Algorithmic and Linear Programming-Based Techniques for the Maximum Utility Problem

by

Paul Lawrence

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A common topic of study in the subfield of Operations Research known as Revenue Management is finding optimal prices for a line of products given customer preferences. While there exists a large number of ways to model optimal pricing problems, in this thesis we study a price-based Revenue Management model known as the *Maximum Utility Problem* (MUP). In this model, we are given a set of n customer segments and m products, as well as *reservation prices* R_{ij} which reflect the amount that Segment i is willing to pay for Product j. Using a number of structural and behavioral assumptions, if we derive a vector of prices for our line of products, we can compute an assignment of customers to products. We wish to find the set of prices that leads to the optimal amount of revenue given our rules for assigning customers to products. Using this framework, we can formulate a Nonlinear Mixed Integer Programming formulation that, while difficult to solve, has a surprising amount of underlying structure. If we fix an assignment and simply ask for the optimal set of prices such that said assignment is feasible, we obtain a new linear program, the dual of which happens to be a set of shortest-paths problems. This fact lead to the development of the *Dobson-Kalish Algorithm*, which explores a large number of assignments and quickly computes their optimal prices.

Since the introduction of the Dobson-Kalish Algorithm, there has been a rich variety of literature surrounding MUP and its relatives. These include the introduction of *utility tolerances* to increase the robustness of the model, as well as new approximation algorithms, hardness results, and insights into the underlying combinatorial structure of the problem. After detailing this history, this thesis discusses a range of settings under which MUP can be solved in polynomial time. Relating it to other equilibria and price-based optimization problems, we overview *Stackelberg Network Pricing Games* as well as the general formulations of *Bilevel Mixed Integer Linear Programs* and *Bilinear Mixed Integer Programs*, showing that our formulation of the latter is in fact a more general version of the former. We provide some new structured instances for which we can prove additional approximation and runtime results for existing algorithms. We also contribute a generalized heuristic algorithm and show that MUP can be solved exactly when the matrix of reservation prices is rank 1. Finally, we discuss techniques for improving the upper bound to the overall problem, analyzing the primal and dual of the linear programming relaxation of MUP. To test the effectiveness of our approach, we analyze numerous examples that have been solved using Gurobi and present possible avenues for improving our ideas.

Acknowledgements

The research for this thesis was supported in part by the following scholarships, grants, and awards: University of Waterloo's International Masters Award for Student Excellence, Sinclair Graduate Scholarship, C&O Graduate Award, UW Grad Scholarship, Mathematics Faculty Research Chair funds, and an NSERC Discovery Grant.

First and foremost, I would like to thank my advisor, Levent Tunçel, for his excellent guidance throughout the entire Master's degree process. Without his direction, encouragement, and editorial prowess, this thesis could not have reached its current level of quality. I would also like to thank my readers, Chaitanya Swamy and Ricardo Fukusawa, for their helpful comments and advice.

I am very grateful for the wonderful administrative staff at the Combinatorics and Optimization department, especially Melissa Cambridge and Carol Morrison, for showing me the occasional sunset, and who also have been very friendly and patient with answering my silly questions. Likewise, I am thankful for the robust and dynamic community of graduate students in both the C&O and Pure Maths departments. The fifth floor of MC has provided a rich learning environment, and I cannot think of a better place to have studied.

I would especially like to thank the labor organizers at CUPE and the incredible community of graduate students in OrganizeUW. They have succeeded in organizing a union for sessionals, and are actively working toward the effort to unionize TAs and RAs. Hopefully, through collective bargaining, we can move closer towards a system of labor that is equitable for all of us.

Throughout my entire life, I have been incredibly blessed to have wonderful communities of friends everywhere I have lived. To all of these people, both inside and outside of Waterloo, I thank you for your company over the past two turbulent years. In Waterloo, you have welcomed me into a diverse community, kept me sane, and made me feel safe, welcome, and inspired. For those outside of Waterloo, you have stayed in touch, and know that this simple act has meant everything to me. There are far too many of you to list, so I will simply say that if you are reading this, you know who you are.

