Joint Assortment and Quality Consistent Discrete Pricing Under the Nested Logit Model

1 Introduction

1.1 Problem Introduction and Motivation

Discrete choice models have been used for several decades to capture central features of customer preferences. In particular, they provide a compact expression of how product demand changes as the offered assortment changes in composition or price. Some models, such as the basic attraction model, have been justified with axioms; see [14]. Alternatively, models like the multinomial logit (MNL) model have been justified with random utility theory; see [15]. This paper considers an important extension to the MNL model: the nested logit (NL) model introduced by [23]. Justifications and extensions for the NL model are provided in [5, 16]. The NL model was developed primarily to avoid the independence of irrelevant alternatives property suffered by the MNL model; see [4]. Under the NL model customers first select a nest and then a product within the selected nest. The nesting structure is designed to capture correlations among product utilities, where closely correlated products are in the same nest. A nest can be a particular product category, products with closely related features, or a distinct sales channel.

In this paper we study the joint assortment and quality consistent pricing problem where customer choice is governed by the NL model. A key piece of input to the problem is a quality ranking among the products where higher ranked items are higher quality. In addition to the quality ordering we are given a cost for each product, a discrete collection of prices at which products can be priced, and the parameters that govern the NL model. Our objective is to find an assortment of products to offer, and quality consistent prices to set, so that expected revenue is maximized. Prices are quality consistent when higher ranked items have higher prices. We assume that each item has some fixed cost per sale and that higher quality items have higher costs. We make only a mild assumption on how the parameters of the NL model depend on price; we require that the probability of a customer purchasing a product does not increase when the price of the product is increased.

The joint assortment and quality consistent pricing problem using the MNL model has been studied in [6]. In [6] they present a polynomial time algorithm that gives an optimal assortment and prices. Their technique uses a general result: any constrained assortment optimization problem under the MNL model where the constraints are totally unimodular can be modelled with a linear program. For assortment optimization problems under the NL model there is no such linear program. This greatly complicates assortment optimization problems under the NL model and, in particular, prevents us from applying the techniques of [6] to our problem. Additionally, the quality consistent constraint in the NL model does not appear to be captured by any totally unimodular matrix.

More generally, the joint assortment and quality consistent pricing problem has been studied under the MNL model in [6] and under the NL model in [9]. In [9] they reduce the joint assortment and pricing problem to a pure assortment problem and develop a framework for solving constrained assortment optimization under the NL model. They apply this general framework to simple constraints; quality consistent pricing constraints introduce complications that are outside this scope. Further, some quality rankings introduce constraints that do not fall naturally into the constrained assortment optimization framework of [9].

Practically, introducing quality consistent pricing constraints is motivated by substantial literature that focuses on the role of quality rankings in marketing. This literature typically refers to quality orderings as price ladders. Studies such as [3] use theoretical models to find optimal product line prices and, more specifically, how to construct optimal price ladders. There are several empirical studies that show marketers frequently use price ladders to signal the relative quality of items and that consumers respond to these signals; see([20, 21, 8, 17]). There have also been studies such as [12, 24] that show product pricing and product quality are not well correlated, indicating that price ladders are primarily a tool used by marketers and not the result of some exogenous constraint.

Quality rankings are a well used and intuitive tool that marketers use to manage revenue. We include quality rankings in our model to provide a "lever"; marketers can use the quality ranking, which is intuitive, to predictably influence the optimized solution. Including a lever in the model overcomes a potential shortcoming of using either the NL or MNL model to formulate optimization problems: the parameters of the model do not have a transparent influence on the final solution. Without a lever, if a marketer finds a solution unacceptable, for managerial reasons, then they have limited ability to massage the input to get a desirable solution. By including an intuitive lever we aim to overcome this difficulty.

1.2 Literature Review

The pure assortment optimization problem, where prices are fixed apriori, is a classic problem. Under the MNL model it is well known that the optimal assortment is revenue ordered ([22]) while under the NL model [7] recently showed the structure of the optimal assortment depends on the parameters of the model.

The joint assortment and pricing problem without a quality consistent constraint has also been studied. Under the MNL model (blah) show that the problem can be modelled as a convex optimization problem. More recently [6] has shown that the problem is captured by a non-trivial linear program. Under the NL model [9] gives a tractable algorithm to find an optimal assortment and prices. A common thread in all of these results is reducing the joint assortment and pricing problem to a pure assortment problem with a capacity constraint. This is done by creating copies of items, one item copy for every price level, and allowing only a single item copy to be offered. We take the same approach.

A related stream of work considers pricing problems where prices can take values in some continuous interval, but assumes that the parameters of the choice model have a strict functional dependence on product prices. When there are no price bounds then there is no assortment problem: it is optimal to offer all available products. Under the MNL model it has been shown that when all products share the same price sensitivities then there is some constant markup for all items; see [1, 2, 13, 10]. Under the NL model [11] show that there is a constant "adjusted markup" in each nest when there is product-differentiated price sensitivity parameters. When prices are bounded [18] shows that it is not necessarily optimal to offer all available products and considers the joint assortment and pricing problem under the NL model; they exhibit tractable algorithms to approximate the problem.

Quality consistent constraints have been studied in the context of optimization problems before. [19] introduces a quality ordering motivated by the automotive industry, where vehicles are offered at different trim levels that naturally imply a quality ordering. Additionally, [6] shows that the joint assortment and quality consistent pricing problem under the MNL model can be modelled as a linear program.

1.3 Paper Organization

We consider two different types of quality orderings. In Section 2 we consider quality orderings within nests: items that can be offered in the same nest are ordered with respect to quality, but products in different nests are not comparable. Since nests contain items with correlated utilities this is a natural set of constraints to consider. If, for example, a marketing team delegates each product category to a single marketeer then the individual marketeers will have a sense of the quality ranking for products in their category. However, the team as a whole makes no judgement about the relative quality of items across different categories.

In Section 3 we consider total quality orderings: every item available is part of the same quality ranking. We stipulate that the ranking of items in the same nest is contiguous; if some product is ranked between two products, both of which are in the same nest, then all three products must be in the same nest. This type of constraint is applicable when a single marketer manages an entire product line.

2 Joint Quality Consistent Pricing and Assortment Within Nests

2.1 Model

In the joint pricing and assortment problem we are given a collection of items, each of which can be priced at one of k distinct levels. Our objective is to choose a set of items to offer for sale, and prices for the offered items, so that expected revenue is maximized. The collection of items is decomposed into nests and, within each nest, we would like the prices of the offered items to be consistent with their quality.

More formally, the collection of items is partitioned into m nests indexed by $M = \{1, \dots, m\}$. A nest can represent a product category, a sales channel, or a retail location. Each nest contains n items indexed by $N = \{1, \dots, n\}$. This implies that there are mn total items that can be offered for sale. Each item can be priced at one of k discrete price levels indexed by $P = \{1, \dots, k\}$. We let item ij be the index of item j in nest i. Within each nest there is a quality ordering $>_q$ among items so that $ij >_q ij'$ implies that item ij is higher quality than item ij'. In this section we consider an ordering $>_q$ that is only defined within each nest and cannot be extended to all items. When two items ij and ij' are both offered for sale and $ij >_q ij'$ then, in order to be quality consistent, the price of item ij must be higher than the price of item ij'. To ease notation we let item index correspond to quality: item j has higher quality than item j' when $j > j'$. Similarly, we let price level index correspond to relative price: price level p is higher than price level p' when $p > p'$. We let the per-unit cost of item ij be c_{ij}. We assume higher quality items have higher cost, so that $c_{ij} \geq c_{ij'}$ when $j > j'$.

