Optimizing pricing strategies through learning the market structure

Abstract

This study explores the integration of market structure learning into pricing strategies to maximize revenue in e-commerce and retail environments. We consider the problem of determining the revenue maximizing price of a single product in a market of heterogeneous consumers segmented by their product valuations; and analyze the pricing strategies for varying levels of prior market structure knowledge. The proposed mechanisms utilize customer preference data, and adopt a two-stage approach: an offline learning stage that focuses on learning market structure, including the number of segments and their sizes and valuations, and an online learning stage that prioritizes revenue maximization and iteratively updates these estimates to refine pricing strategies dynamically. Experimental results demonstrate the effectiveness of these methods in adapting to different levels of market knowledge, revealing the economic value of learning market structure in achieving near-optimal revenues. Our findings suggest that even with limited prior market knowledge, firms can benefit from incremental learning strategies to reach profit levels that are relatively close to those achieved in full-information scenarios.

Keywords: preference learning, pricing, machine learning, revenue management

1 Introduction

Pricing plays a decisive role in the e-commerce and retail sectors. Firms aim to maximize revenue while simultaneously uncovering the key market structure characteristics, such as the number of customer segments, segment sizes, and the willingness-to-pay values within each segment. The effectiveness of a pricing strategy relies on the extent to which these characteristics are observable. However, retail firms often have limited knowledge of the market characteristics.

For instance, pricing becomes particularly challenging when launching a product in a market with an unknown structure – such as introducing a new product – where insufficient knowledge can significantly impact the firm's bottom line, leading to suboptimal outcomes and lost opportunities. For example, Apple's iPhone XR [1] and Samsung's Galaxy S20 series [2] experienced sales figures that fell far below expectations, partly due to high prices. These examples highlight the importance of understanding market structure to design effective pricing strategies.

Advances in information technology and data analytics have enabled retail firms to use transaction-level data to learn more about market structures and adjust price levels over time. Changing the prices frequently becomes less costly: online retailers can update prices quickly, and brick-and-mortar stores achieve the same via digital price tags. This capability helps retail firms optimize pricing strategies as they gather more sales data. By integrating price optimization with learning the market structure, retail firms can develop pricing strategies that elevate their bottom lines.

This study examines the interplay between market structure learning and pricing for a single product. We focus on a setting where a monopolist firm offers a single product with unlimited inventory to a market of heterogeneous consumers based on their willingness-to-pay values. We consider a market structure characterized by the number of segments, the size of each segment, and the product valuation level of customers belonging to existing segments. While the firm does not fully observe the market structure, it could have partial knowledge of the market structure, leading to three distinct scenarios: i) the firm has prior knowledge of the number of segments together with their sizes and learns the product valuation levels of the existing segments, ii) the firm has prior knowledge of the number of segments only, and learns the product valuation levels along with the segment sizes, and iii) the firm has no prior knowledge of the market structure.

This study aims to reveal the economic value of learning market structure characteristics and to examine how varying levels of information affect the balance between learning and optimization and the resulting revenue the firm achieves. To this end, we develop mechanisms that optimize pricing while learning unknown parameters under two different structures: deterministic and probabilistic. Our proposed mechanism comprises two stages: the first stage (i.e., offline learning) prioritizes learning the product valuations of the customer segments and then passes this information to the next stage; the second stage (i.e., online learning) maximizes revenue while updating estimations after each sales transaction. We also compare the proposed mechanism to a benchmark scenario where the firm possesses full market information, thereby demonstrating the value of information at different levels.

The Operations Research/Management Science (OR/MS) literature extensively examines pricing and demand learning. Most studies focus on developing easy-to-implement policies that balance learning and optimization, often demonstrating promising performance in terms of the solution quality. The relevant literature can be broadly categorized into two approaches: parametric and nonparametric methods.

In parametric approaches, the relationship between price and demand is associated with a predefined functional form, and various techniques are utilized to estimate unknown parameters. For example, Bertsimas and Perakis [3], Besbes and Zeevi [4], den Boer and Zwart [5], and Keskin and Zeevi [6] employ traditional methods such as linear regression and maximum likelihood estimation. Within this framework, Bayesian learning approaches have also been investigated. Araman and Caldentey [7] and Farias and Van Roy [8] estimate the unknown parameters utilizing Bayesian methods.

Conversely, nonparametric approaches do not assume an explicit demand function upfront; instead, demand patterns are inferred from data. While [4] and [9] perform worst-case analyses, [10] estimate demand information using regression trees. The

scope of this literature stream has been expanded through studies that incorporate various dimensions, including seasonal patterns [11], reference effects [12], a changing market environment [13], time-varying willingness-to-pay distribution [14], unknown transition probabilities [15], a non-stationary environment [16], a multi-product setting [17], and diffusion models [18].

A common theme across these studies is the trade-off between learning and optimization: controlled price variations are necessary to maintain the quality of future estimates. Most of these works focus on aggregate demand learning rather than learning from individual binary preference data (i.e., whether an individual customer buys the product or not). While some recent studies, such as [14] and [15], focus on customer preference learning in the context of a time-varying willingness-to-pay distribution and quality-differentiated products, respectively, they do not infer market segmentation structures. Our study builds on this stream of research by using preference learning to uncover heterogeneous market segmentation structures.

Despite the rapid growth of this literature (see [19, 20], for comprehensive reviews), the pricing and learning in markets with heterogeneous customer segments have remained underexplored. Existing studies often assume a predefined market structure. However, learning multi-segment market structures is of utmost importance for firms that serve diverse customer bases with varying willingness-to-pay values. Our work explicitly accommodates a multi-segment market structure. In this study, we consider a market composed of multiple customer segments with heterogeneous valuation levels. Accordingly, our contributions are twofold: i) we use customer preference data to infer the underlying market segmentation structure (i.e., the number of segments, segment fractions, and willingness-to-pay levels), and ii) we quantify the economic value of learning the market structure knowledge on revenue performance.

The remainder of the paper is structured as follows. Section 2 formalizes the problem setting, outlining the firm's decision process and the customer segmentation process. Section 3 presents the proposed learning methodology, elaborating on how customer preference data is used to infer the market segmentation structure and refine pricing strategies. In Section 4, through numerical experiments, we demonstrate the economic values of learning market structure and its impact on revenue performance. Section 5 concludes with a summary of managerial insights and potential avenues for future research. Additional details of some algorithms and extended version of our experimental results are provided in the Appendix.

2 Problem Definition

Our problem consists of two decision-makers: a seller and an individual customer from an infinite pool. The dynamic between the seller and the customer works as shown in Figure 1. The seller determines a price for the product and offers this price to the arriving customer. Each customer has a *valuation* for the product which is the highest price that s/he is willing to pay for this product. The customer buys the product if the valuation is at least the offered price; otherwise, s/he leaves without a purchase. Customers are grouped into distinct segments based on their purchasing preferences. Those within the same segment demonstrate homogeneous purchasing profiles, meaning they share

similar willingness-to-pay for the product. Conversely, customers belonging to different segments exhibit heterogeneous preferences. We assume there is a single product and the market consists of n distinct customer segments, where the valuation level of each segment i $(i = 1, \ldots, n)$ for the product is denoted by v_i.

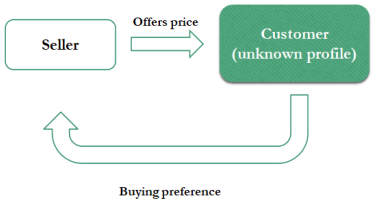


Fig. 1: Information flow between seller and customer

The feedback received by the seller during the interaction is the customer's preference regarding the purchase of the product; whether the customer buys the product or does not buy the product at the given price. Using this preference information, the seller may update the price for the product for the next customer. The seller may have some prior knowledge on the market structure, such as the number of customer segments and their relative fractions. However, while setting the price, the seller does not have information about the specific segment to which an arriving customer belongs to. Hence, the seller cannot customize prices based on customer segmentation. This type of customer-seller interaction is commonly observed by in the e-commerce industry [21, 22].

We assume that the market consists of n segments, v_i is the valuation for customers in segment i, and δ_i is the fraction of customers in the market that belong to segment i, for $i = 1, \ldots, n$. Without loss of generality, we assume that the segments are in ascending order according to their v_i values, i.e. $v_1 < v_2 < ... < v_n$. The goal is to learn the price that maximizes the seller's revenue.

The data collected through the customer-seller interaction, which we refer to as the *preference data set* consists of T data points corresponding to T customers, and two columns $\{(x^t, y^t)\}$, for $t = 1, 2, .., T$. For each customer t, the offered price x^t is within the range $(0, 100)$, while the buying preferences y^t take binary values (1 if customer t buys the product at a price of x^t, and 0 otherwise).

We divide the problem into two stages. The first stage, referred to as *Offline Learning*, focuses on estimating the parameters of the market structure. During this phase, prices are uniformly offered within a range of 0 and 100, $x^t \sim U(0, 100)$. We refer to this stage as *offline* because learning occurs only once, at the end, using the complete dataset. The second stage, referred to as *Online Learning*, aims to maximize the seller's revenue. In this stage, the prices offered to customers are dynamically updated based on the current estimates of the market structure. This stage is termed *online* because the learning algorithm iteratively updates the market structure parameters following each customer interaction.

We examine the seller's pricing problem under varying levels of prior knowledge about the market structure: (i) complete knowledge of the number of segments and their relative sizes (best-case scenario), (ii) knowledge only of the number of segments, and (iii) no prior market information (worst-case scenario). This study aims to establish a balance between offline and online learning stages and quantify the revenue loss associated with each level of incomplete market knowledge.

3 Methodology

3.1 Offline Learning

The primary objective of the offline learning stage, as previously stated, is to learn the market structure, namely the parameters n, v_i, and δ_i by utilizing the given preference dataset. This stage can be regarded as a warm-up stage where the estimated parameters will be passed to the next stage (online learning) for reward (revenue) maximization.

The learning methods proposed for offline learning are based on learning the customers' valuations through the expected probability of purchase function shown in Figure 2. In this graph, the x-axis represents offered prices and valuations, while the y-axis represents the expected probability of purchase for the given prices. For example, prices below v_1 are below the valuation for all customer segments, therefore the expected probability of purchase is 1 (all customers purchase at this price); prices in the range $(v_1, v_2]$ exceeds the valuation of customers in the first segment, however fall below the valuations for all other segments, therefore the expected probability of purchase is $1 - \delta_1$ (all customers except segment 1 purchase at this price). Due to this structure, the expected probability of purchase is a non-increasing step-function of the price.

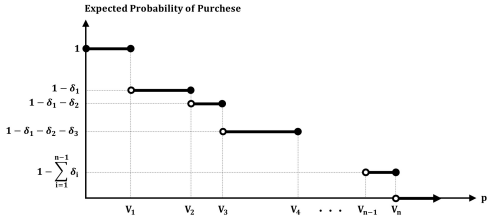


Fig. 2: Expected purchase probability as a function of offered prices for deterministic product valuations

In such a market structure, the revenue maximizing price, P^*, has to be one of the break points; i.e. the valuations. In other words, if you set a price where all segments would purchase, the reward maximizing option is v_1; if you set a price where all segments except segment 1 would purchase, the reward maximizing option is v_2 etc. This formula for P^*, provided in Section 3.2.1, requires three pieces of information: 1) the number of segments n, 2) segment fractions δ_i, and 3) valuations v_i. All of

these three pieces of information can be extracted from Figure 2. In the case of perfect information, all necessary market parameters are given and the reward maximizing price can be correctly calculated. In the absence of these pieces of information, our proposed algorithms compute accurate estimations to be used in reward maximizing price calculations.

We also study market structures where the valuations are not necessarily deterministic, but rather follow a random distribution for each segment, which we refer to as *probabilistic valuations*. In such markets, we lose the nice step structure in Figure 2. The resulting structure depends on the distribution and the interval of uncertainty.

One example of such uncertain intervals is the uniform distribution, where customers' valuations in segment i follow $v_i \sim U(l_i, u_i)$. For instance, if prices below l_1 are offered to customers, the expected probability of purchase is 1, as all customers would make a purchase. For prices in the range $(u_1, l_2]$, the expected probability of purchase becomes $1 - \delta_1$ since those prices exceed the valuations of all customers in segment 1. For prices within the interval $[l_1, u_1]$, the expected probability of purchase decreases uniformly from 1 to $1 - \delta_1$ in this model, resulting in an expected purchase probability function that exhibits a gradual decline, as shown in Figure 3.

As illustrated in the figure, unlike the deterministic valuations, where purchase probabilities show sharp declines, the uniform probabilistic valuations model results in a gradual decrease in purchase probabilities.

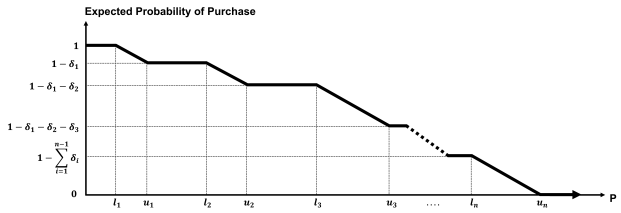


Fig. 3: Expected purchase probability as a function of offered prices for probabilistic product valuations

In the subsequent sections we investigate the problem under various scenarios, each exploring different levels of market structure knowledge. The methodology we follow in the offline phase is identical for deterministic and probabilistic valuations. The performance of this approach, i.e. utilizing algorithms tailored for deterministic valuations to probabilistic valuations, is analyzed in Section 4.

