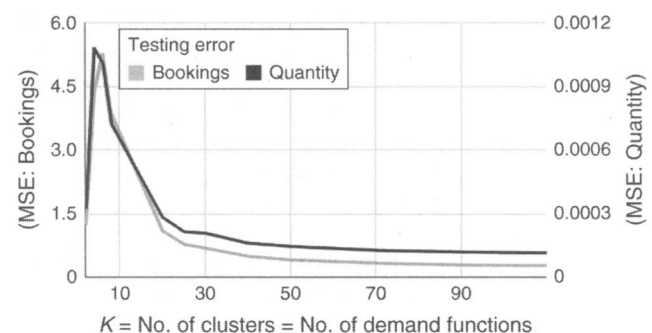


Figure 3. (Color online) Mean Squared Error of Demand Prediction for Different Values of K



the demand set is close to the true demand function (i.e., large bias), so the prediction error is large. For very few demand functions ($2 \leq K \leq 10$), the bias is so large that the prediction error is even worse than that of fixed pricing ($K = 1$). Therefore, it is important to choose a large enough K so that at least one of the demand functions in the demand set is close to the true demand function. We select K to be around 100 in our final implementation at Groupon. Once K is chosen, the demand functions are generated according to Step 3 in the aforementioned process. Notice that the prediction error will eventually go up as a result of overfitting when we select a K that is sufficiently large (i.e., large variance). This trend is not shown in Figure 3, because we did not have enough historical deals in this example to demonstrate overfitting.

5.2. Implementation Details

Compared with the model defined in Section 3, we face some additional constraints in practice because of Groupon's business rules, so our pricing algorithm must be adjusted to include these implementation details.

For each deal, we suppose that the allowable price set is a continuous interval $\mathcal{P} = [p, \bar{p}]$. Each time period is defined as one day, and prices of deals are changed at midnight local time. Since customer traffic for Groupon is not necessarily time homogeneous, in the data preprocessing step, the sales data have been normalized to remove the time effect. More specifically, the normalized sales quantity is the original sales quantity multiplied by a normalization factor, which depends only on time (e.g., number of days after launch, holiday/weekend) and is specific for each deal category. After this normalization step, we can treat demand in different time periods as stationary.

The main result in Section 4 shows that the first price change captures most of the benefit of dynamic pricing, which reduces regret from $\Theta(T)$ to $\Theta(\log T)$. So we decided to apply the single price change policy at Groupon (i.e., the mPC policy defined in Section 4.1 with $m = 1$).

In the Groupon implementation, the initial price of a deal is negotiated by the local merchant and Groupon, so we treat the initial price as a fixed input—this is the same price that Groupon would have used under its existing fixed price policy. Since a finite set of linear demand functions has only finite nondiscriminative prices in the price interval $[p, \bar{p}]$, it is unlikely that the initial price is nondiscriminative. In fact, in all the examples that we tested, the initial price P_1 is discriminative with respect to the demand set.

When changing price, Groupon decided to allow for price decreases only. The main reason for this decision is that many merchants use Groupon as a marketing channel to attract new customers, so Groupon is

unwilling to reduce sales quantity by increasing price, even if it may potentially increase revenue. Groupon further imposes a constraint that price can only be decreased between 5% and 30%. Therefore, if the pricing algorithm recommends either a price decrease of less than 5% or a price increase, then no price change is made. If the algorithm recommends a price decrease of more than 30%, then the price is decreased by only 30%.

Moreover, the agreement between Groupon and the local merchant specifies that the merchant's share of revenue is not affected after price decrease. For example, suppose a deal has an initial price of \$20, and both Groupon and the merchant receive \$10 from each deal purchased under the initial price. If the pricing algorithm reduces the price to, say, \$15, then Groupon would receive \$5 and the merchant would still receive \$10.² This agreement guarantees that merchants always benefit from price decrease, and hopefully, this would make merchants more willing to accept the dynamic pricing policy, which is designed to be a win-win strategy for both Groupon and local merchants. In practice, Groupon always gives the merchants two options: they can either use the existing fixed price policy or choose to use the dynamic pricing algorithm.