We immediately reduce the joint assortment and pricing problem to a pure assortment problem by introducing item copies. We create k copies of each item, one for each price level, and impose the constraint that at most one copy of each item can be offered. We let ijp be the index of item j in nest i priced at level p. For brevity will will refer to this as item copy ijp. When item copy ijp is offered we are implicitly offering item ij at price level p. We let d_p be the price of any item copy priced at level p. Note that this price is shared across items: any item priced at level p has a posted price of d_p. Recall that from our indexing convention $d_p > d_{p'}$ when $p > p'$. The net revenue we receive for item copy ijp is $r_{ijp} = d_p - c_{ij}$. Since c_{ij} is constant for item ij, r_{ijp} increases in price index when the item is fixed. In what follows we will explicitly distinguish items, which can have any price, from item copies, which correspond to an item offered at a fixed price level.

The probability of selling an item copy will be governed by the nested logit model. Suppose S is a set of item copies offered for sale. We treat S as a vector of index sets, $S = (S_1, \dots, S_m)$ where S_i is the set of indexes for item copies offered in nest i. Customers select an item to purchase from S in a two stage decision process. In the first stage a customer selects some nest to purchase from or chooses to purchase nothing and leaves the system. If the customer selects a nest then, in the second stage, he chooses an item copy to purchase from within the selected nest.

The probability of purchasing item copy ijp is related to a preference weight. We let v_{ijp} be the preference weight of item copy ijp and $V(S_i) = \sum_{jp \in S_i} v_{ijp}$ be the sum of preference weights of items in S_i. We require that the preference weight of ijp is inversely related to price: $v_{ijp} \geq v_{ijp'}$ when $p < p'$. This is sensible since, in most settings, when the same item is priced at a higher level it becomes less attractive to consumers. The probability of a customer selecting item copy ijp given that nest i was selected is $\frac{v_{ijp}}{V(S_i)}$. The expected revenue we receive given that the customer has selected nest i is

$$R_i(S_i) = \sum_{jp \in S_i} \frac{v_{ijp}}{V(S_i)} r_{ijp}.$$

When we offer assortment $S = (S_1, \dots, S_m)$ the probability of the customer selecting nest i is given by

$$P_i(S_1, \dots, S_m) = \frac{V_i(S_i)^{\gamma_i}}{v_0 + \sum_{l \in M} V_l(S_l)^{\gamma_l}}.$$

where v_0 is the preference weight of leaving the system and γ_i is a nest dissimilarity parameter for nest i. The expected revenue we obtain from each customer is

$$\Pi(S_1, \dots, S_m) = \sum_{i \in M} P_i(S_1, \dots, S_m) R_i(S_i).$$

Only a small collection of assortments $\mathcal{S}$ are feasible in our setting. Specifically, a set $S \in \mathcal{S}$ is feasible if each item has a unique price and prices are quality consistent. Because the quality ordering is defined only within nests we can decompose $\mathcal{S} = (\mathcal{S}_1 \times \dots \times \mathcal{S}_m)$. Here $\mathcal{S}_i$ is the collection of feasible assortments within nest i. To guarantee that we have valid prices in nest i at most one item copy $ijp \in S_i$ for every item ij. To guarantee quality consistent prices in nest i if $ijp \in S_i$ then $ij'p' \notin S_i$ for all items copies $ij'p'$ where $j' > j$ and $p' < p$; intuitively higher item index implies higher price index. We can write the assortment optimization problem succinctly as

$$\text{argmax}_{(S_1, \dots, S_m) \in \mathcal{S}} \Pi(S_1, \dots, S_m). \tag{1}$$

We can visualize any $S \in \mathcal{S}$ with a two dimensional array: the column dimension is item index and the row dimension is price index. We label columns with a double index: column ij corresponds to item ij. The columns are ordered so that columns corresponding to nest i come before all columns corresponding to nest $i+1$. Additionally, among columns with the same nest index, items with lower item index come earlier in the ordering. Rows are ordered by increasing price index. Item copy ijp corresponds to the entry in column ij and row p. To visualize S we put a mark in the cell in column ij, row p when item copy $ijp \in S$. See Figure 1.

row/column	$1,1$	$1,2$	$1,3$	$1,4$	$2,1$	$2,2$	$2,3$	$2,4$	$3,1$	$3,2$	$3,3$	$3,4$
1					$*$				$*$	$*$		
2	$*$	$*$										
3											$*$	
4				$*$								$*$
5							$*$	$*$				

Figure 1: Array representation of a feasible set where $m = 3$, $n = 4$, $k = 5$.

The array highlights two important features of feasible assortments. First, within each nest, as item index increased so does price index. This is the key feature of quality consistent prices. Second, there is no interaction between the price levels between nests; the price levels in a nest do not put any restrictions on the price levels in other nests. This lack of interaction is crucial for what follows and allows us to stitch together a full assortment from partial assortments in each nest.

2.2 Solution Concept

Although we have reduced the original joint pricing and assortment problem to a pure assortment problem we will need to further reduce the assortment problem to a parametric longest path problem. This further reduction relies heavily on the following result that appears in [9].

Theorem 1. *Let $S^* = (S_1^*, \dots, S_m^*) \in \mathcal{S}$ be an optimal solution to problem 1 with objective value Z^*. We have*

$$S^* = \operatorname{argmax}_{S=(S_1,\dots,S_m)\in\mathcal{S}} \left\{ \sum_{i\in M} \left[V(S_i)(R(S_i) - \gamma_i Z^* - (1-\gamma_i)R_i(S_i^*) \right] \right\}.$$

Further, if there exists $\mathcal{S}_i$'s so that $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_m$ then

$$S_i^* = \operatorname{argmax}_{S_i\in\mathcal{S}_i} \left\{ V(S_i)(R(S_i) - \gamma_i Z^* - (1-\gamma_i)R_i(S_i^*) \right\}.$$

A requirement for Theorem 1 to hold is that S_i contains the empty set for every nest i; this is automatically satisfied in the models we consider in this paper and so we leave it implicit in the statement of the theorem.