3.1.1 Scenario 1

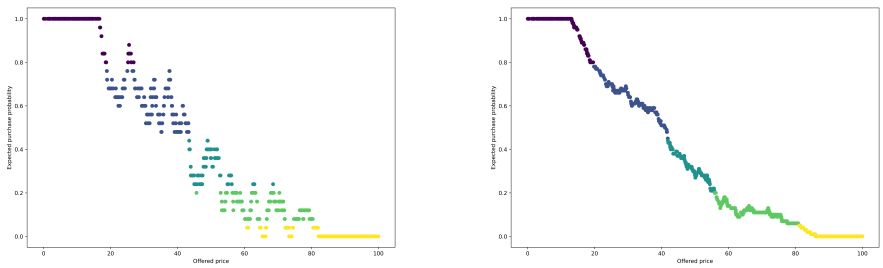
Scenario 1 assumes the highest level of prior knowledge: the number of segments (n), segment fractions (δ_i), and their order are known. Here, the ordering refers to the correspondence between each δ_i value and its respective segment, i.e., we can associate the lowest paying segment with δ_1 and highest paying segment with δ_n etc. In this

scenario only the product valuations are unknown, and we examine the problem of learning customers' valuations.

The main goal is learning customer valuations (v_is), which are not necessarily integer values. For this purpose, we start with offering T customers, randomly selected from the targeted market structure, the product at a price $x^t \sim U(0, 100)$ from continuous uniform distribution; and collect their preference information y^t. This collection constitutes our preference data set.

The first challenge is to convert the preference data with the binary structure to the expected function format in Figure 2. In settings where the suggested price options are discrete, the expected purchase probability of a price can be estimated as the average of the y values associated with the same price x, but this approach loses its validity when the suggested prices are a continuous set, as it is the case in this scenario.

In order to overcome this challange, k-Nearest Neighbors (kNN) algorithm is applied [23]. The algorithm uses the k price values closest to x for each x-value and assigns the average of the y-values corresponding to those prices as the estimate for expected purchase probability of x, which is then denoted by $\hat{y}$. Although this method works well, its performance is highly dependent on the hyperparameter k. Figure 4 shows the $(x^t, \hat{y}^t)$ values obtained when kNN is applied with different k values to the preference dataset consisting of $T = 1000$ data points in a market with $n = 4$ segments. As can be seen in Figure 4(a), choosing a too low value of k increases the fluctuation in the estimated probability of purchase values while choosing a too high value of k as shown in Figure 4(b) results in excessive smoothing of the function and losing the step-structure.

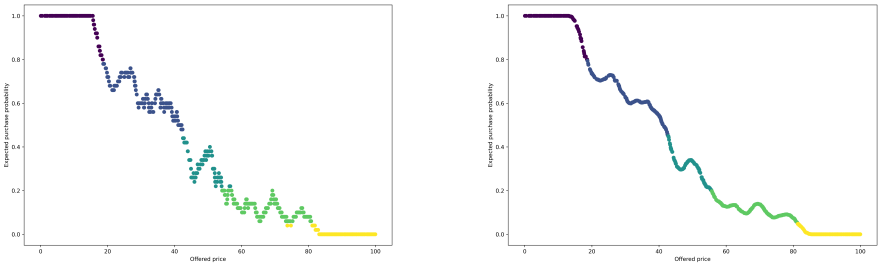


(a) $T = 1000, k = 25$ (b) $T = 1000, k = 100$

Fig. 4: Estimates of the expected probability of purchase with kNN applied for different k values for deterministic product valuations

Alternatively, the desired shape of the purchase probability function can also be achieved through a multi-step conversion process of the preference dataset. In this iterative conversion approach, the preference dataset undergoes successive conversions. In a two-step conversion process, initially, the $\{(x^t, y^t)\}$ dataset is subjected to conversion using kNN, resulting in $\{(x^t, \hat{y}^t)\}$. Subsequently, a second application of kNN is performed on $\{(x^t, \hat{y}^t)\}$, generating a further converted dataset $\{(x^t, \hat{\hat{y}}^t)\}$. Figure 5 illustrates the effect of dataset conversion by applying the kNN algorithm, executed

once and twice. The dataset size remains constant, as does the value of hyperparameter k in both instances. By applying the conversion twice, a smoother shape of the purchase probability function with a cascading trend is achieved. In our proposed algorithm, we opt to perform the conversion process twice. Consequently, our converted dataset is of the form $\{(x^t, \hat{y}^t)\}$.



(a) $(x^t, \hat{y}^t)$ values obtained as a result of one round of kNN

(b) $(x^t, \hat{y}^t)$ values obtained as a result of two rounds of kNN

Fig. 5: The effect of applying one or two rounds of kNN on expected purchase probability estimates for deterministic product valuations

The next step is clustering the points in the data set such that each cluster corresponds to a step in the expected purchase probability function in Figure 2. We call the height of each cluster (i.e. the y-values) the target value of that cluster. Due to the nature of our problem, the expected purchase probability function in an n-segment market structure will consist of $n+1$ clusters; the target value of the first cluster will be 1 and the target value of the $(n+1)^{\text{th}}$ cluster will be 0. The endpoints of these clusters on the x-axis will be the estimations for the valuations.

A standard clustering algorithm to be applied to a dataset consisting of two-dimensional vectors considers the distances in both coordinates when calculating the distance between the elements. However, in this scenario the target height of each cluster (a function of δ_is) is already known. Hence, the clustering performed on the converted data set $(x, \hat{\hat{y}})$ utilizes only the second dimension $\hat{\hat{y}}$; and the cluster centers in this dimension are fixed at target heights.

More specifically, given the fraction of customers being in segment i as δ_i, $i = 1, 2, ..., n$, the cluster center j is calculated as:

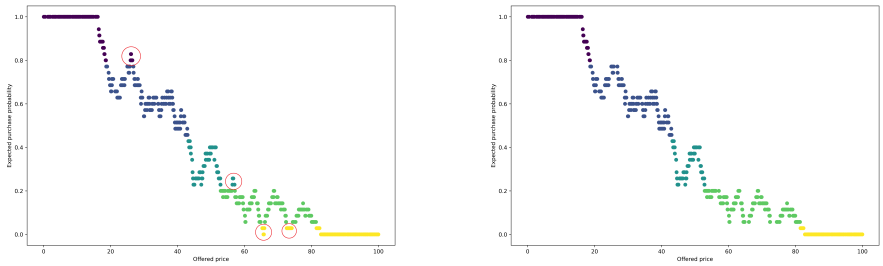
$$p^j = 1 - \sum_{i=1}^{j-1} \delta_i \tag{1}$$

for $j = 1, 2, ..., n+1$. For $t = 1, ..., T$, each point $(x^t, \hat{\hat{y}}^t)$ is assigned to cluster j^t defined as:

$$j^t = \arg \min_{j=1,\ldots,n+1} |\hat{\hat{y}}^t - p^j|. \tag{2}$$

Although clustering is performed using only the second dimension of the data, $\hat{\hat{y}}$, ideally we would like to get clusters consisting of points that are consecutive in the first dimension, x values. As illustrated in Figure 6(a), it has been observed that clustering purely based on $\hat{\hat{y}}$ values may lead to clustering errors where a point that is in the middle of cluster i along the x-axis is assigned to cluster j.

To remedy this situation, we employ a simple IQR method to identify misclassifications through detecting outliers. Namely, after the clusters are determined with (2), for each cluster, its first and third quantiles of x values, Q_1 and Q_3, are calculated, along with $IQR = Q_3 - Q_1$. Any point x^t in that cluster with $x^t > Q_3 + 1.5 \times IQR$ or $x^t < Q_1 - 1.5 \times IQR$ is deemed an outlier [24] and is removed from that cluster. The new clusters obtained after this process can be seen in Figure 6(b).



(a) Before outlier elimination (b) After outlier elimination

Fig. 6: Identifying the wrongly clustered data points (outliers) with the IQR method and eliminating them

Let min_j and max_j be the lower and upper bound of cluster j calculated as the minimum and maximum x^t values among points assigned to cluster j respectively. The unknown valuations are estimated as the average of the consecutive upper and lower bounds. Namely, for $i = 1, 2, ..., n$

$$\hat{v}_i = \frac{max_i + min_{i+1}}{2}. \tag{3}$$

The proposed algorithm explained in this section can be summarized as follows:

Algorithm 1 Learning Valuations

1: Generate T continuous U(0,100) as offered prices (x^t) and collect preference data (y^t)
2: Convert the preference dataset twice using kNN:
3: $\quad \{(x^t, y^t)\} \rightarrow \{(x^t, \hat{y}^t)\}$
4: $\quad \{(x^t, \hat{y}^t)\} \rightarrow \{(x^t, \hat{\hat{y}}^t)\}$
5: Cluster the data points $\{(x^t, \hat{\hat{y}}^t)\}$
6: Eliminate Outliers
7: Estimate the customers' valuations

3.1.2 Scenario 2

Scenario 2 considers a lower level of prior knowledge regarding the market structure. While the number of segments (n) is known, the segment fractions (δ_i) and the customers' valuations (v_i) are both unknown. Instead of learning both unknowns simultaneously, we have observed that a sequential search yields more accurate results. Therefore, our approach is to first learn the y-axis of the estimated probability of purchase function and estimate the segment fractions; and then to apply Algorithm 1 using these segment fraction estimations.

Since the algorithm won't initially be learning the valuations, precision on the x-axis is not the primary concern while generating the offered prices for the offline learning preference data set. Unlike Algorithm 1 where offered prices are generated from continuous $U(0, 100)$ distribution; Algorithm 2 for Scenario 2 uses discrete price values. The seller offers m distinct discrete price values which are in the range of $(0, 100)$ to the customers, with each price being offered m' times, to $T' = m \times m'$ customers selected randomly from the target market structure. The offered prices are selected such that they divide the range $(0, 100)$ into m equal regions (e.g. for $m = 20$, the prices offered are $5, 10, 15, \ldots, 95, 100$).

With this new structure of the preference data set (x^t, y^t), the conversion to the estimated probability of purchase function through kNN is more intuitive. When k is carefully selected as $k = m'$, $\hat{y}^t$ becomes the average preference (y^t) of customers that were offered the same price (x^t). Furthermore, applying kNN twice is meaningless, as for $k = m'$, $\hat{y}^t = \hat{\hat{y}}^t$. Figure 7 illustrates the shape of the estimated expected purchase probability function for deterministic valuations in a market with $n = 4$ segments when discrete prices are offered. In contrast to the continuous price scenario shown in Figure 5, the discrete price scenario exhibits a distinct and discrete pattern characterized by sharp declines when transitioning between clusters.

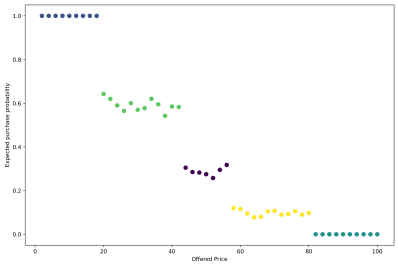


Fig. 7: Estimated expected purchase probability as a function of discrete offered prices for deterministic valuations

Similar to Algorithm 1, the next step is clustering the data points such that each cluster corresponds to a step in the expected purchase probability function. By Scenario 2 assumptions, we know the number of segments n, therefore the number of clusters $n+1$; however we no longer have the information on the target height of each cluster. Therefore, we utilize the k-means clustering algorithm [25]; and set $k = n+1$.

Prior to applying the k-means clustering algorithm, the values along both axes are normalized to fall within the range of $0-1$. This normalization step ensures that the features are on a consistent scale and helps facilitate accurate clustering. Given the cluster centers as the output of k-means clustering algorithm, the second coordinate of cluster center i is used as the estimated purchase probability of cluster i; $\hat{p}^i$.

Using these purchase probability estimations, the segment fractions for $i = 1, \ldots, n$ can be estimated by subtracting the $\hat{p}$ values for two consecutive clusters:

$$\hat{\delta}_i = \hat{p}^i - \hat{p}^{i+1}. \tag{4}$$

The proposed algorithm explained in this section can be summarized as follows:

Algorithm 2 Learning Segment Fractions

1: Generate T' discrete offered prices (x^t) and collect preference data (y^t)
2: Convert the preference dataset using kNN:
3: $\quad \{(x^t, y^t)\} \rightarrow \{(x^t, \hat{y}^t)\}$
4: Cluster the data points $\{(x^t, \hat{y}^t)\}$ using k-means
5: Estimate purchase probability of clusters, $\hat{p}^i$
6: Estimate segment fractions $\hat{\delta}_i$

3.1.3 Scenario 3

Scenario 3 considers the lowest level of prior knowledge. The number of segments (n), segment fractions (δ_i), and customers' valuations (v_i) are all unknown. The problem we examine is the problem of learning the entire market structure. Similar to Scenario 2, the market structure will be learned sequentially; the first phase focuses on estimating the number of segments and segment fraction; and the second phase applies Algorithm 1 using these estimations.

Given a lower and an upper bound on the number of segments, L and U respectively, our proposed learning method is essentially applying Algorithm 2 for all number of segments $n' \in \{L, \ldots, U\}$, and selecting the "best fit". Standard clustering algorithms define error as the sum of individual deviations of data points from the assigned cluster centers; and the main objective is minimizing this error term. Hence, the expectation is that the clustering error decreases as k increases. However, in our case, we are not mainly interested in the deviation in the x-axis (prices) within a cluster. The goal is to have data points with similar expected purchase probabilities to be clustered together. Therefore, after Algorithm 2 is applied for $n' \in \{L, \ldots, U\}$, we measure the fit error, $Err[n']$, as the sum of the absolute deviations between $\hat{y}^t$ and the second coordinate of the assigned cluster center for $t = 1, \ldots, T$. In other words, we calculate the absolute deviation along the y-axis.

The cluster structure we aim to reach is such that as the cluster centers increase along the x-axis, they should decrease along the y-axis. As previously stated, the difference between the y-values of each cluster center $(\hat{p}_i)$, i.e. the height of the clusters, correspond to segment fractions. We would like these differences to be significant, and be above a certain threshold τ. After Algorithm 2 is applied for $n' \in \{L, \ldots, U\}$,

we perform a feasibility check; and define $Feas[n']$=1 if for all $i = 1, \ldots, n'$, we have $\hat{p}^i - \hat{p}^{i+1} \geq \tau$; and $Feas[n']$=0 otherwise. In other words, n' is a feasible fit if it yields a set of clusters whose heights decrease by at least τ at each step, indicating a market where the fraction of each segment is at least τ.