Finally, I want to thank my sister, Rebecca Lawrence, and my parents, Carla and David Tachau Lawrence, for their eternal support throughout my entire educational journey. Without your love and encouragement, none of this would have been possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

At the heart of many problems in business and industry is a simple question: *How can one maximize profit subject to the constraints of their environment?* The continual search for answers to this question has lead in large part to an explosion of interest in the fields of Operations Research and more specifically Revenue Management over the past few decades. The field of Operations Research itself has a detailed and rich history. Broadly, *Operations Research* (OR), also sometimes called *Management Science*, refers to the branch of mathematical sciences that uses analytic and mathematical arguments to improve decision-making. At its core, it shares a close relationship with the technical field of *Mathematical Optimization*, which focuses on finding optimal solutions to various functions subject to constraints. In fact, one of the most fundamental tools in OR, *Linear Programming*, has its origins in Optimization, due to the development of the Simplex Method by George Dantzig in 1947 [14]. OR owes its own origins to World War II, during which British military forces were faced with a difficult resource management problem. Given scarce supplies and a large number of military operations, both the British and United States management hired large teams of scientists to find effective approaches to resource allocation and other tactical problems - hence the term Operations Research, as these scientists were *researching* (military) *operations*. After the war, interest in OR exploded, and over the rest of the century came to be a central tool in many areas of industry and science, including vehicle routing, resource distribution, supply-chain management, and product pricing [29].

Revenue Management (RM), on the other hand, is the specific subfield of OR that is concerned with maximizing profits in a given market economy. The origins of RM are commonly traced back airline companies in the 1970s. BOAC (now British Airways) began offering restricted discounts on certain types of airfare, mixing lower and higher-fare customers in the same flight. This provided beneficial returns for the company, as they were able to stimulate demand for seats that otherwise would have remained empty. Later, in the 1980s, American Airlines expanded on these practices, using analytical techniques to micromanage inventory control and discount policies to combat the emergence of numerous low-cost competitor airlines [57]. Due to the success of RM in the airline industry, the discipline eventually spread to other transportation and hospitality sectors, and is now used in a wide variety of industries, such as finance, product distribution, and telecom services [46].

While within Revenue Management there are many different avenues, models, and techniques to

maximize profit, this thesis is interested in a price-based RM scenario, which is commonly applied in environments where it is reasonable to change the prices of products frequently. In this framework, there are two major barriers to maximizing revenue. The first is that in order to make any sale in the first place, one of our products must be purchased by a customer in preference over a competitor's. The solution to this first requirement is simple: simply make our products more enticing to buyers than our competitor's products are. The second barrier is more difficult to solve. Suppose that we have several different sets of potential customers, each with a different set of valuations for our products. While some customers may only be interested in purchasing less expensive products, we must be careful that our pricing scheme does not entice higher-valued buyers away from more expensive products, a phenomenon known as *cannibalization*. Thus, micromanaging the price of products to maximize revenue is an important component of overall profit. Moreover, speed and accuracy of price adjustments is paramount, as we may need to quickly adjust to competitor's changes in prices, or new developments on the market.

Another concern in evaluating algorithms and models in RM is that practical applications often rely on users being able to adequately assess, or at least estimate, their customer's valuations. In some instances a precise representation is reasonable to achieve, but there is ultimately a tradeoff between time, expenses, and accurate data. There are several approaches to maneuver around this issue. One is to develop techniques for "filling in" missing data entries based on the given ones. Another is to use a stochastic model, where instead of treating our data as deterministic we assume some probabilistic model for the valuations of data. Yet another is to identify underlying structural patterns that appear in the dataset that arise in practical applications, then design the data collection and forecasting techniques to respect such patterns. Additionally, when developing these workaround strategies, it is important to keep in mind the application it is being used for. As we will see later in this thesis, often times in optimization worst-case examples for algorithms are structurally very far apart from the types of instances we expect to see in real-world instances. Hence, it is important to motivate our study of special-case instances properly. Such is the main focus of this thesis - we are primarily interested in identifying structured data cases that arise in applications within RM. Then, we can devise algorithmic approaches to exploit these structures.