Theorem 1 characterizes optimal assortments in a way that we can leverage. Specifically, we will find a collection of assortments that is small and use the characterization in Theorem 1 to guarantee that the collection contains an optimal assortment. We leverage Theorem 1 by manipulating the argmax.

$$\begin{aligned}
S_i^* &= \operatorname{argmax}_{S_i\in\mathcal{S}_i} \{V(S_i)(R(S_i) - \gamma_i Z^* - (1-\gamma_i)R_i(S_i^*))\} \\
&= \operatorname{argmax}_{S_i\in\mathcal{S}_i} \left\{ \sum_{jp\in S_i} v_{ijp}r_{ijp} - \Big(\sum_{jp\in S_i} v_{ijp}\Big)\Big(\gamma_i Z^* + (1-\gamma_i)R_i(S_i^*)\Big) \right\} \\
&= \operatorname{argmax}_{S_i\in\mathcal{S}_i} \left\{ \sum_{jp\in S_i} v_{ijp}r_{ijp} - \Big(\sum_{jp\in S_i} v_{ijp}\Big)u_i^* \right\} \\
&= \operatorname{argmax}_{S_i\in\mathcal{S}_i} \left\{ \sum_{jp\in S_i} v_{ijp}(r_{ijp} - u_i^*) \right\} \qquad (2)
\end{aligned}$$

where $u_i^* = \gamma_i Z^* + (1-\gamma_i)R_i(S_i^*)$. Since u_i^* depends on the optimal solution we do not know its value. However, if we have a collection Q_i that contains, for every value of u_i^*, the argmax in (2) and, additionally, Q_i is small and computable then we do not need access to u_i^* to find the optimal solution. Rather, we can enumerate the objective value for each set in Q_i and find the optimal solution. To ease the exposition we define the function $a(S_i, u_i) = \sum_{jp\in S_i} v_{ijp}(p_{ijp} - u_i)$.

To stitch together the full assortment we use [7]. Since the collection Q_i contains an optimal assortment for nest i, in the language of [7], it is a collection of candidate assortments from which the optimal solution can be recovered. For completeness we exhibit the linear program that finds the optimal objective value:

$$\begin{aligned}
&\min x \\
&v_0 x \geq \sum_{i\in M} y_i, \\
&y_i \geq V(S_i)^{\gamma_i}(R(S_i) - x), \qquad \forall S_i \in Q_i.
\end{aligned}$$

In the above LP it is not hard to see that the first constraint is tight. At least one of the second type of constraints is also tight for each nest i. The assortment corresponding to a tight constraint for nest i is the optimal assortment within that nest. The complete optimal solution is the union of these.

Since Q_i is small and computable we can construct the above LP efficiently. Since Q_i is small the number of constraints is small and so we can also solve the LP efficiently. This gives us an efficient algorithm for finding the optimal solution.

2.3 Computing $\mathbf{Q_i}$

To find Q_i we will create a directed acyclic graph where arc length is controlled by an arc length parameter u_i and, when u_i is fixed, the length of an s-t path is $a(S_i, u_i)$ for some feasible S_i. Since we are focused on a single nest we will drop the nest index i in the remainder of this section. The network consists of $(n+1)k$ internal nodes, a source node s, and a sink node t. The internal nodes form a grid with $n+1$ "columns" and k "rows". We highlight this matrix structure by referring to nodes in matrix notation; node (j,p) has a column index j that corresponds to the item index and a row index p that corresponds to price index. Every internal node (j,p) in columns 1 through n has two outgoing arcs to node $(j+1,p)$. One of these arcs has length 0 and we refer to these as zero arcs. The other arc has length $v_{jp}(r_{jp} - u)$ where u is a parameter; we refer to these as non-zero arcs, even if for some some value of u the arc has length 0. Additionally, there is an arc of length 0 from (j, p) to $(j, p+1)$ for all j and p. Finally, there is an arc of length 0 from s to $(1, 1)$ and from (n, k) to t. See Figure 2.

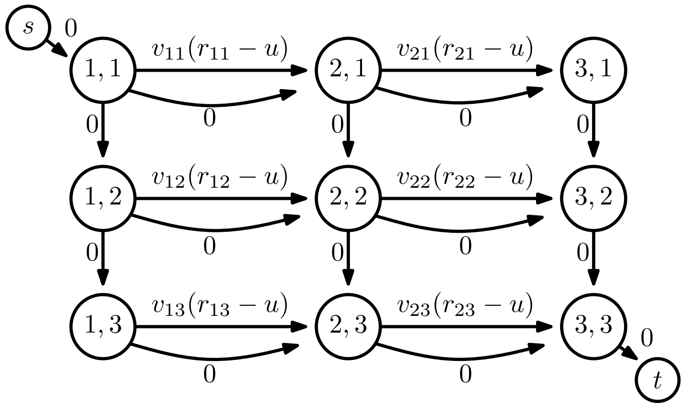

Figure 2: An example network where $n = 2$ and $k = 2$.

We treat an s-t path in this network, say q, as a set of arcs. Any arc $((j, p), (j + 1, p))$ is said to be in column j. It is easy to see that q contains exactly one arc in every column. If arc a in q is also in column j we say that a traverses column j.

There is a one-to-one correspondence between s-t paths in this network and feasible assortments in $\mathcal{S}_i$. A path q can be mapped to a set of item copies S_i: if a non-zero arc $((j, p), (j + 1, p)) \in q$ then item copy $jp \in S_i$. Note that, using this mapping, if $((j, p), (j + 1, p)) \in q$ is a zero arc then item copy $jp \notin S_i$. Intuitively this mapping produces a feasible assortment because any s-t path only moves "down" as it moves "to the right". This corresponds, via the above mapping, to items with higher indexes having higher prices. We make this more formal in the following lemma.

Lemma 2. *Every s-t path in the network maps to a feasible set in $\mathcal{S}_i$.*

Proof. Let q be an s-t path and let S_i be generated from the mapping above. To show that $S_i \in \mathcal{S}_i$ we need to verify that at most one item copy is offered for every item and that the offered copies are quality consistent.

First we show that there is at most one item copy for every item. Consider item j. Every item copy in S_i that corresponds to item j must be associated with an arc that traverses column j. We can show that there is exactly one arc traversing column j by considering the cut C that contains s and all nodes in columns 1 through j. All arcs crossing C are directed outwards and so q must contain exactly one of these arcs. This arc can either be a non-zero arc, in which case S_i contains one copy corresponding to item j, or a zero arc, in which case S_i does not include a copy corresponding to item j.

To see that quality consistency holds consider item j and suppose that item copy $jp \in S_i$. From the mapping this implies arc $a = ((j, p), (j + 1, p)) \in q$. Now consider the cut C that contains s and rows 1 through $p - 1$. Again, all arcs are directed out of this cut and q contains exactly one arc that crosses it. Further, since $a \in q$ the arc in q that crosses C must be in some column before column j. If quality consistency were violated then there would be an item copy $ij'p' \in S_i$ with $j' > j$ and $p' < p$. For this to happen q would need to include an arc at some row above p in a column after j and, as a result, q would cross cut C in some column after j. This cannot happen and so quality consistency holds. $\square$

Mapping a feasible assortment S_i to an s-t path q uses the inverse of the above mapping: $((j,p),(j+1,p)) \in q$ when $jp \in S_i$. Because increasing item index implies increasing price index there is some set of arcs we can add to this to produce a valid s-t path.