After all $n' \in \{L, \ldots, U\}$ are tested; we estimate n as the minimizer of $Err[n']$ among the values of n' that satisfy $Feas[n'] = 1$:

$$\hat{n} = \arg \min_{n' \in \{L,\ldots,U\}} \{Err[n'] \ : \ Feas[n'] = 1\}.$$

Purchase probability of clusters ($\hat{p}$) and segment fractions ($\hat{\delta}$) using $\hat{n}$ are estimated as explained in Section 3.1.2. The proposed algorithm explained in this section can be summarized as follows:

Algorithm 3 Learning Number of Segments and Segment Fractions

1: Generate T' discrete offered prices (x^t) and collect preference data (y^t)
2: Convert the preference dataset using kNN:
3: $\quad \{(x^t, y^t)\} \rightarrow \{(x^t, \hat{y}^t)\}$
4: **for** $n' = L, \ldots, U$ **do**
5: $\quad$ Cluster the data points $\{(x^t, \hat{y}^t)\}$ using k-means with $k = n' + 1$
6: $\quad$ Calculate fit error: $Err[n']$ = deviation on y-axis
7: $\quad$ Run feasibility check: $Feas[n'] = 1$ if feasibility check is satisfied, 0 otherwise
8: **end for**
9: Estimate number of segments $\hat{n}$
10: Estimate purchase probability of clusters, $\hat{p}^i$, using $\hat{n}$
11: Estimate segment fractions $\hat{\delta}_i$, using $\hat{n}$

3.2 Online Learning

Online learning stage differentiates itself from offline learning stage in two main areas. First, the estimations for valuations are updated after each customer interaction rather than as a batch at the end; hence the name *online* learning. Second, the objective shifts from learning the market structure to reward maximization.

Online learning stage is initialized with the market parameters estimated in offline learning stage. Given a market structure, the calculation of reward maximizing prices differs based on whether the valuations are deterministic or probabilistic. Therefore, in this section we propose two different approaches based on the nature of valuations.

3.2.1 Deterministic Valuations

As previously mentioned, in a market with deterministic valuations, the reward-maximizing price, P^*, has to be one of the break points illustrated in Figure 2; i.e. the valuations. For instance, if you set a price where all segments would purchase, the price with the highest reward is v_1; if you set a price where all segments except segment 1 would purchase, the price with the highest reward is v_2 etc. In other words, if

the offered price is v_i, then all customers from segments $i, \ldots, n$ would purchase, making the probability of purchase $\sum_{j=i}^{n} \delta_j = 1 - \sum_{j=1}^{i-1} \delta_j$. Among n possible candidates, the reward maximizing break point i^* is calculated as

$$i^* = \arg \max_{i=1,\ldots,n} \left\{ \left(1 - \sum_{j=1}^{i-1} \delta_j \right) \times v_i \right\} \tag{5}$$

and the reward maximizing price $P^* = v_{i^*}$. The optimal expected reward corresponding to this reward maximizing price is

$$R^* = \left(1 - \sum_{j=1}^{i^*-1} \delta_j \right) \times P^*. \tag{6}$$

At each iteration, the estimation for the reward maximizing break point, $\hat{i^*}$ is calculated with (5) using $\hat{v}_i$ instead of v_i, $\hat{\delta}_i$ instead of δ_i, and $\hat{n}$ instead of n when the said market information is unknown.

An important issue that is encountered while selecting the reward maximizing price is the asymmetric penalty associated with estimation errors. Namely, overestimating a valuation may cost significantly more than underestimating, even by the same error margin. This is due to the fact that by offering a price more than the valuation, we lose a segment, which directly impacts the reward collected. Consider the following example: When the product valuations (v_i) are [18, 43, 56, 81], respectively, and the segment fractions (δ_i) are [0.4, 0.3, 0.2, 0.1], the price that maximizes the reward is $P^* = 43$ (by (5)). When this price is offered, the expected reward is $(43) \times (0.6) = 25.8$ (by (6)), where all customers except those in the first segment buy the product. If this valuation is underestimated as $\hat{v}_2 = 42$, and the price offered to the customer is 42, then using the true valuations still all customers except the customers in the first segment purchase the product, and the expected reward is $(42) \times (0.6) = 25.2$. If this valuation is overestimated as $\hat{v}_2 = 44$ and the price offered to the customer is 44, then customers in both the first and second segments leave without a purchase, as the price 44 exceeds both $v_1 = 18$ and $v_2 = 43$, and the expected reward becomes $(44) \times (0.3) = 13.2$. As observed, the estimation error amount in these two cases are identical, but the consequences of these errors differ significantly leading to substantial reward loss.

To address this asymmetry and mitigate potential reward losses, we propose deducting a variable value of α (which can be considered as a buffer) from the determined offered price and then offering this new price to the incoming customer, in order to avoid overpricing. Throughout the online learning iterations, as the estimations improve, the length of the buffer should decrease; and ideally as $v_{\hat{i^*}} \to v_{i^*}$, then $\alpha \to 0$. In order to determine when to decrease α, we detect a convergence in $v_{\hat{i^*}}$ using moving average. Taking the moving average of $v_{\hat{i^*}}$ over a window size of ω, if the percentage difference between two consecutive moving averages are below a threshold τ for γ consecutive iterations, we assume a *steady trend* and decrease α by a rate of η.

The proposed algorithm in this section can be summarized as follows:

Algorithm 4 Reward Maximization in Deterministic Setting

for $j = 1, \ldots, T_{online}$ **do**
  Calculate $\hat{i^*}$
  Determine the offered price: $x = \hat{v}_{\hat{i^*}} - \alpha$
  Offer price x and collect customer's buying preference y
  Add the new data point (x, y) to the preference data set
  Update $\hat{v}_i$s using the updated data set and Algorithm 1 (Lines 2-7)
  Calculate the moving average of $\hat{v}_i$s and check for *steady trend*
  if *steady trend* **then**
    Update $\alpha \leftarrow \eta\alpha$
  end if
end for

3.2.2 Probabilistic Valuations

When the valuations are probabilistic, then there is no longer a guarantee that the reward maximizing price occurs at one of the break points. Consider the following example where the valuations for each segment i follow a uniform distribution $v_i \sim U(l_i, u_i)$, with non-overlapping intervals. For any two prices x' and x'' in the interval $(u_i, l_{i+1}]$ for a fixed $i = 1, \ldots, n-1$, the probability of purchase for x' and x'' are identical, $(1 - \sum_{j=1}^{i} \delta_j)$. Therefore the reward maximizing price in the interval $(u_i, l_{i+1}]$ is the highest possible value, l_{i+1}. Since the intervals $(u_i, l_{i+1}]$ are dominated by l_{i+1}, we can safely assume that the overall reward maximizing price belongs to one of the intervals $[l_i, u_i]$ for $i = 1, \ldots, n$. Figure 8 is the magnified version of Figure 3, focusing on one such interval: the break point of segment i.

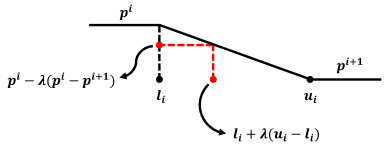


Fig. 8: Expected purchase probability as a function of offered prices for probabilistic product valuations at the break point of segment i

Each price in this interval can be represented as $l_i + \lambda_i(u_i - l_i)$ where $\lambda_i \in [0, 1]$ for $i = 1, \ldots, n$, with the corresponding purchase probability as $p^i - \lambda_i(p^i - p^{i+1})$, and the expected reward as

$$(l_i + \lambda_i(u_i - l_i)) \times (p^i - \lambda_i(p^i - p^{i+1})) \tag{7}$$

. The reward maximizing price in each interval is no longer guaranteed to be one of the break points l_i or u_i. The maximizer of the term (7) can be calculated as

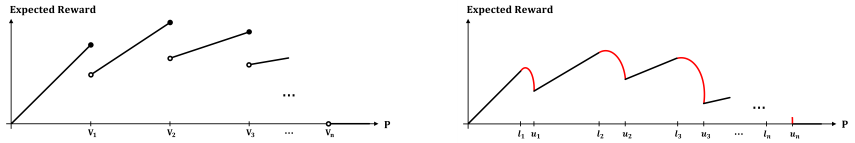
$$\lambda_i^* = \frac{p^i(u_i - l_i) - l_i(p^i - p^{i+1})}{2(p^i - p^{i+1})(u_i - l_i)}. \tag{8}$$

Using λ_i^*s for each interval, the optimal price in interval i is $\bar{v}_i = l_i + \lambda_i^*(u_i - l_i)$, and the corresponding purchase probability is $\bar{p}^i = p^i - \lambda_i^*(p^i - p^{i+1})$. The interval with the maximum reward is then calculated as

$$i^* = \arg \max_{i=1,\dots,n} \left\{ \bar{v}_i \times \bar{p}^i \right\}. \tag{9}$$

Since online learning stage still has ongoing interaction with the customers, learning and improving estimations are expected to continue in this stage. In other words, the data collected by offering the reward maximizing price should contribute to improving the estimation for the reward maximizing price. However, as seen in equations (8) and (9), the calculation of λ_i^* and the optimal interval utilizes the information on the endpoints of the randomness interval, l_i and u_i; and in cases where the optimal price is in the interior of the interval, the data collected by offering a price in the interior does not help with improving the estimations of the end points. Therefore, this strategy of calculating the optimal price converges prematurely.

Instead of estimating market parameters using the preference data, we take a different approach and utilize reward data. Given a price x^t and customer preference y^t, the reward collected in this transaction, r^t, is x^t if $y^t = 1$, and 0 if $y^t = 0$. Depending on the distribution of the valuations, the expected reward function may have a structure as illustrated in Figure 9a for deterministic and Figure 9b for non-overlapping uniform valuations.



(a) Deterministic Valuations (b) Probabilistic Valuations

Fig. 9: Expected reward as a function of offered prices

We propose using a method that works on (x^t, r^t) reward data set instead of (x^t, y^t) preference data set; converts (x^t, r^t) to the expected reward function $(x^t, \hat{r}^t)$ using kNN similar to Algorithm 1; and searches for the price with the maximum expected reward.

Following the data collection in the offline stage, the boundary of each interval i (l_i, u_i) is estimated using one of the methods described in Appendix A to get $(\hat{l}_i, \hat{u}_i)$. The optimal interval $\hat{i^*}$ is selected with (8) and (9) using these estimated boundaries. The goal is to conduct an ϵ-greedy search within the interval $[\hat{l}_{\hat{i^*}}, \hat{u}_{\hat{i^*}}]$. Namely, at each

iteration, with probability ϵ we offer the reward maximizing price x^{t^*}, where

$$t^* = \arg\max_t \{\hat{r}^t \ : \ x^t \in [\hat{l}_{\hat{i}^*}, \hat{u}_{\hat{i}^*}]\} \tag{10}$$

and with probability $1-\epsilon$ we offer a random price $\sim U(\hat{l}_{\hat{i}^*}, \hat{u}_{\hat{i}^*})$. We have also observed the benefits of slightly stretching these bounds by a factor of ϕ; so after calculating $\hat{i}^*$, we update the bounds of the optimal interval as $\hat{l}_{\hat{i}^*} \leftarrow (1-\phi)\hat{l}_{\hat{i}^*}$ and $\hat{u}_{\hat{i}^*} \leftarrow (1+\phi)\hat{u}_{\hat{i}^*}$.

Naturally, the values of ϵ converges to 1 as the estimation accuracy increases with learning. As opposed to the previous approach, the offered prices in this approach indeed improve the estimation of the reward maximizing price. The methodology described uses uniformly random valuations with non-overlapping intervals as an example; however it can be generalized for other distributions with nonoverlapping intervals.

The proposed algorithm can be summarized as follows:

Algorithm 5 Reward Maximization in Probabilistic Setting

1: Estimate valuation bounds $(\hat{l}_i, \hat{u}_i)$ using Algorithm 6, 7, or 8 *(see Appendix A)*
2: Estimate the optimal interval $\hat{i}^*$
3: Widen search interval $\hat{l}_{\hat{i}^*} \leftarrow (1-\phi)\hat{l}_{\hat{i}^*}$ and $\hat{u}_{\hat{i}^*} \leftarrow (1+\phi)\hat{u}_{\hat{i}^*}$
4: **for** $j = 1, \ldots, T_{online}$ **do**
5: Determine the offered price (x):
6: *With probability ϵ, $x =$ reward maximizing price
7: *With probability $1-\epsilon$, $x \sim U(\hat{l}_{\hat{i}^*}, \hat{u}_{\hat{i}^*})$
8: Collect reward r
9: Add the new data point (x, r) to the reward data set
10: Recalculate the $\hat{r}$ values in the updated data set using kNN
11: Update ϵ using Subroutine 1 or 2 *(see Appendix B)*
12: **end for**

4 Experimental Results

4.1 Experimental Setting

In this study, we present the results of a market with $n = 4$ segments, and segment fractions $\delta = [0.4, 0.3, 0.2, 0.1]$. For deterministic markets, we take $v = [18, 43, 56, 81]$ as product valuations; and for probabilistic markets $v = [U(17.1, 18.9), U(40.85, 45.15), U(53.2, 58.8), U(76.95, 85.05)]$ as respective distributions of valuations. Notice that the distributions in probabilistic valuations are selected to be centered at the given deterministic valuations, with 5% deviation on both sides. Furthermore, for probabilistic valuations, the optimal price in each interval is calculated to be $\bar{v} = [17.1, 40.85, 53.2, 76.95]$.

The goal is designing algorithms that eventually reach the reward maximizing prices in these markets. In the deterministic market, the optimal decision is to forgo

segment 1 and set the price to $P^* = 43$ to reach the maximum expected reward $R^* = 25.8$ per customer. The optimal decision in the probabilistic market also forgoes the first segment, has $P^* = 40.85$ and $R^* = 24.51$ per customer. In both of these markets the optimal break point is $i^* = 2$.