The specific Revenue Management problem we are interested in for this thesis is a classic pricing problem. Given a line of products and a set of potential buyers, we wish to pick a set of prices for our products that maximizes the amount of revenue generated through purchases. Formally, we may assume that we have a set $I = \{1, \ldots, n\}$ of *customer segments* and a set $J = \{1, \ldots, m\}$ of products, where each segment $i \in I$ has N_i members. Each of the customer segments reflects a set of customers that we assume to have identical purchasing behavior. While there are many different methods of modeling customer choice [54], in this thesis we focus on those based on reservation prices.

For a customer segment i and Product j, we let R_{ij} denote the *reservation price* of Segment i for Product j. This number reflects the value that the members of Segment i hold for Product j - or in other words, the amount that the members are willing and able to spend on the product. If the price of Product j is set to some value π_j, then the *utility* (or *surplus*, the terms are used interchangeably) of Segment i for Product j is equal to $R_{ij} - \pi_j$. We additionally assume there are market competitors; let CS_i denote the maximum surplus of Segment i across all competitor products. We will assume without loss of generality that R_{ij} and CS_i are nonnegative for all i and j.

While there are many options for mathematically modelling customer behavior, the scenario we are interested in is the *envy-free pricing* model, where we assume that a customer will always choose to purchase a product that maximizes their utility. Formally, given a set of products J and a price vector π, Segment i will choose to purchase Product j only if

$$j \in \arg\max_{k \in J} \{R_{ik} - \pi_k\} \tag{1.1}$$

and

$$R_{ij} - \pi_j \geq CS_i, \tag{1.2}$$

where the second assumption guarantees that our customer segment does not desire any competitor product over Product j. We also assume WLOG that $R_{ij} \geq CS_i$ for all i, j - otherwise, Segment i will never purchase Product j and we can remove any corresponding constraints from our model. A large number of optimal pricing models additionally require that the surplus of a potential purchase must be nonnegative in order for a segment to buy a particular product. This condition is implicit in our model through constraint (1.2) as CS_i is nonnegative for all i. Furthermore, we make the following assumptions:

a. *Homogeneity*: Each customer segment represents a selection of individual customers with identical valuations and behaviors.

b. *Unlimited Capacity*: There is no constraint on production capacity for any product.

c. *Unit Demand*: Each customer segment buys at most one product and at most one unit of a product.

d. *Non-Differentiated Pricing*: Every customer segment pays the same price for each product.

e. *Static Competition*: The price of any products sold by our competitors is fixed, and thus CS_i is treated as a constant.

f. *Tie-Breaking*: In the case of utility ties, we always assume that a segment i purchases the product with a higher price. In the case that products have the same price, we assume that the segment purchases the product with the lowest index (i.e. a segment prefers Product 1 to Product 2, Product 2 to Product 3, and so on).

These assumptions are common in the literature, yet workarounds exist for each. For example, see Section 5 of [51] for additions to the model that can handle capacity constraints. The last one in particular is highly optimistic, yet does not change the worst-case analysis of the problem. Future work, namely [51] and this thesis, introduce an additional parameter called *utility tolerance* that provides a more sophisticated method of tiebreaking and also adds an additional layer of robustness to the model. Putting these tools together, we arrive at the following Nonlinear Mixed

Integer Program:

$$\begin{aligned}
\max & \sum_{i=1}^{n}\sum_{j=0}^{m} N_i \pi_j \theta_{ij}, && && (1.3)\\
\text{s.t. } & (R_{ij}-\pi_j)\theta_{ij} \geq (R_{ik}-\pi_k)\theta_{ij}, && \forall i,j,\ \forall k \neq j,\\
& (R_{ij}-\pi_j)\theta_{ij} \geq CS_i\theta_{ij}, && \forall i,j,\\
& \sum_{j=1}^{m}\theta_{ij} \geq 1, && \forall i,\\
& \pi_j \geq 0, && \forall j,\\
& \theta_{ij} \in \{0,1\}, && \forall i,j,
\end{aligned}$$

which has decision variables defined as follows:

$$\begin{aligned}
\pi_j &:= \text{price of Product } j,\\
\theta_{ij} &:= \begin{cases} 1, & \text{if customer segment } i \text{ buys Product } j,\\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