Because there is a one-one correspondence between s-t paths and feasible assortments we will abuse notation and let Q_i refer to both a collection of assortments and a collection of s-t paths. Since there is a one-to-one correspondence between s-t paths and feasible assortments in nest i, it is trivial to see that longest paths in the network map to the assortments that maximize (2). Recall that our objective is to show that Q_i, a collection that contains the the argmax in (2) for every value of u_i^*, is small and computable. When we consider Q_i as a collection of paths our objective is to show that the number of longest paths in the above network is small across all values of u and that we can find the set of longest paths.

To show that Q_i is small we will need to introduce row indices. Notice that the arc from column j to $j+1$ in q is "horizontal": the row index of its endpoints are the same. We call this the row index of arc j in path q and denote it by $row_q(j)$. We will also need to select a unique s-t longest path for every value of u. We order the paths lexicographically, where the j^{th} component of a path is $row_q(j)$. Whenever there are multiple paths of the same length for some value of u we will select the lexicographically largest.

Longest s-t paths in this network have a particular structure. We will show that if the arc that traverses column j is a zero arc then the arcs that traverse columns $j+1,\dots,m$ are also zero arcs. Further, all zero arcs are necessarily in the k^{th} row.

Lemma 3. *Let q be a longest s-t path for parameter u. Suppose the arc in q that traverses column j is a zero arc. Then the arcs traversing columns $j+1,\dots,m$ are also zero arcs. Further, all zero arcs in q are in the k^{th} row.*

Proof. We prove the first part of the argument by showing that if the arc traversing column j is a zero arc then the arc traversing column $j+1$ is a zero arc. We can then use inductive reasoning to show the claim.

Suppose, for contradiction, that the arc in q that traverses column j is a zero arc and the arc traversing column $j+1$ is a non-zero arc. Let $l = row_q(j+1)$. Since q is a longest path the arc traversing column $j+1$ must have positive length and $r_{j+1,l} - u > 0$. Since costs are increasing with item index we have

$$r_{j,l} - u = d_l - c_j - u \geq d_l - c_{j+1} - u = r_{j+1,l} - u > 0$$

implying $v_{j,l}(r_{j,l} - u) > 0$. Since the row index of the the arc traversing column j is at most l, we can swap the zero arc traversing column j in q for the zero arc in row l. This swap would create a path longer than q, so q is not a longest path. Contradiction.

The second part of the lemma is direct consequence of the first part. All of the zero arcs form a contiguous sub-path of q that ends at t. Since we assume we have the lexicographically largest path all of these zero arcs are in the k^{th} row. $\square$

The following lemma is the first time we use our assumption that preference weight is increasing in to price index.

Lemma 4. *Consider any node v in column c. Let q be the longest v-t path when the arc length parameter is u and q' be the longest v-t path when the arc length parameter is $u' > u$. We have that $row_q(j) \leq row_{q'}(j)$ for $j = c,\dots,n$.*

Proof. Suppose for contradiction that there is some arc where the inequality does not hold. Let j be the first column where $row_q(j) > row_{q'}(j)$ and let j' be the first column after j where $row_q(j') \leq row_{q'}(j')$. Let $\tilde{q}$ and $\tilde{q}'$ be the respective subpaths of q and q' between columns j and j'. From these definitions it is possible that $row_{q'}(j') = k$, though all other row indices must be strictly less than k. To ease exposition we will assume without loss of generality that $row_q(j') < k$.

From Lemma 3 all arcs in $\tilde{q}$ and $\tilde{q}'$ are positive. Combining this with our assumption that v_{jp} is decreasing in p we get

$$\sum_{jp\in\tilde{q}'} v_{jp} \geq \sum_{jp\in\tilde{q}} v_{jp}$$

We can swap $\tilde{q}'$ for $\tilde{q}$ in path q to construct a new path in the network. Similarly, we can swap

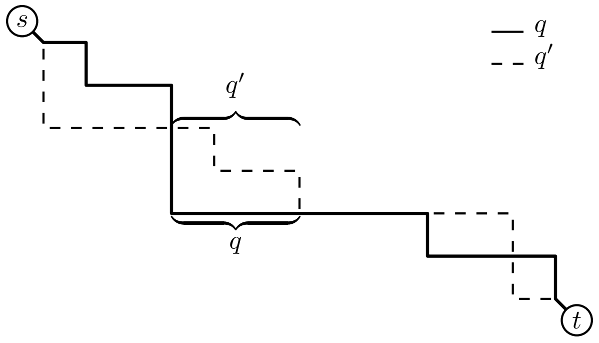

Figure 3: An example network where $n = 2$ and $k = 2$.

$\tilde{q}$ for $\tilde{q}'$ in q'. See Figure 3. Since q and q' are the longest paths for their respective values of u making these swaps does not increase path length. We can conclude that under parameter u, $\tilde{q}$ is longer than $\tilde{q}'$ and under parameter u', $\tilde{q}'$ is longer than $\tilde{q}$. Hence

$$\sum_{jp\in\tilde{q}'} v_{jp}r_{jp} - u' \sum_{jp\in\tilde{q}'} v_{jp} > \sum_{jp\in\tilde{q}} v_{jp}r_{jp} - u' \sum_{jp\in\tilde{q}} v_{jp}$$

$$\sum_{jp\in\tilde{q}} v_{jp}r_{jp} - u \sum_{jp\in\tilde{q}} v_{jp} > \sum_{jp\in\tilde{q}'} v_{jp}r_{jp} - u \sum_{jp\in\tilde{q}'} v_{jp}$$

Adding these inequalities gives

$$u' \sum_{jp\in\tilde{q}'} v_{jp} + u \sum_{jp\in\tilde{q}} v_{jp} < u' \sum_{jp\in\tilde{q}} v_{jp} + u \sum_{jp\in\tilde{q}'} v_{jp}$$
$$u' \sum_{jp\in\tilde{q}'} v_{jp} - u \sum_{jp\in\tilde{q}'} v_{jp} < u' \sum_{jp\in\tilde{q}} v_{jp} - u \sum_{jp\in\tilde{q}} v_{jp}$$

which implies

$$\sum_{jp\in\tilde{q}'} v_{jp} < \sum_{jp\in\tilde{q}} v_{jp}.$$

This is a contradiction. □

Lemma 5. *For any $v = (r, c)$, there are at most nk longest v-t paths over all values of u.*

Proof. We define a potential function on v-t paths $f(q) = \sum_{c \leq j \leq n} row_q(j)$. $f(q)$ is the sum of row indexes in the path q. We have $\max_q f(q) = kn$ and $\min_q f(q) \geq 0$. From Lemma 4, if an increase in u precipitates a change in longest path, say from q to q', then some row index must decrease. Hence $f(q') \leq f(q) - 1$. Over all values of u there can be at most nk such changes and hence nk possible paths. □

To compute Q_i we extend our notation from above and let $Q_i(j,p)$ be the set of longest paths from node (j,p) to t for all values of u. From Lemma 5 $Q_i(j,p)$ is always polynomially sized. We will use a dynamic program to find $Q_i(j,p)$ for all (j,p). The base case for the dynamic program is when $(j,p) = (n,k)$: we have $Q_i(n,k) = \{((n,k),(n+1,k))\}$. The inductive step requires an aggregation step and a thinning step.