For different settings and algorithms, each experiment is replicated $R = 1000$ times to provide stability in the results and reduce the variance due to randomness. A summary of these replications (such as mean and median) is reported in the remainder of this section. Extended results of these experiments can be found in Appendix C.

4.2 Offline Learning Experiments

The main goal in this section is parameter tuning and determining the minimum number of customer interactions T that yields satisfactory estimates of the unknown market parameters before proceeding to online learning stage. In Section 3.1, we introduced three algorithms for three levels of prior market information. We will investigate each of these three settings in the following subsections.

4.2.1 Scenario 1

In Scenario 1 the only missing information about the market structure are the valuations, and these values are estimated using Algorithm 1. Algorithm 1 has two hyperparameters to be tuned: the k value in kNN, and the number of data points (customer interactions) T; and we study the impact of the values assigned to these parameters on the performance of Algorithm 1. In this setting, we use the following two metrics to measure performance.

The first metric considers how close our estimated valuations are to the true valuations. Let $\hat{v}_i^r$ be the estimation for v_i in replication r. The estimation error of replication r is defined as

$$\texttt{vError}^{\texttt{r}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{v}_i^r - v_i|}{v_i} \times 100. \tag{11}$$

We report the estimation error `vError`, as the median of $\texttt{vError}^{\texttt{r}}$ over all replications.

Since the proposed online algorithm Algorithm 4 calculates the optimal breakpoint using the estimations output by Algorithm 1, and conducts a local search in that neighborhood, it is also important to check whether the results of Algorithm 1 direct the online algorithm to the correct neighborhood. As previously calculated true optimal breakpoint is $i^* = 2$. Let $\hat{i}^{*r}$ be the estimated optimal breakpoint in replication r, calculated using (5) with known and estimated market information. The second metric `TrueBP`, calculated as (12), describes the percentage of times the estimations obtained by Algorithm 1 lead to the online algorithm choosing i^* as the optimal breakpoint.

$$\texttt{TrueBP} = \frac{1}{R}\sum_{r=1}^{R}\mathbb{1}_{\{\hat{i}^{*r}=i^*\}} \times 100. \tag{12}$$

Table 1 displays the results for various T and k values. Notice that the value of k is in practice dependent on the total number of data points. Therefore we chose not

to have a fixed set of values tested for k, but rather have them adaptable based on T. Along with `vError` and `TrueBP` explained previously, the columns $\hat{v}_i$ in Table 1 correspond to the median of $\hat{v}_i^r$ among r estimates.

Table 1: Algorithm 1 performance in Scenario 1 deterministic valuations

		$\hat{v}_1$	$\hat{v}_2$	$\hat{v}_3$	$\hat{v}_4$	`vError`	`TrueBP`
$T = 50$	$k = 11$	21.17	41.55	57.01	74.39	15.85	63.6
$T = 100$	$k = 15$	18.59	42.1	56.45	77.11	12.1	82.2
$T = 250$	$k = 30$	18.04	42.92	55.95	79.46	7.05	96.6
$T = 500$	$k = 35$	18.09	42.94	55.88	80.1	4.11	99.7
$T = 1000$	$k = 45$	18.02	43.03	55.83	80.72	2.3	99.9

While selecting the hyperparemeters for the algorithms, in general, upon observing experimental results, we assume 7% is a sufficient and feasible upperbound for estimation error percentages, and 95% is a sufficient and feasible lower bound on selecting the correct interval percentages. Therefore, for each algorithm, we select the minimum number of data points required to remain in these bounds.

Table 1 demonstrates the significant impact of the number of data points on the estimation error `vError`; as the error decreases with more data. There is no direct relationship between k and the error. It is observed that the error reaches a minimum with increasing k, and then starts increasing. This phenomenon was also explained in Section 3.1.1. Overall, $T = 500$ customer interactions are sufficient for an estimation error of 4.11% when $k = 35$. With these hyperparameters, the estimations of Algorithm 1 point to the correct breakpoint with probability 0.997; justifying the local search in the online learning stage.

In the probabilistic setting, for the estimation errors we use a similar formulation to (11). However, since exact v_i values fo not exist, we consider the deviation from the optimal valuation in each interval $\bar{v}_i$, which are used for determining the optimal breakpoints. Using $\hat{\bar{v}}_i^r$ as the estimations of $\bar{v}_i^r$ in replication r, we calculate

$$\bar{\texttt{v}}\texttt{Error}^{\texttt{r}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{\bar{v}}_i^r - \bar{v}_i|}{\bar{v}_i} \times 100 \tag{13}$$

and report the estimation error $\bar{\texttt{v}}\texttt{Error}$, as the median of $\bar{\texttt{v}}\texttt{Error}^{\texttt{r}}$ over all replications.

Similarly, the proposed online algorithm Algorithm 5 is restricted to search the interval $[\hat{l}_{\hat{i}^*}, \hat{u}_{\hat{i}^*}]$, we also measure whether the optimal price $P^* = 40.85$ falls within the given interval. Let I^{*r} be the search interval in replication r with widening scale $\phi = 0$ as described in Section 3.2.2, the metric `WithinInt` is calculated as

$$\texttt{WithinInt} = \frac{1}{R}\sum_{r=1}^{R}\mathbb{1}_{\{P^* \in I^{*r}\}} \times 100. \tag{14}$$

We have also experimented for $\phi = 0.05$, and for the widened intervals I_w^{*r}, `WithinIntWide` is reported as

$$\texttt{WithinIntWide} = \frac{1}{R}\sum_{r=1}^{R} \mathbb{1}_{\{P^* \in I_w^{*r}\}} \times 100. \tag{15}$$

Table 2 displays the results for the same set of T and k values. Along with $\bar{\texttt{v}}\texttt{Error}$, `WithinInt` and `WithinIntWide`, the columns $\hat{\bar{v}}_i$ in Table 1 correspond to the median of $\hat{\bar{v}}_i^r$ among r estimates.

Table 2: Algorithm 1 performance in Scenario 1 probabilistic valuations

		$\hat{\bar{v}}_1$	$\hat{\bar{v}}_2$	$\hat{\bar{v}}_3$	$\hat{\bar{v}}_4$	$\bar{\texttt{v}}\texttt{Error}$	`WithinInt`	`WithinIntWide`
$T = 50$	$k = 5$	28.23	37.86	46.74	59.22	30.23	50.2	58.9
$T = 100$	$k = 5$	24.35	38.41	46.29	58.45	24.39	44.4	53.1
$T = 250$	$k = 20$	25.79	36.82	50.1	73.34	20.59	65.2	75
$T = 500$	$k = 35$	22.08	37.55	50.7	75.11	13.09	80.2	91.6
$T = 1000$	$k = 45$	17.41	39.1	51.86	77.03	7.28	81.2	98.8

We see a similar pattern with probabilistic valuations. Naturally, this setting involves a more difficult learning problem; therefore for the same number of data points, the estimation errors are higher than that of deterministic valuations. In this setting, with $T = 1000$ we are able to get sufficiently close estimates. Since fine-tuning of the price estimations is done in the online stage, it is critical that the reward maximizing price falls in the search interval of the online algorithm, i.e. "within interval" probabilities. In that respect, we observe the significant benefit of widening the search interval by $\phi = 0.05$, as with the widened interval, the probabilities increase drastically. For $T = 1000$, we select $k = 45$ as the maximizer of `WithinIntWide`.

4.2.2 Scenario 2

In Scenario 2, the proposed methodology is to initially learn the segment fractions δ_i using Algorithm 2; then to use these estimations to apply Algorithm 1 to find the reward maximizing price. Unlike Algorithm 1 where the prices offered to T customers are selected from $U(0, 100)$; Algorithm 2 offers m distinct prices (dividing the range $[0, 100]$ into m equal intervals) m' times each; and sets $k = m'$, taking the average of the m' replications for each m price value. Therefore, only the impact of m and m' on the performance need to be observed.

We have tested for two different values of m, $m = 20$ and $m = 50$, discretizing $[0, 100]$ into a coarser and a finer grid respectively. For a fair comparison, for each m value, we selected m' values such that the total number of data points required $T = m \times m'$ is constant (i.e. $m = 20$, $m' = 50$ vs $m = 50$, $m' = 20$).

In order to assess the performance of Algorithm 2, we first consider the estimation error of segment fractions δ. Namely, let $\hat{\delta}_i^r$ be the estimation for δ_i in replication r. The estimation error of replication r is defined as

$$\delta\texttt{Error}^{\texttt{r}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{\delta}_i^r - \delta_i|}{\delta_i} \times 100 \tag{16}$$

We report the estimation error $\delta\texttt{Error}$, as the median of $\delta\texttt{Error}^{\texttt{r}}$ over all replications. Table 3 displays the results for the various m and m' values. Along with $\delta\texttt{Error}$, the columns $\hat{\delta}_i$ in Table 3 correspond to the median of $\hat{\delta}_i^r$ among r estimates.

Table 3: Algorithm 2 performance in Scenario 2 deterministic valuations

		$\hat{\delta}_1$	$\hat{\delta}_2$	$\hat{\delta}_3$	$\hat{\delta}_4$	$\delta\texttt{Error}$
$m = 20$	$m' = 100$	0.4	0.301	0.199	0.098	8.58
	$m' = 200$	0.4	0.3	0.201	0.099	5.86
	$m' = 500$	0.4	0.301	0.2	0.1	3.77
$m = 50$	$m' = 40$	0.398	0.303	0.201	0.1	8.89
	$m' = 80$	0.401	0.301	0.199	0.1	6.23
	$m' = 200$	0.399	0.301	0.199	0.1	3.81

As expected, for a given m, as m' (therefore T) increases, the estimation error decreases. It is also observed that for the same number of data points, choosing a coarser grid and more replications per price (i.e. $m = 20$) yields slightly lower estimation error, compared to a finer grid with fewer replications per price (i.e. $m = 50$). For Algorithm 2 in Scenario 2 with deterministic valuations, we set $T = 4000$ data points, where the prices are determined according to $m = 20$ and $m' = 200$, for sufficiently accurate estimates.

Since the ultimate goal is estimating the reward maximizing price, we also study the impact of segment fraction estimations of Algorithm 2 on the performance of Algorithm 1. Table 4 reports the performance metrics of Algorithm 1 for $T = 500$ and $k = 35$; using the estimations of Algorithm 2 for the same m and m' selections.

Table 4: Algorithm 1 performance in Scenario 2 deterministic valuations

		$\hat{v}_1$	$\hat{v}_2$	$\hat{v}_3$	$\hat{v}_4$	$\texttt{vError}$	$\texttt{TrueBP}$
$m = 20$	$m' = 100$	18.12	42.89	55.94	80.26	4.33	98.9
	$m' = 200$	18.11	42.88	56.03	80.14	4.36	99.2
	$m' = 500$	18.05	42.98	56.03	80.02	4.21	99.8
$m = 50$	$m' = 40$	18.12	42.92	56.08	80.24	4.39	98.9
	$m' = 80$	18.08	42.99	56.03	79.96	4.2	99.3
	$m' = 200$	18.14	42.8	55.96	80.46	4.25	99

When Algorithm 2 estimations are used in Algorithm 1; despite relatively lower accuracy of $\hat{\delta}$s, we can observe that the accuracy of $\hat{v}$ estimations are close. Our

selected hyperparemeters $m = 20$ and $m' = 200$ give estimation errors lower than our 7% threshold, while selecting the correct breakpoint with 0.992 probability.

A similar set of experiments is conducted for the probabilistic valuations setting. Table 5 displays the median $\hat{\delta}_i$ estimations and δ`Error` for the same set of m and m' values. Following these results, Table 6 reports the probabilistic setting performance metrics of Algorithm 1 for $T = 1000$ and $k = 45$; using the estimations of Algorithm 2 for the same m and m' selections.

Table 5: Algorithm 2 performance in Scenario 2 probabilistic valuations

		$\hat{\delta}_1$	$\hat{\delta}_2$	$\hat{\delta}_3$	$\hat{\delta}_4$	δ`Error`
$m = 20$	$m' = 100$	0.4	0.317	0.187	0.094	9.42
	$m' = 250$	0.4	0.317	0.189	0.093	6.73
	$m' = 500$	0.4	0.319	0.187	0.092	5.78
$m = 50$	$m' = 40$	0.387	0.3	0.203	0.094	10.29
	$m' = 100$	0.388	0.299	0.199	0.095	6.94
	$m' = 200$	0.387	0.301	0.196	0.094	5.55

Table 6: Algorithm 1 performance in Scenario 2 probabilistic valuations

		$\hat{\bar{v}}_1$	$\hat{\bar{v}}_2$	$\hat{\bar{v}}_3$	$\hat{\bar{v}}_4$	$\bar{\text{v}}$`Error`	`WithinInt`	`WithinIntWide`
$m = 20$	$m' = 100$	17.65	38.51	53.12	53.12	8	78.3	95.6
	$m' = 250$	17.75	38.84	53.01	53.01	7.52	79.4	98.5
	$m' = 500$	17.21	38.72	53.11	53.11	7.22	80.6	98.2
$m = 50$	$m' = 40$	17.05	38.21	51.35	51.35	8.09	80.2	94.8
	$m' = 100$	17.18	38.47	51.83	51.83	8.13	79.2	98
	$m' = 200$	17.33	38.59	51.87	51.87	7.5	82.2	97.6

With probabilistic valuations, we still observe that a coarser grid with more replications give more accurate estimations than a finer grid with fewer replications. We again observe that widening the search interval by $\phi = 0.05$ has a significant impact on including the optimal price in the online stage search interval. In this setting, the minimum number of data points with an estimation error below 7% is $T = 5000$ with $m = 20$ and $m' = 250$. These δ estimations used in Algorithm 1 has a 0.985 probability of having the optimal price in the online stage search interval.