We may additionally preprocess our data such that $R_{ij} \leftarrow \max\{R_{ij} - CS_i, 0\}$ and $CS_i \leftarrow 0$ for all i, j. This allows us to replace the second set of constraints with a simple nonnegativity constraint. Thus, (1.3) becomes:

$$\begin{aligned}
\max & \sum_{i=1}^{n}\sum_{j=0}^{m} N_i \pi_j \theta_{ij}, && && (1.4)\\
\text{s.t. } & (R_{ij}-\pi_j)\theta_{ij} \geq (R_{ik}-\pi_k)\theta_{ij}, && \forall i,j,\ \forall k \neq j,\\
& (R_{ij}-\pi_j)\theta_{ij} \geq 0, && \forall i,j,\\
& \sum_{j=1}^{m}\theta_{ij} \geq 1, && \forall i,\\
& \pi_j \geq 0, && \forall j,\\
& \theta_{ij} \in \{0,1\}, && \forall i,j.
\end{aligned}$$

For the remainder of this thesis, we refer to (1.4) as the *Maximum Utility Problem*, or MUP (also called the *Envy-Free Optimal Pricing Problem* in the literature). Unsurprisingly, this optimization problem is $\mathcal{NP}$-hard and $\mathcal{APX}$-hard [27]. While the notion of $\mathcal{NP}$-hardness is commonly known to most mathematicians (there exists no polynomial time algorithm to solve MUP exactly unless $\mathcal{P} = \mathcal{NP}$), the class $\mathcal{APX}$ is not studied as frequently. Formally, an optimization problem is in $\mathcal{APX}$ if it is in $\mathcal{NP}$ and there exists a polynomial-time approximation algorithm with approximation ratio bounded by a constant [5]. Additionally, a problem is said to have a *polynomial-time approximation scheme* (PTAS) if there exists a polynomial-time algorithm to approximate that problem within a factor of $1 + \epsilon$ for *every* $\epsilon \geq 0$. A classic example of a problem with a PTAS is the *Euclidean Traveling Salesman Problem*, where the travel nodes are plotted on the Euclidean plane [4]. Then, a problem ϕ is in $\mathcal{APX}$-hard if, for every problem in $\mathcal{APX}$, there exists a PTAS reduction (a specific type of approximation-preserving reduction) to ϕ. As a consequence, if $\mathcal{P} \neq \mathcal{NP}$, then no

$\mathcal{APX}$-hard problem has a PTAS [45]. For additional nonapproximability results, see [10], which discusses the hardness of various pricing problems related to MUP, as well as [28] and [15], which discuss *Unique Coverage*, a problem that is closely related to envy-free pricing.

Additionally, we note that the Maximum Utility Problem belongs to a larger class of optimization problems whose solutions can be characterized as economic equilibria. In this context, *equilibria* refers to a solution to a problem in which multiple participants are solving some optimization function that may be parameterized by the decisions of other parties. Given these decisions, a solution to the overall problem is one in which no party can optimize further assuming the actions of other participants stay fixed. As is the case with combinatorial optimization in general, many of the techniques for one problem in this class may be easily modified for use in another. Thus, MUP is an important topic of study not just for applications in Revenue Management, but also to enrich our understanding of price-based optimization problems in general.

The outline of this thesis is as follows. In Chapter 2, we introduce the celebrated Dobson-Kalish algorithm, a classic method for the Maximum Utility Problem, and detail much of the relevant literature surrounding both the algorithm and MUP itself. In Chapter 3, we review literature that discusses some specially structured instances of MUP for which exact, strongly polynomial-time algorithms exist. We also include in Chapter 3 several frameworks for tying generalizations of pricing problems and economic equilibria together. We begin with Stackelberg Network Pricing Games and Bilevel Mixed Integer Linear Programs, then contribute an additional generalization in the form of Bilinear Mixed Integer Programs. Chapters 4 and 5 detail further research into the structure of the model, offering new algorithmic insights and observations based on linear programming relaxations. We summarize the thesis and present opportunities for future research in Chapter 6. The original contributions of this thesis are Section 3.4, Chapter 4, and Chapter 5.