In the aggregation step we find a set of candidate paths $C(j,p) \supset Q_i(j,p)$. For any value of u the longest path from (j,p) to t must first go either "down" or "to the right" and so must include either arc $((j,p),(j+1,p))$ or $((j,p),(j,p+1))$. Corresponding to each alternative, $C(j,p)$ includes paths in $Q_i(j+1,p)$ prefixed with arc $((j,p),(j+1,p))$ and paths in $Q_i(j,p+1)$ prefixed with $((j,p),(j,p+1))$. It is easy to see that $C(j,p)$ must be a superset of $Q_i(j,p)$. Since, from Lemma 5 $|Q_i(j+1,p)| \leq nk$ and $|Q_i(j,p+1)| \leq nk$ we get $C(j,p) \leq 2nk$.

If this was the only step of the dynamic program then $C(j,p)$ would include all possible paths in the network from node (j,p) to t and would grow exponentially. To combat this we introduce a thinning that eliminates paths in $C(j,p)$ that are not longest paths for any value of u.

Notice that the length of any path $q \in C(j,p)$ is a linear function of u. Let $q(u)$ be the length of path q as a function of u. We will find the longest paths for all values of u by enumerating all of these functions, finding the pairwise intersections between them, and considering only the largest $q(u)$ at any intersection point. See Figure 4. We outline this procedure more precisely.

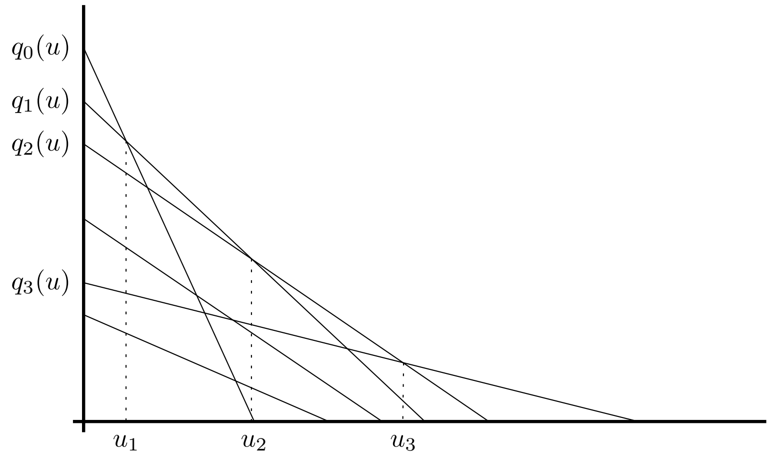

Figure 4: A pictorial representation of the thinning step. The procedure traces the upper envelope.

We will iteratively find longest s-t paths $q_0, \ldots, q_l$ and values of u, $u_0, \ldots, u_l$ where q_i is the longest path for $u \in [u_i, u_{i+1})$. The smallest value of u we consider is $u_0 = 0$. When $u = u_0$ we can find the longest path in $C(j,p)$ (and hence the longest path in $Q_i(j,p)$) by enumeration. To find u_1 we find the intersection of $q_0(u)$ (as a linear function of u) with $q(u)$ for all $q \in C(j,p)$. The smallest non-zero intersection point is u_1. The function $q_1(\cdot)$ where $q_1(u_1) = q_0(u_1)$ corresponds to the longest path for values of u just greater than u_1.

To find all longest paths we apply this procedure iteratively. However, since we have determined the longest paths for values up to u_1 we treat u_1 as a cut off value: as we apply this process iteratively we ignore intersections that happen for values of $u < u_1$. By applying the steps iteratively until there are no more intersections we construct all of $Q(j,p)$. By Lemma 5, $|Q(j,p)| \leq nk$.

3 Quality Consistent Pricing Across Nests

In this section we extend the quality ordering $>_q$ to be a total ordering among the items. We have $ij >_q i'j'$ when either $i > i'$ or, $i = i'$ and $j > j'$: item ij is higher quality if it has either a higher nest index or, if the nest index is the same, has a higher item index.

We will use the same reduction to a pure assortment optimization problem that we used in the previous setting. The notation from the previous setting remains largely unchanged. Feasible assortments $S \in \mathcal{S}$ are somewhat different though. To ensure unique prices $ijp \in S$ implies $ijp' \notin S$ for all p'. To ensure quality consistent prices, whenever $ijp \in S$ then $i'j'p' \notin S$ for all $i'j'p'$ where either $i' > i$ and $p' < p$ or $i = i'$, $j' > j$ and $p' < p$. We can still characterize the assortment optimization problem succinctly as

$$\text{argmax}_{(S_1,\ldots,S_m)\in\mathcal{S}}\Pi(S_1,\ldots,S_m). \tag{1}$$

Our starting point is again Theorem 1. However, $\mathcal{S}$ does not decompose by nest so we can only use the first part of the Theorem. We can use manipulations, similar to those used for (2), to get

$$S^* = \text{argmax}_{S=(S_1,\ldots,S_m)\in\mathcal{S}} \left\{ \sum_{i\in M} \sum_{jp\in S_i} v_{ijp}(r_{ijp} - u_i) \right\}$$

where $u_i = \gamma_i Z^* + (1-\gamma_i)R_i(S_i^*)$.

From the above expression we can see a close similarity to (2). We can offer a broad view of how the techniques we use in this section compare to the techniques of section 2. The differences are easily identified by visualizing $S \in \mathcal{S}$ in an array, as we did previously. See Figure 5.

We can observe three key features when we consider $S \in \mathcal{S}$ as an array. First, between each pair of nests, say nest i and $i+1$, there is at least one price level that is both a lower bound on price levels in nest i and an upper bound on price levels in nest $i+1$. We call this price level a splitting price index between nest i and $i+1$. Second, the price levels in any nest are bounded between two splitting price indexes, except nest 1, which has no lower bound, and nest m, which has no upper bound. Finally, the splitting price indexes form an increasing sequence. A starting point for the techniques of this section re-characterize $\mathcal{S}$ using these properties.

The primary difficulty we face in this section is that, unlike the previous section, the price levels between nests are coupled: the price levels in nest i are an upper bound on the price levels in nest $i+1$. This coupling between nests prevents using the methods of [7] to stitch together a complete assortment.

However, if we knew splitting price indexes from an optimal solution apriori then we could recover the optimal assortment. We can restrict price levels in each nest to be between the lower and upper bounds determined by splitting price indexes. By doing this we can make pricing decisions in a nest independent of pricing decisions in other nests. Relating this to Section 2, placing bounds on the price levels in nest i manifests itself as truncating the rows of the network used to find Q_i. We can use this modified network, as before, to find a collection that contains the

row/column	$1,1$	$1,2$	$1,3$	$1,4$	$2,1$	$2,2$	$2,3$	$2,4$	$3,1$	$3,2$	$3,3$	$3,4$
1	$*$											
2		$*$										
3				$*$								
4												
5					$*$							
6								$*$	$*$	$*$		
7											$*$	
8												

Figure 5: Array representation of a feasible set where $m = 3$, $n = 4$, $k = 8$.

optimal assortment and price levels in nest i, provided we know the splitting prices apriori. We can use the same arguments of the previous section to show that this collection is small.