5.3. Field Experiment Results

The field experiment at Groupon consists of two stages. In the first stage we focused on fine-tuning the pricing algorithm, while the second stage was used as a final evaluation of the algorithm performance.

In the first stage, we tested different ways to generate demand function sets (Φ). The method presented in the previous subsection was the approach that we finally selected. We also used the first-stage experiment to test different price switching times. In the definition of algorithm mPC for $m = 1$, the price is switched at period $\lceil M_\Phi(P_0) \log T \rceil$, where the constant $M_\Phi(P_0)$ is given by Equation (4). However, this constant is mainly designed for proving the theoretical regret bound. In practice, we tested several price switching times (between one to seven days) through live experiment, and the switching time with the best performance was chosen.

The second (evaluation) stage of the field experiment lasted for several weeks. During this testing period, 1,295 deals were selected by our dynamic pricing algorithm for price decrease. These deals span five product categories: beauty/healthcare, food/drink, leisure/activities, services, and shopping. Table 1 provides information on the number of deals and the average daily bookings per category. We note that if a deal was tested for dynamic pricing but was not selected for price decrease, that deal is not included in this data set.

In the field experiment, we focused on two performance metrics. One is the total amount of money paid

Table 1. Deals Selected in the Field Experiment

	Beauty/health	Food/drink	Leisure/activities	Services	Shopping
Number of deals selected	591	111	259	274	60
Mean bookings per day	35.1	88.0	37.6	27.2	9.3

by customers to Groupon, referred to as *bookings*, and it is directly related to Groupon's market share. The other is the part of the revenue that Groupon keeps after paying local merchants, referred to simply as *revenue*. For each product category, we compare the average bookings and revenue per day before and after the price change. Since the initial price is determined by Groupon's current fixed price policy, comparing the average daily bookings and daily revenue before and after the price change measures the revenue lift of our dynamic pricing policy over Groupon's existing fixed price policy. As we did in the data preprocessing step, the lift has been normalized to control for the time-varying demand effect. Specifically, when generating Figure 4, we multiply the booking and revenue quantities per day by a normalization factor, which depends only on time (e.g., number of days after launch, holiday/weekend) and deal category.

Figure 4 shows the average increase in daily bookings and revenue after price decrease. Overall, daily bookings are increased by 116%, and daily revenue is increased by 21.7%. Among the five categories, beauty/healthcare, food/drink, and shopping have significant increases in both revenue and bookings. Services category has almost no revenue change, but significant bookings increase. The leisure/activities category has a negative gain in revenue. For Groupon, beauty/healthcare and food/drink are the two leading categories in terms of revenue generation. Our pricing algorithm has solid revenue gain in these two categories.

Further analysis of the field experiment result shows that reducing price has a much bigger impact on deals that have fewer bookings per day, which holds across

all categories. Overall, the average increase in daily revenue is 116% for deals with bookings per day below the median, while the increase is only 14% for deals with bookings per day above the median. This effect also explains the big increase in bookings and revenue for the shopping category, the category with the smallest mean daily bookings among all five categories; see Table 1. Therefore, the increases on the shopping deals are also the most significant.

Finally, our pricing algorithm has a poor performance for the leisure/activities category, despite the fact that this category has almost the same level of average daily bookings as the beauty category. We suspect the reason is that some features of customer demand for leisure/activities deals are not captured by our demand model. For example, it might be that the weekend/holiday effect is stronger for this category than we estimated. The data preprocessing step does include time normalization for the weekend/holiday effect, but it is likely that customers purchase leisure/activities deals a few days before holidays, instead of during holidays. Therefore, further work is needed to improve the demand prediction method for the leisure/activities category.