4.2.3 Scenario 3

In Scenario 3, no market information is available apriori; and the proposed methodology is to employ Algorithm 2 search strategy for a range of number of segments

$n \in \{L, \dots, U\}$, and select the minimizer of estimation error among feasible estimations. In our experiments we tested for the range $\{1, \dots, 7\}$. Regarding the performance of Algorithm 3, we first report in Table 7 the frequency of $\hat{n}$ estimations; i.e. among 1000 replications, the probability of $\hat{n}$ being estimated as $i \in \{1, \dots, 7\}$.

Table 7: Algorithm 3 segment estimations in Scenario 3 deterministic valuations

		Frequency of						
		$\hat{n}=1$	$\hat{n}=2$	$\hat{n}=3$	$\hat{n}=4$	$\hat{n}=5$	$\hat{n}=6$	$\hat{n}=7$
$m=20$	$m'=100$	0	0	0.001	0.719	0.255	0.022	0.003
	$m'=250$	0	0	0	0.94	0.06	0	0
	$m'=500$	0	0	0	0.984	0.016	0	0
$m=50$	$m'=40$	0	0	0	0.461	0.396	0.099	0.044
	$m'=100$	0	0	0	0.868	0.126	0.004	0.002
	$m'=200$	0	0	0	0.976	0.024	0	0

We can observe that in a significant majority of the experiments, n is estimated correctly; with accuracy increasing with T. In the case of a faulty estimation, the most probable error is creating one additional cluster and estimating $\hat{n} = 5$. We can observe that the coarser grid choice gives slightly more accurate estimates in this setting as well.

Although estimating n correctly is significant in terms of learning the market structure, what is critical for revenue maximization is to estimate the existing valuations correctly and being led to the correct neighborhood before starting the online phase. Therefore, we also report the estimated reward maximizing price and its distance from the optimal reward maximizing price as performance metrics of Algorithm 1 for $T = 1000$ and $k = 45$. Assuming $\hat{v}^r_{opt}$ to be the optimal price calculated by the market parameter estimations of Algorithm 3 in replication r; the column $\hat{v}_{opt}$ in Table 8 reports the median of these values among r replications. Furthermore defining

$$\texttt{v}\hat{}_{\texttt{opt}}\texttt{Error}^{\texttt{r}} = |\hat{v}^r_{opt} - P^*| \tag{17}$$

as the absolute difference between the true optimal reward maximizing price $P^* = 43$, and its estimation $\hat{v}^r_{opt}$; the column $\texttt{v}\hat{}_{\texttt{opt}}\texttt{Error}$ in Table 8 reports the median of $\texttt{v}\hat{}_{\texttt{opt}}\texttt{Error}^{\texttt{r}}$ values among r replications. Since Algorithm 3 may output different values for $\hat{n}$, (11) and (12) would not be a base for a fair comparison, we focused on only comparing the optimal valuation and its deviation. Using these estimations, we set the hyperparameters as $m = 20$ and $m' = 250$, with $T = 5000$ data points.

For the probabilistic setting, we report in Table 9 the frequency of $\hat{n}$ estimations, calculated identically. Furthermore, to test the performance of Algorithm 1 with $T = 1000$ and $k = 45$, we report in Table 10 the median of $\hat{v}^r_{opt}$ as $\hat{v}_{opt}$, and the median of the distances in (17), where $P^* = 40.85$; along with `WithinInt` and `WithinIntWide`.

Probabilistic valuations require significantly more data points for learning. In this setting, our selection of hyperparameters are $m = 20$ and $m' = 400$, with $T = 8000$

Table 8: Algorithm 1 performance in Scenario 3 deterministic valuations

		$\hat{v}_{opt}$	$\hat{\mathtt{v}}_{\mathtt{opt}}\mathtt{Error}$
$m = 20$	$m' = 100$	42.966	0.825
	$m' = 250$	42.941	0.71
	$m' = 500$	43.12	0.774
$m = 50$	$m' = 40$	42.693	1.042
	$m' = 100$	42.892	0.821
	$m' = 200$	43.009	0.744

Table 9: Algorithm 3 segment estimations in Scenario 3 probabilistic valuations

		Frequency of						
		$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 3$	$\hat{n} = 4$	$\hat{n} = 5$	$\hat{n} = 6$	$\hat{n} = 7$
$m = 20$	$m' = 200$	0	0	0	0.812	0.168	0.019	0.001
	$m' = 400$	0	0	0	0.948	0.052	0	0
	$m' = 750$	0	0	0	0.986	0.014	0	0
$m = 50$	$m' = 80$	0	0	0	0.378	0.442	0.126	0.054
	$m' = 160$	0	0	0	0.552	0.343	0.09	0.015
	$m' = 300$	0	0	0	0.696	0.277	0.027	0

Table 10: Algorithm 1 performance in Scenario 3 probabilistic valuations

		$\hat{v}_{opt}$	$\hat{\mathtt{v}}_{\mathtt{opt}}\mathtt{Error}$	`WithinInt`	`WithinIntWide`
$m = 20$	$m' = 200$	38.949	2.104	78.1	96.7
	$m' = 400$	38.849	2.118	78.8	97.6
	$m' = 750$	38.751	2.167	78.8	98.1
$m = 50$	$m' = 80$	39.325	2.019	65.9	88.6
	$m' = 160$	38.847	2.175	73.8	93.5
	$m' = 300$	38.748	2.248	76.9	95.5

data points. This selection yields a probability of 0.976 for the optimal price being within the online stage search interval.

4.3 Online Learning Experiments

The main goal of the seller in this study is revenue maximization, and to analyze the impact of market structure information on the total reward collected. Offline learning stage prioritizes deriving market parameter estimates in the vicinity of true parameter values, and neglects revenue maximization. The total loss from the maximum reward can originate either from suboptimal decisions due to poor parameter estimations,

or from the offline learning stage that disregards revenue maximization. Since different market knowledge scenarios require different number of customer interactions for offline learning stage; their loss from the optimal reward is expected to be at different levels.

For a fair comparison among the three scenarios, we allow them a fixed number of customer interactions, T_{total}, and allocate different portions to offline learning pricing, T_{off}, and online learning pricing, T_{on}. Eventually, we will compare the total reward collected from these T_{total} customers for each scenario, and relate this loss to missing market information.

In the remainder of this section, we investigate settings with deterministic and probabilistic valuations separately.

4.3.1 Deterministic Valuations

The scenario with the least prior market knowledge, Scenario 3, requires the most data during the offline learning stage; and as the bottleneck, we determine the horizon T_{total} based on Scenario 3. As previously discussed, Scenario 3 with deterministic valuations requires 5000 discrete random prices for Algorithm 3 and 1000 continuous random prices for Algorithm 1; making $T_{off} = 6000$. With an additional $T_{on} = 1000$ customers with online learning pricing, we set $T_{total} = 7000$ for deterministic valuations horizon.

Scenario 2 requires 4000 discrete random prices for Algorithm 2 and 500 continuous random prices for Algorithm 1, and Scenario 1 requires 500 continuous random prices for Algorithm 1. With $T_{total} = 7000$, T_{on} for Scenario 2 and Scenario 1 is 2500 and 6500 respectively. The distribution of prices in these three scenarios are summarized in Figure 10.

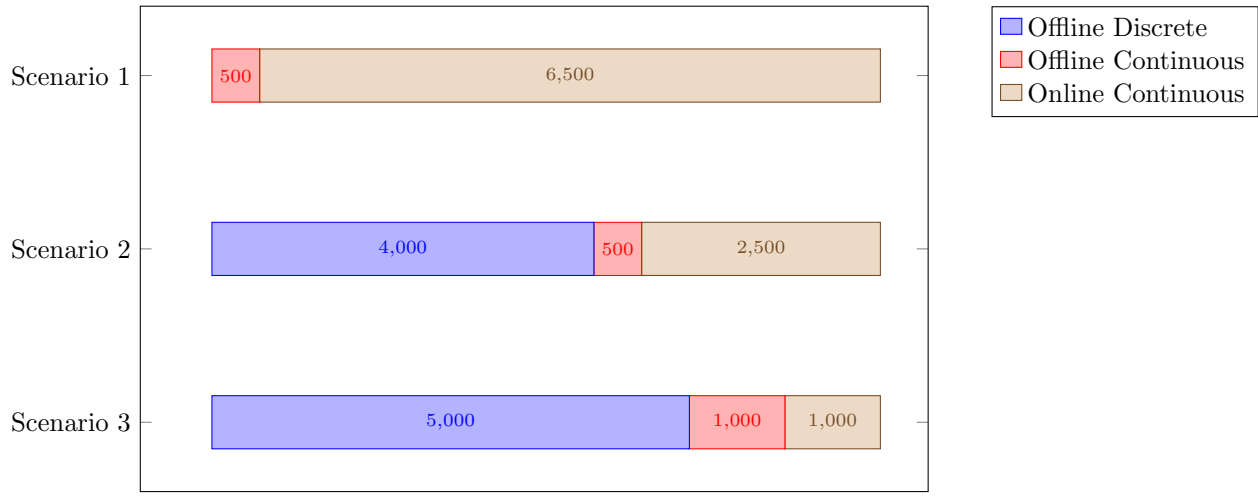


Fig. 10: Distribution of data points utilized in offline and online phases for each scenario with deterministic valuations

Settings with deterministic valuations utilize Algorithm 4 in the online stage. As for the parameters used in Algorithm 4 implementation, we set the exploitation rate $\epsilon = 1$, initial value for $\alpha = 0.25$, decay rate $\eta = 0.99$, and steady trend parameters , $\tau = 0.001$, $\omega = 100$, and $\gamma = 50$.

In each replication, given T_{off} offline learning customers, and T_{on} online learning customers, we measure the following metrics.

1. `Total Offline Reward:` The total reward collected through T_{off} customer interactions with offline pricing
2. `Average Offline Reward:` The average reward collected through T_{off} customer interactions with offline pricing (`Total Offline Reward`$/T_{off}$)
3. `Total Online Reward:` The total reward collected through T_{on} customer interactions with online pricing
4. `Average Online Reward:` The average reward collected through T_{on} customer interactions with online pricing (`Total Online Reward`$/T_{on}$)
5. `Total Reward:` The total reward collected through T_{off} customer interactions with offline pricing and T_{on} customer interactions with online pricing (`Total Reward` = `Total Online Reward` + `Total Offline Reward`)
6. `Average Total Reward:` The average reward through T_{off} customer interactions with offline pricing and T_{on} customer interactions with online pricing (`Total Reward`$/T_{total}$)
7. `Optimal Average Expected Reward:` Optimal expected reward per interaction, calculated using true market parameters
8. `Optimal Total Expected Reward:` Optimal expected reward of T_{total} customer interactions
   (`Optimal Total Expected Reward` $\times T_{total}$)

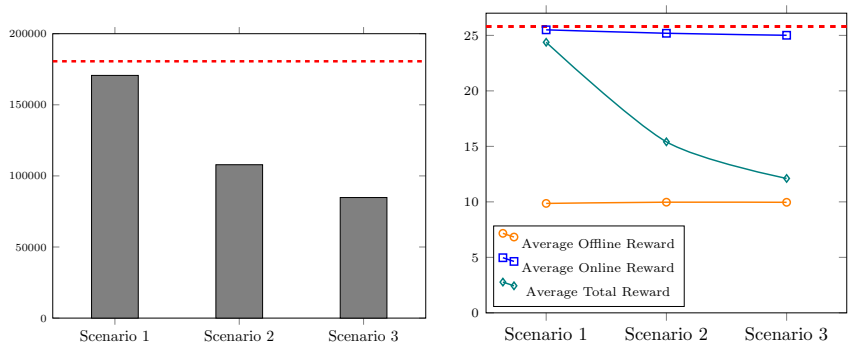
In Table 11, we provide the median of these metrics among 1000 replications.

Table 11: Rewards collected during the learning horizon for deterministic valuations

	Scenario 1	Scenario 2	Scenario 3
`Average Offline Reward`	9.86	9.97	9.96
`Average Online Reward`	25.5	25.19	25.01
`Average Total Reward`	24.38	15.41	12.11
`Optimal Average Expected Reward`		25.8	
`Total Offline Reward`	4,929.46	44,868.70	59,771.26
`Total Online Reward`	165,727.57	62,982.88	25,010.45
`Total Reward`	170,671.21	107,860.71	84,785.19
`Optimal Total Expected Reward`		180,600	

The results in Table 11 are also visualized in Figure 11. The dashed red lines in both figures denote the optimal reward corresponding to full information setting, which can be interpreted as the benchmark. The prices offered during the offline learning stage is clearly not a successful strategy for revenue maximization. For all three scenarios, we

observe a similar reward corresponding to around 40% of the optimal average expected reward. The results further indicate that the amount of customers allocated to learn the market structure through offline learning was sufficient in bringing the `Average Online Reward` close to the optimal reward, promising convergence to the optimal value with a longer horizon. For the current horizon of 7000 customers, `Average Online Reward` for the three scenarios are 98.8%, 97.6%, and 96.9% of the optimal expected reward per customer respectively. The main distinction between the three scenarios is observed in the total rewards, and is caused by the varying duration of the offline learning stage. Although the impact of warm-up periods disappears on the infinite horizon case, for the current horizon the reward lost due to lack of market information is 5.5%, 40.3%, and 53.1% of the optimal expected total reward for scenarios 1, 2, and 3 respectively. These results are promising in showing how well we can learn given the market segment structure except for individual product valuations for each segment (i.e. Scenario 1), and how critical market segment structure is (i.e. the number of segments and their corresponding fractions).



(a) Total Reward (b) Average Reward

Fig. 11: Reward comparison by scenarios with deterministic valuations

4.3.2 Probabilistic Valuations

Similarly, for probabilistic valuations, we start with Scenario 3 to determine the horizon T_{total}. Scenario 3 with probabilistic valuations requires 8000 discrete random prices for Algorithm 3 and 1000 continuous random prices for Algorithm 1. Additionally, we will have 3000 customers with online pricing, 2000 of which will be utilized for the exploitation rate ϵ to gradually reach 1 from its original value, and the remaining 1000 will be pure exploitation. This sets the horizon to $T_{total} = 12000$ for probabilistic valuations. With Scenario 2 requiring 5000 discrete random prices for Algorithm 2 and 1000 continuous random prices for Algorithm 1, and Scenario 1 requiring 1000 continuous random prices for Algorithm 1, we get the distribution of prices given in Figure 12.