Unfortunately we cannot determine the splitting price indexes of an optimal solution apriori. Instead, we must expand the collection we consider in each nest: for each nest we will find a collection that contains the optimal assortment and price levels for all pairs of lower and upper bounds. There are a small number of pairs and the size of the collection will only increase polynomially.

Finding this collection for every nest will be a preprocessing step for a stitching algorithm that finds the optimal combination of assortments across nests. The stitching algorithm uses the third observation gleaned from the array: the splitting price indexes form an increasing sequence. We will perform a search over all increasing sequences of prices to find the one that gives us the optimal solution.

3.1 Re-characterizing $\mathcal{S}$

We make these ideas more formal by re-characterising $\mathcal{S}$. Decomposing $\mathcal{S}$ by nests directly, as in the previous setting, is not possible. An observation from the array visualization is that in each nest the price indexes were within a lower and upper bound. Correspondingly, we let $\mathcal{S}_i(p_l, p_u)$ be the collection of all assortments for nest i where the lowest price index among offered item copies is at least p_l and the highest price index is at most p_u. For a feasible assortment $S = (S_1, \ldots, S_m)$ each S_i is in $\mathcal{S}_i(p_l, p_u)$ for some pair p_l, p_u.

We also observed that there was a splitting price index between each pair of adjacent nests. Formally, the splitting price index for nests i and $i+1$ is a price index p_2 such that $S_i \in \mathcal{S}_i(p_1, p_2)$ and $S_{i+1} \in \mathcal{S}_{i+1}(p_2, p_3)$ for some p_1 and p_3. The splitting price index between nests i and $i+1$ can be the highest price index in nest i, the lowest price index in nest $i+1$, or any price in between these two indexes. In what follows we will fix the splitting price index between nests i and $i+1$ to be the highest price index in nest i.

Our final observation from the array visualization was that the splitting price indexes form a non-decreasing sequence. The highest price index in nests $1, \ldots, m$, our splitting price indexes, form a length m non-decreasing sequence σ. The i^{th} entry of σ, $\sigma(i)$, is the highest price index in nest i. We will see shortly that we can ease notation by considering length $m+1$ non-decreasing sequences where $\sigma(m+1) = k$; the $m+1^{st}$ entry behaves as a dummy upper bound on price index in nest m. Let Σ be the set of all length $m+1$ non-decreasing price sequences where $\sigma(m+1) = k$.

We can now fully re-characterize $\mathcal{S}$. To construct a set $S \in \mathcal{S}$ we first fix a sequence $\sigma \in \Sigma$ and then, for every nest i, we choose a set $S_i \in \mathcal{S}_i(\sigma(i), \sigma(i+1))$. Note that although $\sigma(m+1) = k$ is fixed there is no restriction on the highest price index in nest m; k behaves as only an upper bound in $S_m(\sigma(m), k)$. With this re-characterization in hand we can write

$$\begin{aligned} S^* &= \operatorname{argmax}_{\substack{\sigma\in\Sigma \\ S_i\in\mathcal{S}_i(\sigma(i),\sigma(i+1))}} \left\{ \sum_{i\in M} \sum_{jp\in S_i} v_{ijp}(r_{ijp} - u_i) \right\} \\ &= \operatorname{argmax}_{\sigma\in\Sigma} \left\{ \sum_{i\in M} \max_{S_i\in\mathcal{S}_i(\sigma(i),\sigma(i+1))} \sum_{jp\in S_i} v_{ijp}(r_{ijp} - u_i) \right\} \end{aligned}$$

where $u_i = \gamma_i Z^* + (1-\gamma_i)R_i(S_i^*)$.

The above expression for S^* suggests an approach to our problem. First, for all values of u_i and all pairs (p_l, p_u) we solve the inner maximization over $\mathcal{S}_i(p_l, p_u)$ for each nest individually. This generates a collection of candidate assortments. Then we can stitch together these candidate assortments by maximizing over Σ. We take this approach in the following two subsections.

3.2 Inner Maximization: Finding $\mathbf{Q_i(p_l, p_u)}$

We will solve the inner maximization by identifying a set $Q_i(p_l, p_u)$ that is small and contains the inner max for any value of u_i when the lowest price index is at least p_l and the highest price index is at most p_u. We then have

$$S^* = \operatorname{argmax}_{\sigma\in\Sigma} \left\{ \sum_{i\in M} \max_{S_i\in Q_i(\sigma(i),\sigma(i+1))} \sum_{jp\in S_i} v_{ijp}(r_{ijp} - u_i) \right\} \tag{3}$$

To find $Q_i(p_l, p_u)$ for fixed i, p_l, and p_u we will use a truncated version of the network we used to find Q_i in the previous setting. Instead of $(n+1)k$ internal nodes there are only $(n+1)(p_u - p_l)$ internal nodes. Otherwise the structure is identical. Every internal node (j, p) in columns 1 through n has two outgoing arcs to node $(j+1, p)$. One of these arcs is length 0 and the other is length $v_{ijp}(r_{ijp} - u)$ where u is a parameter. The arc from s to $(1, p_l)$ has length 0, as does the arc from (n, p_u) to t. See Figure 6.

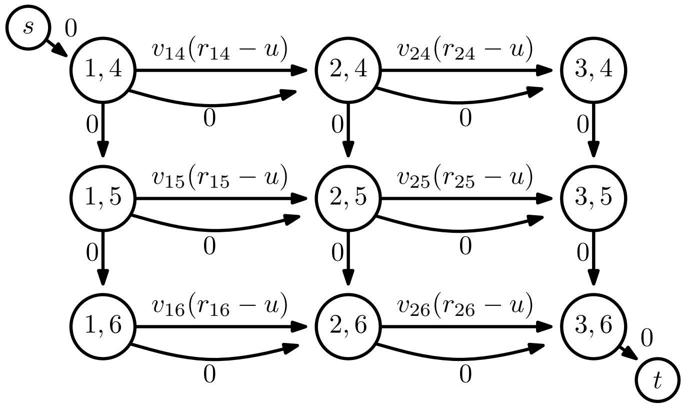

Figure 6: An example network where $n = 2$ and $p_l = 4$ and $p_u = 6$.

We can repeat the arguments from the previous setting to argue that this network has the same properties. Any s-t path in this network maps to a feasible assortments in nest i where the bounds on price indexes are p_l and p_u. Conversely, feasible assortments map to s-t paths. There are also a polynomial number of paths in $Q_i(p_l, p_u)$ and the same dynamic program can be used to construct $Q_i(p_l, p_u)$.