Chapter 2

Price Setting Subproblem and the Dobson-Kalish Algorithm

Let $\theta \in \{0,1\}^{n\times m}$ be a *feasible assignment* if there exists some $\pi \in \mathbb{R}^m_+$ such that (θ, π) is a feasible solution to the Maximum Utility Problem formulation (1.4). Now, assume that we are given some θ and are asked to find a pricing vector π^* such that (θ, π^*) is feasible and the objective function of MUP is optimized. Assuming that, under θ, each segment is assigned at most one product, we introduce the following notation:

$$\begin{aligned} C_j &:= \text{set of customer segments who buy Product } j \text{ in } \theta. \\ B &:= \{j : C_j \neq \emptyset\}. \\ M_j &:= \sum_{i\in C_j} N_i. \end{aligned}$$

We may intuit B to be the set of products that are bought by at least one segment in θ, and M_j to denote the total number of customers who buy Product j. Then (1.4) simplifies to:

$$\begin{aligned} \max & \sum_{j=1}^{n} M_j \pi_j, \\ & R_{ij} - \pi_j \geq R_{ik} - \pi_k, && \forall j \in B,\ \forall k \in J \setminus \{j\},\ \forall i \in C_j, \\ & R_{ij} - \pi_j \geq 0, && \forall j \in B,\ \forall i \in C_j, \\ & \pi_j \geq 0, && \forall j. \end{aligned}$$

Rearranging constraints, we can further simplify to:

$$\begin{aligned} \max & \sum_{j=1}^{m} M_j \pi_j, && (2.1) \\ \text{s.t. } & \pi_j - \pi_k \leq \min_{i\in C_j}\{R_{ij} - R_{ik}\}, && \forall j \in B,\ \forall k \neq j, \\ & \pi_j \leq \min_{i\in C_j}\{R_{ij}\}, && \forall j \in B. \end{aligned}$$

Solving (2.1) computes the optimal pricing vector π^* for the fixed assignment θ - however, there exists a very efficient method of solving (2.1). Notice that this LP is the dual of a flow problem on a digraph $D = (N, A)$, with edge costs $\min_{i \in C_j}\{R_{ij} - R_{ik}\}$ for each edge $kj \in A$, and a source node 0 with edge costs $\min_{i \in C_j}\{R_{ij}\}$ for each edge $0j \in A$. Then, we see that (2.1) is simply maximizing the values of the node potentials of D, with the convention that the potential of the source is set to $\pi_0 = 0$.

Given this observation, we can note some special facts of (2.1): namely, that there exists a feasible solution of (2.1) if and only if D has no negative-cost directed cycle, and the optimal solution of (2.1) is to set π_j equal to the length of the shortest directed path from node 0 to node j in D [13]. This construction can be seen explicitly when we take the dual of (2.1):

$$\begin{aligned} \min & \sum_{j,k} r_{jk}\omega_{jk} + \sum_j \sigma_j w_j, & \\ \text{s.t.} & \sum_{k \neq j} \omega_{jk} - \sum_{k \neq j} \omega_{kj} + w_j = M_j, & \forall j \in B, \\ & \sum_{k \neq j} \omega_{kj} = 0, & \forall j \notin B, \\ & \omega_{jk}, w_j \geq 0, & \forall j, k, \end{aligned}$$

where

$$r_{jk} := \min_{i \in C_j}\{R_{ij} - R_{ik}\}, \qquad \sigma_j := \min_{i \in C_j}\{R_{ij}\}. \tag{2.2}$$