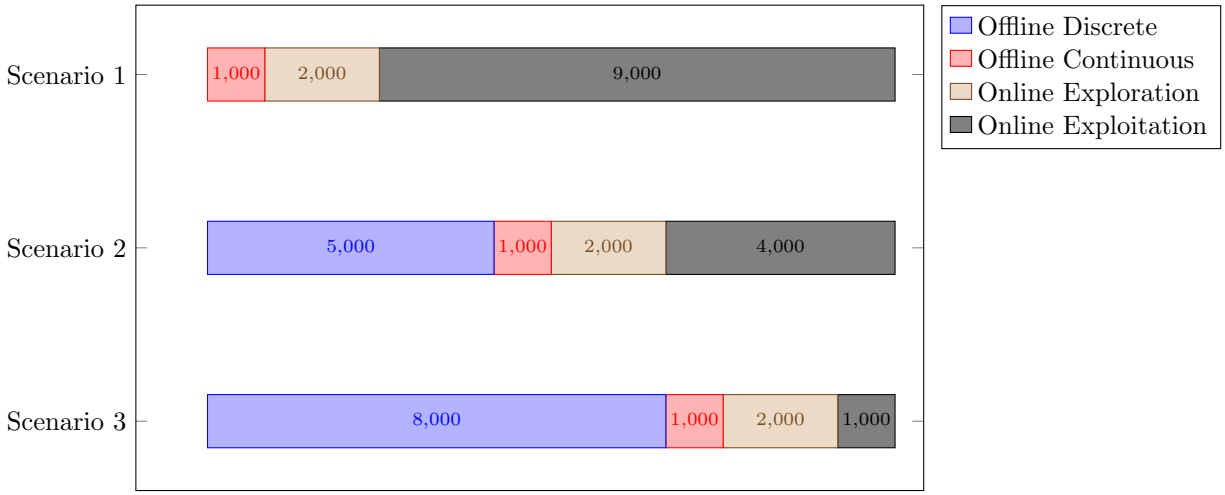


Fig. 12: Distribution of Data Points Utilized in Offline and Online Phases for Each Scenario with Probabilistic Valuations

Settings with probabilistic valuations utilize Algorithm 5 in the online phase. Algorithm 8 is used as the interval estimation methods, which returns the averages of estimations by Algorithms 6 and 7. As for the parameters used in these algorithm implementations, we set the relaxation factor $\rho = 1.05$ in Algorithm 6, proximity threshold $\tau = 0.01$ in Algorithm 7, and widen the interval by $\phi = 0.05$. Exploitation rate ϵ is updated by Subroutine 2 with $\epsilon_0 = 0$.

In each replication, given T_{off} offline learning customers, and T_{on} online learning customers, we measure the following metrics in addition to the metrics stated in the previous section.

1. `Total Exploitation Reward`: The total reward collected through the last $T_{on} - 2000$ customer interactions with online pricing with $\epsilon = 1$
2. `Average Exploitation Reward`: The average reward collected through the last $T_{on} - 2000$ customer interactions with online pricing with $\epsilon = 1$ (`Total Exploitation Reward`$/(T_{on} - 2000)$)

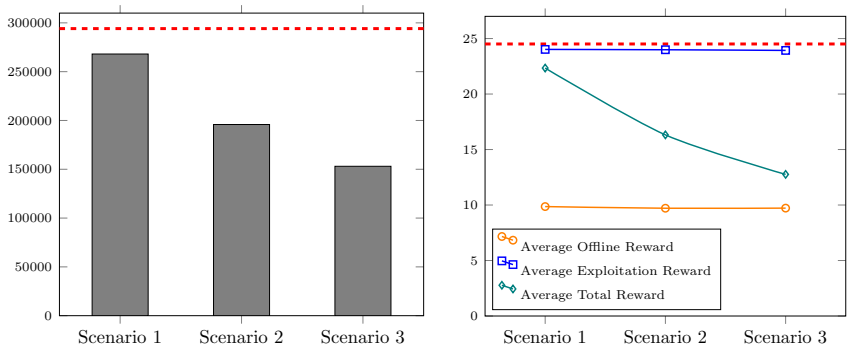
In Table 12, we provide the median of these metrics among 1000 replications.

The results in Table 12 are also visualized in Figure 13. The optimal reward corresponding to full information setting is again denoted by the dashed red lines in both figures. We observe a similar poor performance for the average offline rewards, around 40% of the benchmark value. Instead of `Average Online Reward`, we reported `Average Exploitation Reward` as those were the values that were offered with pure reward maximization motivation. These results are also favorable, corresponding to 98%, 97.9%, and 97.6% of the optimal average expected reward for the three scenarios respectively. The reward lost due to lack of market information in this setting is 8.9%, 33.4%, and 48% of the optimal expected total reward for scenarios 1, 2, and 3 respectively. The value of valuation information appears to have increased, and the value of

Table 12: Rewards collected during the learning horizon for deterministic valuations

	Scenario 1	Scenario 2	Scenario 3
`Average Offline Reward`	9.86	9.71	9.72
`Average Online Reward`	23.48	22.95	21.86
`Average Exploitation Reward`	24.02	23.99	23.93
`Average Total Reward`	22.34	16.32	12.76
`Optimal Average Expected Reward`		24.51	
`Total Offline Reward`	9,858.55	58,246.42	87,497.51
`Total Online Reward`	258,241.77	137,688.89	65,580.91
`Total Exploitation Reward`	216,211.62	95,972.96	23,927.58
`Total Reward`	268,079.28	195,861.93	153,063.94
`Optimal Total Expected Reward`		294,120	

number of segments and the segment fractions information appears to have decreased; however the difference between these values of information remain to be significant.



(a) Total Reward (b) Average Reward

Fig. 13: Reward comparison by scenarios with probabilistic valuations

5 Conclusion

In decision-making systems, particularly in multiperiod problems, firms often lack essential data before making decisions. This issue is especially relevant in pricing, where incomplete knowledge of market structures presents significant challenges. Our study emphasizes the importance of leveraging customer purchasing preferences to iteratively learn unknown market parameters, such as the number of segments, segment sizes, and willingness-to-pay values of the segments, and dynamically adapt pricing strategies. Our algorithms effectively bridge the gap between prior knowledge and revenue maximization by integrating offline learning to explore market dynamics and online learning to refine pricing. By incorporating offline learning to explore and

understand market structure and online learning to exploit these insights for revenue maximization, our approach ensures iterative improvement and convergence.

Our experiments demonstrated that even in the absence of any prior market knowledge, it is possible to learn the market structure and determine the optimal price using only customers' purchasing preferences.

While learning market structures provides valuable insights, it comes at a cost. The objectives of learning the market structure and maximizing the revenue are not perfectly aligned. When learning is prioritized to improve decision-making with limited prior knowledge, there is an opportunity cost due to delayed revenue optimization. To quantify these trade-offs, we analyzed three scenarios with varying levels of prior knowledge: (i) knowledge of the number of segments and segment sizes, (ii) knowledge of only the number of segments, and (iii) no prior market knowledge. By calculating the rewards achieved under each scenario and comparing them to the full-information case, we attributed the differences in rewards to incomplete market information. The findings are that missing only the valuation information leads to 5-9 % loss, missing segments size and valuation information leads to 33-40% loss, and having no market information leads to 48-53% loss of total reward that can be collected for the studied experimental setting.

It should be noted that the differences calculated and reported in this study should not be deemed "exact" values of information; but are rather dependent on the existing market and our parameter selection based on our thresholds for a successful parameter estimation. For this study, we conducted numerous experiments for fine-tuning purposes, and in settings where fewer customers were allocated for offline learning, we either observed convergence to local but not global optimal prices when the online learning stage remains pure exploitation, or a loss due to additional exploration in the online learning stage. In either case missing the segment size and the number of segment information led to significant reward loss.

The findings observed in our experiments may provide valuable insights for managerial decision-making. These results allow firms to compare the costs of implementing data-driven adaptive pricing strategies with the potential expenses of acquiring market insights from external sources. Decision-making can be further enhanced by tailoring experiments to reflect more relevant market structures, by incorporating existing beliefs about the market. It is also possible to conduct targeted simulations and calculate confidence intervals for potential reward losses.

In this study, we have outlined a comprehensive roadmap for addressing the challenges of pricing under incomplete market knowledge through the learning algorithms we proposed. These algorithms successfully balance exploration and exploitation, adapting to varying levels of prior knowledge and market structures. The results achieved demonstrate the effectiveness of our approach, providing a foundation for future research and practical applications in dynamic pricing environments.

Acknowledgements. This work was supported by TÜBİTAK (Scientific and Technological Research Council of Turkey), grant number 221M392.

References

[1] Jefferson, G.: Did Apple retail prices get too high in 2018? Consumers say yes. https://www.usatoday.com/story/tech/talkingtech/2018/12/29/did-apple-retail-prices-get-too-high-2018-consumers-say-way-yes/2432445002/. Accessed: 01-16-2025 (2018)

[2] Rado, M.: Samsung Galaxy S20 series still struggling to sell. https://www.phonearena.com/news/Samsung-galaxy-s20-struggling-sales_id123344. Accessed: 01-16-2025 (2020)

[3] Bertsimas, D., Perakis, G.: Dynamic pricing: A learning approach. Mathematical and computational models for congestion charging, 45–79 (2006)

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

[5] Boer, A.V., Zwart, B.: Simultaneously learning and optimizing using controlled variance pricing. Management Science **60**(3), 770–783 (2014)

[6] Keskin, N.B., Zeevi, A.: Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies. Operations Research **62**(5), 1142–1167 (2014)

[7] Araman, V.F., Caldentey, R.: Dynamic pricing for nonperishable products with demand learning. Operations Research **57**(5), 1169–1188 (2009)

[8] Farias, V.F., Van Roy, B.: Dynamic pricing with a prior on market response. Operations Research **58**(1), 16–29 (2010)

[9] Eren, S.S., Maglaras, C.: Monopoly pricing with limited demand information. Journal of Revenue and Pricing Management **9**(1), 23–48 (2010)

[10] Ferreira, K.J., Lee, B.H.A., Simchi-Levi, D.: Analytics for an online retailer: Demand forecasting and price optimization. Manufacturing & Service Operations Management **18**(1), 69–88 (2016)

[11] Chen, N., Wang, C., Wang, L.: Learning and optimization with seasonal patterns. Operations Research (2023) https://doi.org/10.1287/opre.2023.0017 . Ahead of print

[12] Boer, A.V., Keskin, N.B.: Dynamic pricing with demand learning and reference effects. Management Science **68**(10), 7112–7130 (2022)

[13] den Boer, A.V.: Tracking the market: Dynamic pricing and learning in a changing environment. European Journal of Operational Research **247**(3), 914–927 (2015)

[14] Kazemi, M.S., Fotopoulos, S.B., Wang, X.: Minimizing online retailers' revenue loss under a time-varying willingness-to-pay distribution. International Journal of Production Economics **257**, 108767 (2023)

[15] Keskin, N.B., Li, M.: Selling quality-differentiated products in a markovian market with unknown transition probabilities. Operations Research **72**(3), 885–902 (2024)

[16] Rana, R., Oliveira, F.S.: Real-time dynamic pricing in a non-stationary environment using model-free reinforcement learning. Omega **47**, 116–126 (2014)

[17] Yang, X., Zhang, J., Hu, J.-Q., Hu, J.: Nonparametric multi-product dynamic pricing with demand learning via simultaneous price perturbation. European Journal of Operational Research **319**(1), 191–205 (2024)

[18] Zhang, M., Ahn, H.-S., Uichanco, J.: Data-driven pricing for a new product. Operations Research **70**(2), 847–866 (2022)

[19] Araman, V.F., Caldentey, R.: Revenue Management with Incomplete Demand Information. John Wiley & Sons, Ltd, ??? (2011)

[20] den Boer, A.V.: Dynamic pricing and learning: Historical origins, current research, and new directions. Surveys in Operations Research and Management Science **20**(1), 1–18 (2015)

[21] Ferreira, K.J., Simchi-Levi, D., Wang, H.: Online network revenue management using thompson sampling. Operations research **66**(6), 1586–1602 (2018)

[22] Miao, S., Chao, X.: Dynamic joint assortment and pricing optimization with demand learning. Manufacturing & Service Operations Management **23**(2), 525–545 (2021)

[23] Kramer, O.: K-nearest neighbors. In: Kramer, O. (ed.) Dimensionality Reduction with Unsupervised Nearest Neighbors, pp. 13–23. Springer, ??? (2013). https://doi.org/10.1007/978-3-642-38652-7_2

[24] Beyer, H.: Tukey, john w.: Exploratory data analysis. addison-wesley publishing company reading, mass. — menlo park, cal., london, amsterdam, don mills, ontario, sydney 1977, xvi, 688 s. Biometrical Journal **23**(4), 413–414 (1981)

[25] Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) **28**(1), 100–108 (1979)

Appendix A Methods to estimate the intervals of probabilistic valuations

When Algorithm 1 is applied to a market with uniformly distributed valuations with nonoverlapping intervals (l_i, u_i), as illustrated in Figure 3, the data points with $x^t \in [l_i, u_i]$ will be divided such that approximately half is assigned to one cluster, and the remaining to the subsequent cluster. Consider the clustering outcome of estimated purchase probabilities of one such market, shown in Figure A1. The vertical dashed blue lines indicate the randomness intervals (l_i, u_i), and the points in these intervals are divided approximately halfway and assigned different cluster assignment colors.