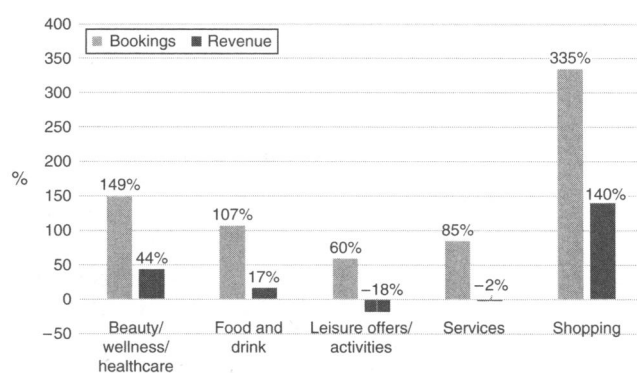
6. Conclusion

We consider a dynamic pricing problem where the latent demand model is unknown but belongs to a finite set of demand functions. The seller faces a constraint that the price can be changed at most m times. We propose a pricing policy that incurs a regret of $O(\log^{(m)} T)$, where T is the length of the sales horizon and $\log^{(m)} T$ is m iterations of logarithm.

We then implement the pricing algorithm at Groupon, a website that sells deals from local merchants. We design a process to generate linear demand function sets from historical data and use it as an input to our pricing algorithm. The algorithm incorporates Groupon's business rules, which allow at most one price decrease per deal. The field experiment shows that the algorithm has significantly improved both daily revenue and bookings per deal.

Acknowledgments

The authors gratefully acknowledge the assistance of Groupon's Data Science department for gathering data, sharing valuable business expertise and helping implement the field experiment. The authors also thank Alex Weinstein of the MIT Operations Research Center for assisting the data analysis in the early stage of this project.

Figure 4. Bookings and Revenue Increase by Deal Category

Appendix. A Note on the Definition of a Nonanticipatory Policy

Conventionally, a pricing policy π is said to be nonanticipatory if

$$P_t^\pi = f(P_1^\pi, X_1, P_2^\pi, \dots, P_{t-1}^\pi, X_{t-1}; T)$$

for some function f . The last argument T is the length of sales horizon. We claim that P_t^π can be alternatively expressed as

$$P_t^\pi = g(X_1, X_2, \dots, X_{t-1}; T)$$

by another suitably defined function g .

The claim is shown by induction on t . For $t = 1$, we have $P_1^\pi = f(\emptyset; T)$, for which we can express in terms of another function g by simply defining $g(\emptyset; T) = f(\emptyset; T)$. Suppose the claim is true for $t - 1$; that is, we have constructed a function g such that $P_s^\pi = g(X_1, X_2, \dots, X_s; T)$ for every $s \in \{1, \dots, t - 1\}$. Then we can extend the definition g to the t th period offered price P_t^π by defining

$$\begin{aligned} g(X_1, X_2, \dots, X_{t-1}; T) \\ = f(g(\emptyset; T), X_1, g(X_1; T), \dots, g(X_1, \dots, X_{t-2}; T), X_{t-1}; T); \end{aligned}$$

we see that $P_t^\pi = g(X_1, X_2, \dots, X_{t-1}; T)$. This completes the proof of the claim, hence justifying the definition by (1).

Endnotes

¹ We use linear demand functions to approximate local price elasticity, but our method can also be adapted for other forms of parametric demand families such as Cobb–Douglas functions, as long as the demand functions can be specified by finitely many parameters.

² In our pricing algorithm, we can easily include merchant's revenue share by redefining the optimal price as $p_i^* \in \arg \max_{p \in \mathcal{P}} (p - c)d_i(p)$, where c is the merchant's fixed revenue split. All the results in Section 4 go through with this modification.