We can simplify this formulation by removing all ω_{kj} for $j \notin B$ since the second set of equality constraints and the nonnegativity constraints together imply $\omega_{kj} = 0$, for all $j \notin B$. Finally, by adding a single redundant constraint, we can turn the above formulation into a set of $|B|$ shortest path problems on the digraph D:

$$\begin{aligned} \min & \sum_{j,k} r_{jk}\omega_{jk} + \sum_j \sigma_j w_j, & \\ \text{s.t.} & \sum_{k \neq j} \omega_{jk} - \sum_{k \neq j} \omega_{kj} + w_j = M_j, & \forall j \in B, \\ & -\sum_j w_j = -\sum_j M_j, & \\ & \omega_{jk}, w_j \geq 0, & \forall j, k \in B. \end{aligned} \tag{2.3}$$

We refer to the linear program (2.3) as the *Price Setting Problem*. In this formulation, there exists a node for each product $j \in B$ and one extra "dummy" product (node 0, or the source node) that represents a customer buying nothing. For every two product nodes k, j, there exists a directed edge $k \to j$ with cost $r_{jk} = \min_{i \in C_j}\{R_{ij} - R_{ik}\}$. For every product j, we also have a directed edge from node 0 to j with cost $\sigma_j = \min_{i \in C_j}\{R_{ij}\}$. Thus, given the assignment θ, the optimal price π_j of Product j is precisely the cost of the shortest path from node 0 to node j. Note the formulation in [51] instead uses directed edges from j to node 0 for every product j; the constructions are equivalent. Also note that in the case where we do *not* assume $R_{ij} \geq CS_i$ for all i, j, D will not necessarily be complete. [51] also noted some special properties of (1.4) that are due to its relationship with the Price Setting Problem:

a. The segment assignments $C_j, \forall j$ correspond to feasible assignments in (1.4) if and only if the resulting network of (2.3) has no negative cost directed cycle.

b. There exist optimal prices to (1.4) that are integer-valued if all the data are also integer-valued.

c. In every optimal solution, there exists at least one product k such that $\pi_k = \min_{i \in C_k}\{R_k\}$.

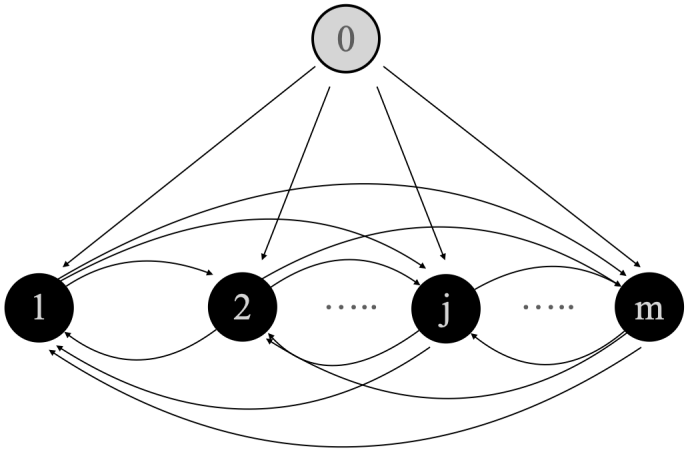

Figure 2.1: The underlying digraph of (2.3), assuming all j are in B

Property **a** is identical to the property we discussed above while discussing node potentials, but it is good here to have an explicit connection to the original formulation (1.4). It is particularly convenient as it provides us a method of determining whether an assignment is feasible. Indeed, for any given assignment, solving the set of shortest paths problems either gives the optimal set of prices or finds a cycle of negative cost, showing that the assignment is infeasible.

More broadly, the advantage of examining the Price Setting Problem is that it can be solved quickly; each instance seeks to solve $|B|$ shortest paths problems. Since the graph D may have negative weights, we can use Bellman-Ford-Moore to compute the shortest paths in strongly polynomial time. However, while solving (2.3) for a single assignment is simple, there are potentially exponentially many possible assignments. Thus, we need a more efficient heuristic to guide our use of this technique. The most fundamental algorithm discussed in the literature is one proposed by Dobson and Kalish [17] in 1988. In addition to using the Price Setting Problem to efficiently find optimal prices for assignments, this algorithm makes specific use of the fact that for every product being offered in the assignment, there is some customer segment assigned to it that "constrains" its price from being raised any higher. In particular, we rely on the following observation:

Proposition 2.0.1. *For every product j that is the child of Product k in the solution of the shortest paths problem* (2.3)*, there exists a customer segment i^* assigned to j that prevents us from raising its price - in particular,*

$$i^* \in \arg\min_{i \in C_j} \{R_{ij} - R_{ik}\}.$$

We refer to i^* as the "critical segment" for Product j.