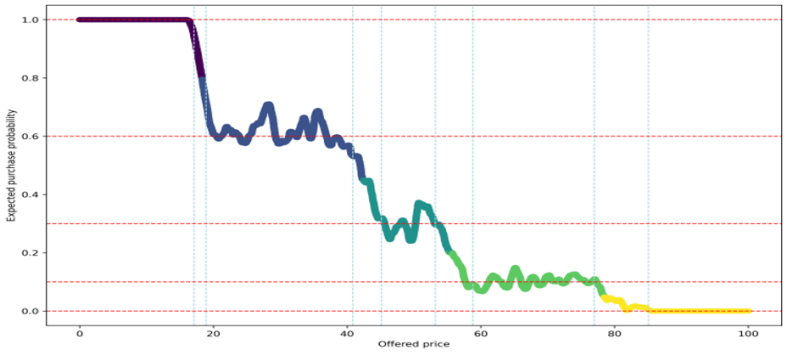


Fig. A1: An example cluster assignment of $(x^t, \hat{y}^t)$ as Algorithm 1 output

In such probabilistic valuations; in clusters obtained as the result of Algorithm 1; we expect to see a structure with three components in ascending order of x values: Component 1 has a decreasing trend in $\hat{y}$, starting with higher values and decreasing to the expected purchase probability of the cluster; Component 2 has a steady trend in $\hat{y}$ around the expected purchase probability of the cluster; and Component 3 has a decreasing trend in $\hat{y}$, starting with the expected purchase probability of the cluster and decreasing to lower values. In Figure A1, you can observe component 1 as the left tail of the cluster that falls between the blue vertical lines, component 2 as the oscillations around the horizontal red lines, and component 3 as the right tail of the cluster that falls between the blue vertical lines. The first and the last clusters are an exception to this pattern; where the first cluster only has components 2 and 3, and the last cluster only has components 1 and 2.

Ideally, the breakpoints between these components correspond to l_i and u_i values. Namely, for cluster i, the transition between component 1 and component 2 is expected to occur at $x = u_{i-1}$, and the transition between component 2 and component 3 is expected to occur at $x = l_i$. Therefore, given the cluster assignments as the output of Algorithm 1; for estimating the parameters l_i and u_i, we analyze each cluster to detect components 1, 2, and 3, and their transition points.

A.1 Method 1

Algorithms 1, 2, and 3 are all batch learning algorithms, where the order of points is irrelevant. Therefore, without loss of generality, we can rearrange the training data set $\{(x^t, y^t)\}_{t=1}^{T}$ such that for any $t_1 < t_2$, $x^{t_1} \leq x^{t_2}$ is satisfied. Given Algorithm 1's output cluster assignments, let t^i_{max} be the highest index among the data points assigned to cluster i. In other words, indices $\{t^{i-1}_{max} + 1, \ldots, t^i_{max}\}$ belong to cluster i, where $t^0_{max} = 0$.

A key element in Method 1 is the difference between the estimated purchase probability of point t, and the estimated expected purchase probability of the assigned cluster i: $d_t = \hat{p}^i - \hat{\hat{y}}^t$ for $i = 1, \ldots, n$ and $t = t^{i-1}_{max} + 1, \ldots, t^i_{max}$.

The value of d_t in component 1 is expected to be negative, and approaching 0 as t increases; and in component 3 it is expected to be positive, and approaching 0 as t decreases. Therefore, to find the bounds for components 1 and 3, we propose to detect these trends in the respective d_t values. Ideally, we would like to observe $d_t \leq d_{t+1}$ in the intervals with decreasing patterns; however, the random realizations may lead to minor fluctuations in $\hat{\hat{y}}^t$ values. In order to account for those fluctuations and preempt premature converge, we relax the condition by a factor of ρ for some ρ slightly greater than 1. Due to the change in sign of d_t in components 1 and 3, this relaxation becomes $\rho \cdot d_t \leq d_{t+1}$ for component 1 (where the values are mostly negative), and $d_j \leq \rho \cdot d_{t+1}$ in component 3 (where the values are mostly positive).

The search for finding the transition between components 1 and 2 start from the beginning of the cluster, and proceed with increasing the value of t; whereas the search for finding the transition between components 2 and 3 start from the end of the cluster, and proceed with decreasing the value of t. Once the respective conditions are violated, the algorithm terminates and estimates the current location as the transition point between components.

Algorithm 6 Interval Estimation Method 1

for $i = 2, \ldots, n + 1$ **do**
  for $t = t^{i-1}_{max} + 2, \ldots, t^i_{max}$ **do**
    if $\rho \cdot d_t > d_{t+1}$ **then**
      $\hat{u}_{i-1} = x^t$
      break
    end if
  end for
end for
for $i = 1, \ldots, n$ **do**
  for $t = t^i_{max} - 1, \ldots, t^{i-1}_{max} - 1$ (reverse order) **do**
    if $d_t > \rho \cdot d_{t+1}$ **then**
      $\hat{l}_i = x^t$
      break
    end if
  end for
end for

A.2 Method 2

Method 1 reaches the break points between the components starting from the end-points of the cluster and moving towards the center through components 1 and 3. Method 2, however, can be interpreted to start from the center and move towards the end points of the cluster through component 2.

In expectation, $\hat{y}^t$ values in component 2 should be close to the expected purchase probability of the cluster p^i. We aim to find the longest interval in the center, such that this proximity is preserved. Given a threshold τ to denote sufficient proximity,

$$t_i^u = \min\left\{t \in \{t_{max}^{i-1}+1, \ldots, t_{max}^i\} \ : \ |\hat{p}^i - \hat{y}^t| \leq \tau\right\} \tag{A1}$$

and

$$t_i^l = \max\left\{t \in \{t_{max}^{i-1}+1, \ldots, t_{max}^i\} \ : \ |\hat{p}^i - \hat{y}^t| \leq \tau\right\} \tag{A2}$$

yield the estimates $\hat{u}_i = x^{t_{i+1}^u}$, and $\hat{l}_i = x^{t_i^l}$ for $i = 1, \ldots, n$.

Algorithm 7 Interval Estimation Method 2

1: **for** $i = 1, \ldots, n+1$ **do**
2: Calculate t_i^u and t_i^l
3: **end for**
4: **for** $i = 1, \ldots, n$ **do**
5: Estimate $\hat{u}_i = x^{t_{i+1}^u}$
6: Estimate $\hat{l}_i = x^{t_i^l}$
7: **end for**

A.3 Method 3

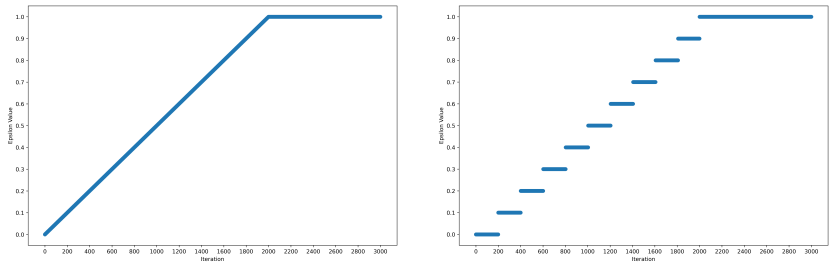
The third method is a hybrid method that combines the two previous methods to achieve a balanced estimation of the bounds. This method involves taking the average of the estimated bounds obtained from methods 1 and 2.

Algorithm 8 Interval Estimation Method 3

1: Run Algorithm 6 to get the bound estimates $\hat{l}_i^1$ and $\hat{u}_i^1$
2: Run Algorithm 7 to get the bound estimates $\hat{l}_i^2$ and $\hat{u}_i^2$
3: **for** $i = 1, \ldots, n$ **do**
4: Estimate $\hat{u}_i = \frac{\hat{u}_i^1 + \hat{u}_i^2}{2}$
5: Estimate $\hat{l}_i = \frac{\hat{l}_i^1 + \hat{l}_i^2}{2}$
6: **end for**

Appendix B Methods for updating ϵ

Assuming dedicating a portion of online learning for exploring the interval of search with a length of T_{online}^{explr}, we start with an initial epsilon value ϵ_0 and gradually increase its value to 1. This is proposed to be achieved in two ways as illustrated in Figure B2: continuously and periodically.



(a) Continuous (b) Periodical

Fig. B2: ϵ value increase patterns

In continuous updates, at each iteration, its value incremented by $\frac{1-\epsilon_0}{T_{online}^{explr}}$ starting from ϵ_0 till $\epsilon = 1$. This is summarized in Subroutine 1.

Subroutine 1 Continuous ϵ Update Subroutine

$\epsilon = \min\left\{1, \epsilon + \frac{1-\epsilon_0}{T_{online}^{explr}}\right\}$

In periodical updates, the exploration phase is divided into ξ periods of length $\frac{T_{online}^{explr}}{\xi}$. The value of ϵ increases every $\frac{T_{online}^{explr}}{\xi}$ iterations, by $\frac{1-\epsilon_0}{\xi}$ for a total of ξ times. This is summarized in Subroutine 2, where j refers to the iteration number.

Subroutine 2 Periodical ϵ Update Subroutine

if $j \mod \frac{T_{online}^{explr}}{\xi} = 0$ **then**
    $\epsilon = \min\left\{1, \epsilon + \frac{1-\epsilon_0}{\xi}\right\}$
end if

Appendix C Extended Results

Table C1: Algorithm 1 performance in Scenario 1 deterministic valuations - Extended

		$\hat{v}_1$	$\hat{v}_2$	$\hat{v}_3$	$\hat{v}_4$	`vError`	`TrueBP`
$T = 50$	$k = 5$	22.43	39.5	55.04	68.06	20.97	47.8
	$k = 7$	19.66	40.54	57.55	72.01	17	62.8
	$k = 9$	20.04	41.28	57.28	74.4	16.46	63.5
	$k = 11$	21.17	41.55	57.01	74.39	15.85	63.6
$T = 100$	$k = 5$	22.17	37.93	54.62	67.11	18.73	49.3
	$k = 7$	19.62	40.17	58.38	71.01	14.91	67.3
	$k = 9$	19.14	41.31	57.46	72.93	13.59	71.6
	$k = 11$	19.04	42.19	56.56	74.99	12.56	76.5
	$k = 13$	18.58	42.46	57.36	76.82	11.92	81
	$k = 15$	18.59	42.1	56.45	77.11	12.1	82.2
$T = 250$	$k = 5$	21.29	37	54.56	67.45	16.07	54.3
	$k = 10$	18.73	41.77	56.35	72.74	9.15	87.2
	$k = 15$	18.28	42.23	56.33	76.82	7.51	92.2
	$k = 20$	18.14	42.54	56.28	78.79	7.12	95.6
	$k = 25$	18.31	42.53	56.03	78.95	7.4	95.9
	$k = 30$	18.04	42.92	55.95	79.46	7.05	96.6
$T = 500$	$k = 15$	18.2	42.68	55.75	77.14	5.52	97.5
	$k = 20$	18.11	42.81	55.79	77.87	5	98.3
	$k = 25$	18.11	42.98	56.03	79.28	4.39	98.8
	$k = 30$	18.16	42.99	56.07	79.63	4.19	99.1
	$k = 35$	18.09	42.94	55.88	80.1	4.11	99.7
	$k = 40$	18.06	43.07	55.91	80.3	4.26	99.8
	$k = 45$	18.06	42.96	55.87	80.41	4.34	99.7
	$k = 50$	18.1	43.14	55.83	80.47	4.49	100
	$k = 60$	18.1	42.82	55.66	80.65	4.54	99.8
$T = 1000$	$k = 20$	18.06	42.83	55.47	77.88	3.63	99.9
	$k = 25$	18.04	42.95	55.81	79.37	2.9	100
	$k = 30$	18.02	43.06	55.79	80	2.66	99.9
	$k = 35$	18.04	43.02	55.86	80.28	2.49	99.8
	$k = 40$	18	42.98	55.98	80.47	2.31	99.9
	$k = 45$	18.02	43.03	55.83	80.72	2.3	99.9
	$k = 50$	18.07	42.94	55.97	80.63	2.34	100
	$k = 60$	18.08	43.03	55.92	80.8	2.44	100
	$k = 75$	18.03	42.95	55.99	80.81	2.61	100
	$k = 100$	18.08	43.12	56.01	80.86	2.98	100

Table C2: Algorithm 1 performance in Scenario 1 probabilistic valuations - Extended

		$\hat{\bar{v}}_1$	$\hat{\bar{v}}_2$	$\hat{\bar{v}}_3$	$\hat{\bar{v}}_4$	`v̄Error`	`WithinInt`	`WithinIntWide`
$T = 50$	$k = 5$	28.23	37.86	46.74	59.22	30.23	50.2	58.9
	$k = 7$	31.81	37.52	48.84	64.95	33.61	42.6	50.7
	$k = 9$	32.95	37.36	48.53	65.73	34.96	39.1	46.6
	$k = 11$	34.88	38.11	48.23	66.23	36.96	39.8	48.3
$T = 100$	$k = 5$	24.35	38.41	46.29	58.45	24.39	44.4	53.1
	$k = 7$	28.13	36.29	48.76	64.18	27.91	38.1	45.6
	$k = 9$	29.65	36.95	49.22	65.82	28.51	44.4	53.1
	$k = 11$	30.92	36.94	48.87	68.93	30.03	41.6	50.6
	$k = 13$	31.31	37.17	48.81	67.97	30.08	44.4	52.2
	$k = 15$	32.25	37.64	48.64	67.47	31.7	44.4	53.1
$T = 250$	$k = 5$	19.42	40.19	44.74	56.99	17.87	32.2	45.4
	$k = 10$	22.94	36.45	51.07	69.38	17.28	39.9	51.4
	$k = 15$	22.73	37.17	50.69	72.03	16.26	59.6	71.6
	$k = 20$	25.79	36.82	50.1	73.34	20.59	65.2	75
	$k = 25$	28.17	37.02	49.29	72.99	23.61	61.8	69.8
	$k = 30$	30.07	37.09	49.03	71.64	26.93	62.4	70.3
$T = 500$	$k = 15$	16.67	39	52.52	74.84	8.37	56.2	74.7
	$k = 20$	18.04	38.35	52.24	75.52	9.06	66.2	82.6
	$k = 25$	19.27	38.17	51.69	75.81	10.12	72.5	90.2
	$k = 30$	20.88	37.6	50.82	75.38	11.52	77.8	90.6
	$k = 35$	22.08	37.55	50.7	75.11	13.09	80.2	91.6
	$k = 40$	23.69	37.15	50	74.17	16.42	84.7	90.5
	$k = 45$	26.09	36.92	49.56	74.11	20.09	83.6	89.4
	$k = 50$	26.16	36.98	49.69	73.37	19.66	83.7	89.6
	$k = 60$	28.88	36.56	49.84	72.09	24.36	84.4	88.5
$T = 1000$	$k = 20$	16.15	40.51	53.45	76.93	4.89	48.1	82.5
	$k = 25$	15.92	40.16	53.36	77.6	5	57	90
	$k = 30$	15.76	39.93	52.97	77.62	5.51	65.8	93.4
	$k = 35$	16	39.4	52.58	77.44	6.3	71.6	97
	$k = 40$	16.57	39.07	52.11	77.31	7.01	77.3	97.7
	$k = 45$	17.41	39.1	51.86	77.03	7.28	81.2	98.8
	$k = 50$	18.21	38.32	51.68	76.47	7.88	82.1	97.6
	$k = 60$	19.84	37.91	50.57	76.2	9.69	89.6	98.7
	$k = 75$	22.05	37.26	50.03	75.26	13.03	93.6	98.7
	$k = 100$	26.21	36.39	49.92	73.52	20.49	95.7	97