3.3 Stitching Nests Together: Outer Maximization for Fixed z

Unfortunately we do not know u_i and cannot make direct use of (3) to stitch together the optimal solution. We take a slightly different strategy. We let $\mathcal{Q}$ contain all feasible assortments that result from stitching together sets in the $Q_i(p_l, p_u)$'s. From (3) the collection $\mathcal{Q}$ contains the optimal assortment. Using the notation of (1)

$$Z^* = \max_{(S_1,\dots,S_m)\in\mathcal{Q}} \Pi(S_1, \dots, S_m).$$

We need to manipulate this expression. For some fixed set $\bar{S}$ and $\bar{Z} = \Pi(\bar{S}_1, \dots, \bar{S}_m)$ we have

$$\begin{aligned}
\bar{Z} &= \Pi(\bar{S}_1, \dots, \bar{S}_m) \\
\bar{Z} &= \sum_{i\in M} \frac{V_i(\bar{S}_i)^{\gamma_i}}{v_0 + \sum_{l\in M} V_l(\bar{S}_l)^{\gamma_l}} R_i(\bar{S}_i) \\
\bar{Z}\Big(v_0 + \sum_{l\in M} V_l(\bar{S}_l)^{\gamma_l}\Big) &= \sum_{i\in M} V_i(\bar{S}_i)^{\gamma_i} R_i(\bar{S}_i) \\
v_0 \bar{Z} &= \sum_{i\in M} V_i(\bar{S}_i)^{\gamma_i} R_i(\bar{S}_i) - \bar{Z} \sum_{i\in M} V_i(\bar{S}_i)^{\gamma_i} \\
v_0 \bar{Z} &= \sum_{i\in M} V_i(\bar{S}_i)^{\gamma_i} \Big(R_i(\bar{S}_i) - \bar{Z}\Big).
\end{aligned}$$

Since this is true for any set $\bar{S}$ we get

$$\begin{aligned}
v_0 Z^* &= \max_{(S_1,\dots,S_m)\in\mathcal{Q}} \sum_{i\in M} V(S_i)^{\gamma_i}(R(S_i) - Z^*) \\
&= \max_{\sigma\in\Sigma} \sum_{i\in M} \max_{S_i\in Q_i(\sigma(i),\sigma(i+1))} V(S_i)^{\gamma_i}(R(S_i) - Z^*)
\end{aligned}$$

where the second equality follows from the definition of $\mathcal{Q}$. For notational convenience we define

$$f_i(p_l, p_u, z) = \max_{S_i\in Q_i(p_l,p_u)} V(S_i)^{\gamma_i}(R(S_i) - z) \qquad F(z) = \max_{\sigma\in\Sigma} \sum_{i\in M} f_i(\sigma(i), \sigma(i+1), z).$$

Since $Q_i(p_l, p_u)$ is small we can compute $f_i(p_l, p_u, z)$ for a fixed (p_l, p_u) pair and fixed z. The remainder of this subsection will exhibit a technique to compute $F(z)$ for fixed z.

With this notation the above equality becomes

$$v_0 Z^* = F(Z^*) \tag{4}$$

We can find Z^* by using (4) and a binary search. Notice that binary search was not a possibility in (3) since u_i was a function of Z^* and an unknown $R(S_i^*)$. We defer the exact details of the binary search.

We compute $F(z)$ with a network, similar in spirit to the network of the previous section. The objective is to construct a network that, for fixed z, allows us to map every $\sigma \in \Sigma$ to an s-t path of length $\sum_{i\in M} f_i(\sigma(i),\sigma(i+1),z)$. As a preprocessing step we will need to compute $f_i(p_l,p_u,z)$ for all i and (p_l,p_u) pairs. We use the network of the previous section to compute $f_i(p_l,p_u,z)$ for a given pair and, since there are a small number of pairs, this preprocessing is tractable. Our objective, after constructing the network, will be to find a longest path. We will show that a longest path in the network has value $F(z)$.

In the network there are mk internal nodes, a start node s, and an end node t. To ease discussion we alternately refer to node t as node $(m+1,k)$. Intuitively, if (i,p) is included in a longest path then, in the assortment that achieves value $F(z)$, the lowest price index in nest i is p. For node (i,p) we refer to i as the nest index and p as the price index. When $i<m$ there is an arc from node (i,p) to $(i+1,p')$ for all $p'\geq p$. This arc has length $f_i(p,p',z)$. Intuitively, when a longest path traverses arc $((i,p),(i+1,p'))$ the lowest price index in nest i is p and the highest price index in nest i is p'. As a result nest i contributes exactly $f_i(p,p',z)$ to $F(z)$; we mirror this and force arc $((i,p),(i+1,p'))$ to contribute $f_i(p,p',z)$ to the length of the longest path. When $i=m$ there is only a single arc from (i,p) to $(i+1,k)=t$. This arc has length $f_m(p,k,z)$. These arcs are distinguished because there is no upper bound on price index in nest m and we artificially push the upper bound to k. There are also 0 length arcs from s to every node in column 1. These arcs do not correspond to any nest but act to set the highest price index in nest 1. See Figure 7.

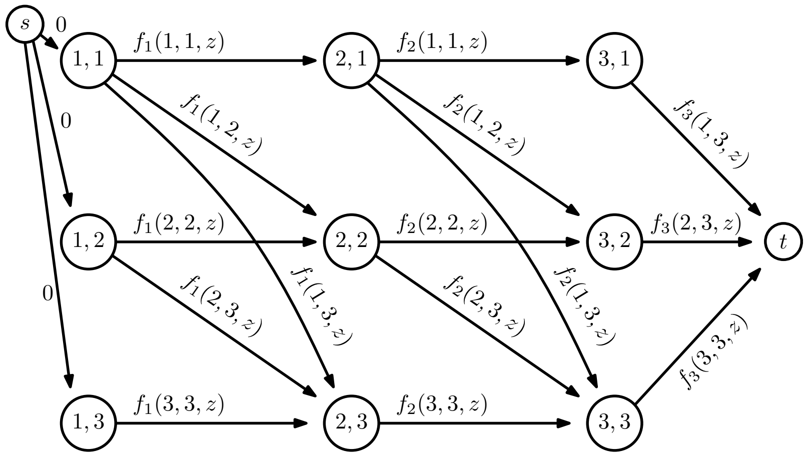

Figure 7: An example network where $m=3$ and $k=3$.

For an s-t path q there is a unique node traversed by q in every column. We let $q(i)$ be the price index of this node and we let $q(m+1)=k$. Conversely, given exactly one node in each column there is a unique s-t path that traverses these nodes (if there is a feasible path).

There is a one-to-one correspondence between $\sigma\in\Sigma$ and s-t paths. Given a σ we can map it to the unique s-t path that contains nodes $(i,\sigma(i))$ for all i. We map q to a σ by letting $\sigma(i)=q(i)$. Note that if q maps to σ then they have the same objective value

$$\sum_{i\in M} f_i(q(i),q(i+1),z)=\sum_{i\in M} f_i(\sigma(i),\sigma(i+1),z)$$

Lemma 6. *Fix z. Computing a longest s-t path in this network is equivalent to finding $F(z)$.*

Proof. Let q^* be the length of the longest path in the above network. Since there is a one-to-one mapping between s-t paths and $\sigma \in \Sigma$ we get

$$\begin{aligned}\sum_{i\in M} f_i(q^*(i), q^*(i+1), z) &= \max_q \sum_{i\in M} f_i(q(i), q(i+1), z)\\ &= \max_{\sigma\in\Sigma} \sum_{i\in M} f_i(\sigma(i), \sigma(i+1), z)\\ &= F(z)\end{aligned}$$

□

3.4 Binary Search: Finding $\mathbf{Z}^*$

In the previous two sections we developed the tools required to perform a binary search. We have characterized Z^* as a fixed point solution to (4). We have found a small and computable $Q_i(p_l, p_u)$ effectively allowing us to calculate $\sum_{i\in M} f_i(\sigma(i), \sigma(i+1), z)$ for fixed σ and z. Finally, we have exhibited a method to find $F(z)$ for fixed z.