References

- Aviv Y, Vulcano G (2012) Dynamic list pricing. Özer Ö, Phillips R, eds. *The Oxford Handbook of Pricing Management* (Oxford University Press, Oxford, UK), 522–584.
- Besbes O, Zeevi A (2009) Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Oper. Res.* 57(6):1407–1420.
- Besbes O, Zeevi A (2015) On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Sci.* 61(4):723–739.
- Bitran GR, Mondschein SV (1997) Periodic pricing of seasonal products in retailing. *Management Sci.* 43(1):64–79.
- Boyacı T, Özer Ö (2010) Information acquisition for capacity planning via pricing and advance selling: When to stop and act? *Oper. Res.* 58(5):1328–1349.
- Broder J (2011) Online algorithms for revenue management. PhD thesis, Cornell University, Ithaca, NY.
- Broder J, Rusmevichientong P (2012) Dynamic pricing under a general parametric choice model. *Oper. Res.* 60(4):965–980.
- Caro F, Gallien J (2012) Clearance pricing optimization for a fast-fashion retailer. *Oper. Res.* 60(6):1404–1422.

- Chen Q, Jasin S, Duenyas I (2015) Real-time dynamic pricing with minimal and flexible price adjustment. *Management Sci.* 62(8):2437–2455.
- den Boer AV (2014) Dynamic pricing with multiple products and partially specified demand distribution. *Math. Oper. Res.* 39(3):863–888.
- den Boer AV (2015) Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys Oper. Res. Management Sci.* 20(1):1–18.
- den Boer A, Zwart B (2014) Simultaneously learning and optimizing using controlled variance pricing. *Management Sci.* 60(3):770–783.
- Feng Y, Gallego G (1995) Optimal starting times for end-of-season sales and optimal stopping times for promotional fares. *Management Sci.* 41(8):1371–1391.
- Harrison J, Keskin N, Zeevi A (2012) Bayesian dynamic pricing policies: Learning and earning under a binary prior distribution. *Management Sci.* 58(3):570–586.
- Keskin NB, Zeevi A (2014) Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies. *Oper. Res.* 62(5):1142–1167.
- Kleinberg R, Leighton T (2003) The value of knowing a demand curve: Bounds on regret for online posted-price auctions. *Proc. 44th Annual IEEE Sympos. Foundations Comput. Sci.* (IEEE Computer Society, Washington, DC), 594–605.
- Mersereau AJ, Rusmevichientong P, Tsitsiklis JN (2009) A structured multiarmed bandit problem and the greedy policy. *IEEE Trans. Automatic Control* 54(12):2787–2802.
- Netessine S (2006) Dynamic pricing of inventory/capacity with infrequent price changes. *Eur. J. Oper. Res.* 174(1):553–580.
- Rothschild M (1974) A two-armed bandit theory of market pricing. *J. Econom. Theory* 9(2):185–202.
- Rusmevichientong P, Tsitsiklis JN (2010) Linearly parameterized bandits. *Math. Oper. Res.* 35(2):395–411.
- Wang Z, Deng S, Ye Y (2014) Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. *Oper. Res.* 62(2):318–331.
- Zbaracki MJ, Ritson M, Levy D, Dutta S, Bergen M (2004) Managerial and customer costs of price adjustment: Direct evidence from industrial markets. *Rev. Econom. Statist.* 86(2):514–533.

Wang Chi Cheung is a research scientist at the Institute of High Performance Computing, Agency for Science, Research and Technology. His research interests include data-driven optimization problems in revenue management and inventory control models.

David Simchi-Levi is a professor of engineering systems at Massachusetts Institute of Technology. He is an INFORMS and MSOM Fellow. The work described in this paper is part of a larger research project that combines machine learning and stochastic optimization to improve revenue, margins, and market share. In addition to implementation at Groupon, companies such as Rue La La, a U.S. online flash sales retailer, and B2W Digital, a large online retailer in Latin America, have implemented some of his pricing algorithms.

He Wang is an assistant professor at H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology. His research interests include online learning problems in revenue management and supply chain management.