Table C3: Algorithm 2 performance in Scenario 2 deterministic valuations - Extended

		$\hat{\delta}_1$	$\hat{\delta}_2$	$\hat{\delta}_3$	$\hat{\delta}_4$	δError
$m = 20$	$m' = 50$	0.4	0.296	0.201	0.097	12.42
	$m' = 100$	0.4	0.301	0.199	0.098	8.58
	$m' = 150$	0.4	0.299	0.201	0.1	7.09
	$m' = 200$	0.4	0.3	0.201	0.099	5.86
	$m' = 250$	0.4	0.3	0.2	0.099	5.22
	$m' = 500$	0.4	0.301	0.2	0.1	3.77
	$m' = 1000$	0.4	0.301	0.199	0.1	2.69
$m = 50$	$m' = 20$	0.386	0.295	0.215	0.1	13.75
	$m' = 40$	0.398	0.303	0.201	0.1	8.89
	$m' = 60$	0.399	0.299	0.201	0.1	7.41
	$m' = 80$	0.401	0.301	0.199	0.1	6.23
	$m' = 100$	0.4	0.301	0.199	0.099	5.53
	$m' = 200$	0.399	0.301	0.199	0.1	3.81
	$m' = 400$	0.4	0.3	0.2	0.1	2.65

Table C4: Algorithm 1 performance in Scenario 2 deterministic valuations - Extended

		$\hat{v}_1$	$\hat{v}_2$	$\hat{v}_3$	$\hat{v}_4$	vError	TrueBP
$m = 20$	$m' = 50$	18.18	42.96	55.98	80.22	4.61	98.7
	$m' = 100$	18.12	42.89	55.94	80.26	4.33	98.9
	$m' = 150$	18.14	42.76	55.93	80.02	4.27	99.4
	$m' = 200$	18.11	42.88	56.03	80.14	4.36	99.2
	$m' = 250$	18.07	43.01	55.95	80.12	4.25	99.9
	$m' = 500$	18.05	42.98	56.03	80.02	4.21	99.8
	$m' = 1000$	18.16	42.88	56.06	80.33	4.19	99.5
$m = 50$	$m' = 20$	17.96	42.42	55.42	80.01	4.96	95.5
	$m' = 40$	18.12	42.92	56.08	80.24	4.39	98.9
	$m' = 60$	18.06	42.86	55.94	80.17	4.48	99
	$m' = 80$	18.08	42.99	56.03	79.96	4.2	99.3
	$m' = 100$	18.07	42.79	56.13	80.3	4.26	99.5
	$m' = 200$	18.14	42.8	55.96	80.46	4.25	99
	$m' = 400$	18.02	42.8	55.98	80.1	4.28	99.6

Table C5: Algorithm 2 performance in Scenario 2 probabilistic valuations - Extended

		$\hat{\delta}_1$	$\hat{\delta}_2$	$\hat{\delta}_3$	$\hat{\delta}_4$	δError
$m = 20$	$m' = 50$	0.396	0.314	0.191	0.094	13.29
	$m' = 100$	0.4	0.317	0.187	0.094	9.42
	$m' = 150$	0.4	0.318	0.186	0.093	8.16
	$m' = 200$	0.4	0.319	0.188	0.093	7.47
	$m' = 250$	0.4	0.317	0.189	0.093	6.73
	$m' = 500$	0.4	0.319	0.187	0.092	5.78
	$m' = 1000$	0.4	0.318	0.19	0.092	5.28
$m = 50$	$m' = 20$	0.375	0.287	0.223	0.095	15.79
	$m' = 40$	0.387	0.3	0.203	0.094	10.29
	$m' = 60$	0.388	0.296	0.201	0.095	8.43
	$m' = 80$	0.388	0.299	0.198	0.095	7.39
	$m' = 100$	0.388	0.299	0.199	0.095	6.94
	$m' = 200$	0.387	0.301	0.196	0.094	5.55
	$m' = 400$	0.387	0.301	0.195	0.093	4.62

Table C6: Algorithm 1 performance in Scenario 2 probabilistic valuations - Extended

		$\hat{\bar{v}}_1$	$\hat{\bar{v}}_2$	$\hat{\bar{v}}_3$	$\hat{\bar{v}}_4$	$\bar{v}$Error	WithinInt	WithinIntWide
$m = 20$	$m' = 50$	17.68	38.58	52.49	52.49	8.56	75.1	96
	$m' = 100$	17.65	38.51	53.12	53.12	8	78.3	95.6
	$m' = 150$	17.33	38.79	52.93	52.93	7.39	79.9	97.8
	$m' = 200$	17.42	38.91	53.07	53.07	7.2	80.3	97.5
	$m' = 250$	17.75	38.84	53.01	53.01	7.52	79.4	98.5
	$m' = 500$	17.21	38.72	53.11	53.11	7.22	80.6	98.2
	$m' = 1000$	17.47	38.51	52.95	52.95	7.16	82.7	98.3
$m = 50$	$m' = 20$	17	37.19	49.26	49.26	9.31	74.7	89.9
	$m' = 40$	17.05	38.21	51.35	51.35	8.09	80.2	94.8
	$m' = 60$	17.06	38.2	51.93	51.93	7.8	81.2	96.6
	$m' = 80$	17.13	38.41	51.91	51.91	7.86	80.2	97.9
	$m' = 100$	17.18	38.47	51.83	51.83	8.13	79.2	98
	$m' = 200$	17.33	38.59	51.87	51.87	7.5	82.2	97.6
	$m' = 400$	17.25	38.41	51.84	51.84	7.76	80.7	98.1

Table C7: Algorithm 3 segment estimations in Scenario 3 deterministic valuations - Extended

		Frequency of						
		$\hat{n}=1$	$\hat{n}=2$	$\hat{n}=3$	$\hat{n}=4$	$\hat{n}=5$	$\hat{n}=6$	$\hat{n}=7$
$m=20$	$m'=50$	0	0	0.003	0.56	0.328	0.085	0.024
	$m'=100$	0	0	0.001	0.719	0.255	0.022	0.003
	$m'=150$	0	0	0	0.831	0.163	0.005	0.001
	$m'=200$	0	0	0	0.894	0.104	0.002	0
	$m'=250$	0	0	0	0.94	0.06	0	0
	$m'=300$	0	0	0	0.954	0.046	0	0
	$m'=350$	0	0	0	0.966	0.034	0	0
	$m'=400$	0	0	0	0.966	0.034	0	0
	$m'=450$	0	0	0	0.992	0.008	0	0
	$m'=500$	0	0	0	0.984	0.016	0	0
	$m'=750$	0	0	0	0.995	0.005	0	0
	$m'=1000$	0	0	0	1	0	0	0
$m=50$	$m'=20$	0	0	0.005	0.414	0.371	0.145	0.065
	$m'=40$	0	0	0	0.461	0.396	0.099	0.044
	$m'=60$	0	0	0	0.633	0.318	0.035	0.014
	$m'=80$	0	0	0	0.757	0.231	0.01	0.002
	$m'=100$	0	0	0	0.868	0.126	0.004	0.002
	$m'=120$	0	0	0	0.918	0.081	0.001	0
	$m'=140$	0	0	0	0.926	0.071	0.003	0
	$m'=160$	0	0	0	0.954	0.045	0.001	0
	$m'=180$	0	0	0	0.973	0.027	0	0
	$m'=200$	0	0	0	0.976	0.024	0	0
	$m'=300$	0	0	0	0.994	0.006	0	0
	$m'=400$	0	0	0	1	0	0	0

Table C8: Algorithm 1 performance in Scenario 3 deterministic valuations - Extended

		$\hat{v}_{opt}$	$\hat{\mathtt{v}}_{\mathtt{opt}}\mathtt{Error}$
$m = 20$	$m' = 50$	42.866	1.05
	$m' = 100$	42.966	0.825
	$m' = 150$	42.983	0.785
	$m' = 200$	43.039	0.811
	$m' = 250$	42.941	0.71
	$m' = 300$	42.957	0.818
	$m' = 350$	42.987	0.692
	$m' = 400$	43.039	0.742
	$m' = 450$	42.946	0.736
	$m' = 500$	43.12	0.774
	$m' = 750$	43.001	0.778
	$m' = 1000$	43.077	0.802
$m = 50$	$m' = 20$	42.535	1.449
	$m' = 40$	42.693	1.042
	$m' = 60$	42.897	0.885
	$m' = 80$	42.941	0.821
	$m' = 100$	42.892	0.821
	$m' = 120$	42.92	0.778
	$m' = 140$	42.969	0.757
	$m' = 160$	42.955	0.761
	$m' = 180$	43.022	0.737
	$m' = 200$	43.009	0.744
	$m' = 300$	43.052	0.752
	$m' = 400$	43.005	0.792

Table C9: Algorithm 3 segment estimations in Scenario 3 probabilistic valuations - Extended

		Frequency of						
		$\hat{n}=1$	$\hat{n}=2$	$\hat{n}=3$	$\hat{n}=4$	$\hat{n}=5$	$\hat{n}=6$	$\hat{n}=7$
$m=20$	$m'=50$	0	0	0.01	0.519	0.322	0.115	0.034
	$m'=100$	0	0	0.001	0.693	0.258	0.04	0.008
	$m'=150$	0	0	0	0.765	0.207	0.025	0.003
	$m'=200$	0	0	0	0.812	0.168	0.019	0.001
	$m'=250$	0	0	0	0.87	0.124	0.004	0.002
	$m'=300$	0	0	0	0.902	0.093	0.005	0
	$m'=350$	0	0	0	0.929	0.069	0.001	0.001
	$m'=400$	0	0	0	0.948	0.052	0	0
	$m'=450$	0	0	0	0.96	0.04	0	0
	$m'=500$	0	0	0	0.963	0.037	0	0
	$m'=750$	0	0	0	0.986	0.014	0	0
	$m'=1000$	0	0	0	0.989	0.011	0	0
$m=50$	$m'=20$	0	0	0.012	0.253	0.456	0.196	0.083
	$m'=40$	0	0	0.003	0.237	0.461	0.215	0.084
	$m'=60$	0	0	0	0.311	0.457	0.153	0.079
	$m'=80$	0	0	0	0.378	0.442	0.126	0.054
	$m'=100$	0	0	0	0.443	0.421	0.093	0.043
	$m'=120$	0	0	0	0.513	0.37	0.082	0.035
	$m'=140$	0	0	0	0.515	0.382	0.08	0.023
	$m'=160$	0	0	0	0.552	0.343	0.09	0.015
	$m'=180$	0	0	0	0.586	0.352	0.052	0.01
	$m'=200$	0	0	0	0.644	0.285	0.06	0.011
	$m'=300$	0	0	0	0.696	0.277	0.027	0
	$m'=400$	0	0	0	0.75	0.233	0.017	0

Table C10: Algorithm 1 performance in Scenario 3 probabilistic valuations - Extended

		$\hat{v}_{opt}$	$\hat{\mathtt{v}}_{\mathtt{opt}}$`Error`	`WithinInt`	`WithinIntWide`
$m = 20$	$m' = 50$	38.836	2.362	67.1	88.2
	$m' = 100$	39.103	2.069	72.3	92.8
	$m' = 150$	38.799	2.166	74.9	95.6
	$m' = 200$	38.949	2.104	78.1	96.7
	$m' = 250$	39.003	2.068	75.9	96.2
	$m' = 300$	39.121	1.951	78.2	97
	$m' = 350$	38.89	2.043	80.6	98
	$m' = 400$	38.849	2.118	78.8	97.6
	$m' = 450$	38.936	1.996	82.2	98.4
	$m' = 500$	38.572	2.358	80.1	97.5
	$m' = 750$	38.751	2.167	78.8	98.1
	$m' = 1000$	38.847	2.161	80.6	98
$m = 50$	$m' = 20$	39.427	2.625	52.6	74.7
	$m' = 40$	39.177	2.243	59.5	83.1
	$m' = 60$	39.097	2.329	63.6	85.3
	$m' = 80$	39.325	2.019	65.9	88.6
	$m' = 100$	38.819	2.333	70.1	91.1
	$m' = 120$	38.933	2.257	69.5	92.1
	$m' = 140$	38.986	2.098	73.7	94.1
	$m' = 160$	38.847	2.175	73.8	93.5
	$m' = 180$	38.528	2.446	78	94.8
	$m' = 200$	38.717	2.31	76	94.9
	$m' = 300$	38.748	2.248	76.9	95.5
	$m' = 400$	38.539	2.399	81.1	96.9