A naive approach for binary search would use (4) by picking an iterate, say z_1, computing $F(z_1)$, and comparing this with $v_0 z_1$. If $v_0 z_1 < F(z_1)$ we pick a larger value of z and re-compute. If $v_0 z_1 > F(z_1)$ then we pick a smaller value and re-compute.

We take a somewhat more sophisticated approach by exploiting the structure of $F(z)$. Recalling that $\mathcal{Q}$ is a collection containing the optimal assortment, and the definition of $f_i(p_l, p_u, z)$ we can see that $F(z)$ can be written as a maximization over sets

$$F(z) = \max_{\sigma\in\Sigma} \sum_{i\in M} f_i(\sigma(i), \sigma(i+1), z) = \max_{S\in\mathcal{Q}} \sum_{i\in M} V(S_i)^{\gamma_i}(R(S_i) - z)$$

For fixed $S = (S_1, \ldots, S_m)$ the function $\sum_{i\in M} V(S_i)^{\gamma_i}(R(S_i) - z)$ is a linear function of z. From this observation it is trivial to see that that $F(z)$ is a piecewise linear function that is convex in z, since it is a maximum of linear functions.

We can refine our binary search to take advantage of the piecewise linear structure of $F(z)$. Given an iterate z_1 we evaluate $F(z_1)$ using the network of the previous section. In addition to giving the value of $F(z_1)$ our technique gives us the assortment S^{z_1} that achieves the value $F(z_1)$. Viewing $F(z_1)$ as a piecewise linear function this allows us to identify the linear piece that is "active" in $F(z)$ when $z = z_1$. That is, we can identify the set S^{z_1} such that $F(z_1) = V(S_i^{z_1})^{\gamma_i}(R(S_i^{z_1}) - z_1)$. See Figure 8.

Suppose that $v_0 z_1 > F(z_1)$. The naive binary search would adjust by choosing a smaller value of z and iterating. Instead we find the intersection of the linear piece $V(S_i^z)^{\gamma_i}(R(S_i^z) - z)$ with the line $v_0 z$. Let z_2 be this intersection point, so that $v_0 z_2 = V(S_i^{z_1})^{\gamma_i}(R(S_i^{z_1}) - z_2)$. Since $v_0 z_1 > F(z_1)$ we have $z_2 < z_1$. The value z_2 becomes our new iterate.

Because $F(z)$ is convex in z each linear piece will only be active once. Hence, the number of iterations before termination is bounded by the number of pieces defining $F(z)$.

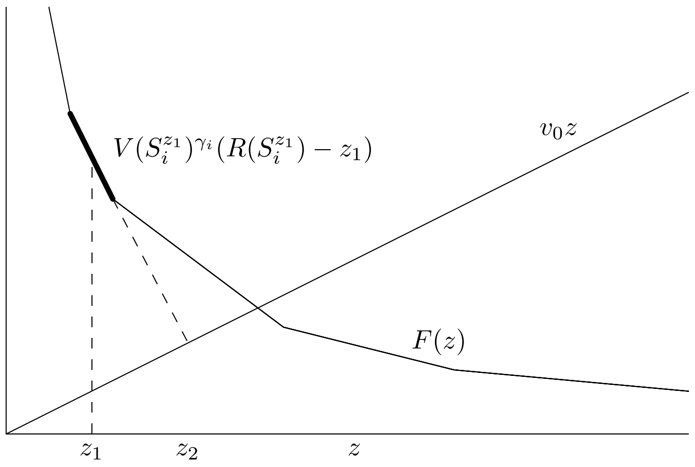

Figure 8: An example showing the iteration procedure in the binary search. Intuitively, the binary search procedure is "traveling" along the dotted lines, with a start position at z_1.

References

[1] S. P. Anderson and A. de Palma. Multiproduct firms: A nested logit approach. *The Journal of Industrial Economics*, 40(3):261–276, 1992.

[2] G. Aydin and Jennifer K. Ryan. Product line selection and pricing under the multinomial logit choice model, 2000. Working Paper.

[3] K. Bagwell. Pricing to signal product line quality. *Journal of Economics & Management Strategy*, 1:151–174, 1992.

[4] M. Ben-Akiva and S. Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT Press, Cambridge, MA., 1997.

[5] A. Borsch-Supan. On the compatibility of nested logit models with utility maximization. *Journal of Econometrics*, 43:373–388, 1990.

[6] J. Davis, G. Gallego, and H. Topaloglu. Assortment planning under the multinomial logit model with totally unimodular constraint structures, 2013.

[7] J. Davis, G. Gallego, and H. Topaloglu. Assortment optimization under variants of the nested logit model. *Operations Research*, 2014. (to appear).

[8] A. Gabor and C. Granger. The pricing of new products. *Scientific Business*, 3:141–150, 1965.

[9] G. Gallego and H.Topaloglu. Constrained assortment optimization for the nested logit model, 2013. Working Paper.

[10] G. Gallego and C. Stefanescu. Upgrades, upsells and pricing in revenue management, 2009.

[11] G. Gallego and R. Wang. Multi-product price optimization and competition under the nested attraction model. Technical report, Columbia Univeristy, 2011.

[12] D. A. Garvin. What does product quality really mean? *Sloan Management Review*, 26, Fall:25–43, 1984.

[13] W. J. Hopp and X. Xu. Product line selection and pricing with modularity in design. *Management Science*, 7(3):172–187, 2005.

[14] R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis.* Wiley, New York, NY, 1959.

[15] D. McFadden. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Economics*, pages 105–142. Academic Press, 1974.

[16] D. McFadden. Econometric models for probabilistic choice among products. *The Journal of Business*, 53(3):S13–29, 1980.

[17] K. Monroe. Buyers subjective perceptions of price. In *Perspectives in Consumer Behavior.* Scott, Foresman and Co., Glenview, IL.

[18] W. Z. Rayfield, P. Rusmevichientong, and H. Topaloglu. Approximation methods for pricing problems under the nested logit model with price bounds, 2012. Working Paper.

[19] P. Rusmevichientong, B. Van Roy, and P. W. Glynn. A nonparametric approach to multiproduct pricing. *Operations Research*, 54:82–98, 2006.

[20] T. Scitovszky. Some consequences of the habit of judging quality by price. *Review of Economic Studies*, 120:100–105, 1944.

[21] S. M. Shugan. Price-quality relationships. *AMA Educators Proceedings*, 51:627–632, 1985.

[22] K. Talluri and G. Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

[23] H. C. W. L. Williams. On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and Planning*, 9(3):285–344, 1977.

[24] E. Yoon and V. Kijewski. Dynamics of the relationship between product features, quality evaluation, and pricing. *Pricing Strategy and Practice*, 5(2):45–60, 1997.