Proof. Let Product j be the child of Product k in the shortest paths tree. By definition, the cost of edge kj is $\pi_j^* - \pi_k^* = \min_{i \in C_j}\{R_{ij} - R_{ik}\} = R_{i^*j} - R_{i^*k}$. So $\pi_j - R_{i^*j} = \pi_k - R_{i^*k}$, and thus if π_j was increased, Segment i^* would gain a larger utility by switching to purchasing Product k instead of j. □

Algorithm 1 Dobson-Kalish Reassignment Heuristic (DK88)

Require: A feasible product-segment assignment and its corresponding optimal tree solution from solving (2.3).

repeat
  for all Products/nodes j where $C_j \neq \emptyset$ **do**
    Suppose edge (k, j) is in the spanning tree solution.
    if $k \neq 0$ **then**
      For every i^* such that $i^* \in \arg\min_{i \in C_j}\{R_{ij} - R_{ik}\}$, reassign Segment i^* to Product/node k.
    else
      For every i^* such that $i^* \in \arg\min_{i \in C_j} R_{ij}$, delete Segment i^*.
    end if
    Resolve the shortest path problem on the new network and record the change in the objective value.
    Restore the original network.
  end for
  Perform the reassignment that resulted in the maximum increase in the objective value. Resolve shortest path problems and update the optimal spanning tree.
until No reassignment improves the objective value.

After obtaining the optimal prices for some initial feasible assignment, the algorithm enters into a sequence of iterations. In each iteration, we consider every single critical segment, temporarily reassign it to the product k that is constraining the price of Product j, and recompute the shortest paths on the new underlying network, recording the overall objective value. Over all possible new assignments, the algorithm selects the one that leads to the best improvement in objective value, and reassigns the respective customer segment to its new product, before iterating again. The algorithm terminates when it cannot find a reassignment that improves the objective value. Note that assigning a customer segment to the dummy node translates to that segment purchasing nothing. As a drawback, however, the algorithm has no method of "reviving" these removed segments - once they are assigned to the dummy node, they cannot be reassigned again. The same is true for products - once a product has no segments purchasing it in a given assignment, it is effectively removed from the instance. Let's illustrate the motion of reassignments through an example.

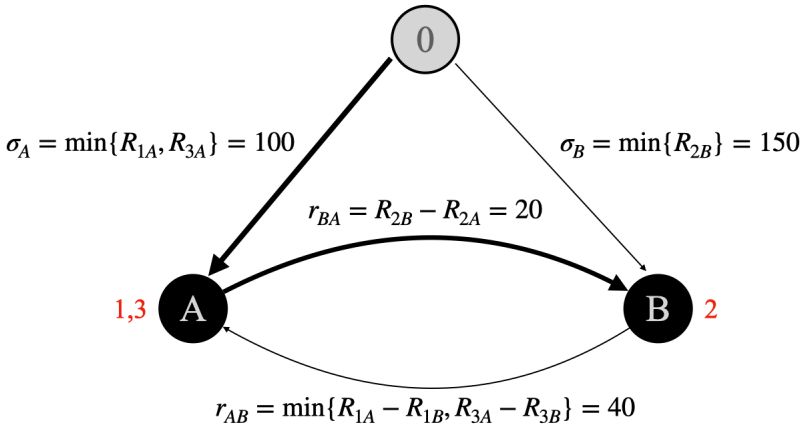

Figure 2.2: The initial assignment of Example 2.0.2, with optimal shortest paths highlighted.

Example 2.0.2. *Consider an instance of* MUP *with the following dataset:*

i	N_i	R_{iA}	R_{iB}
1	1	100	60
2	1	130	150
3	1	220	120

Table 2.1: Example of a simple potential instance

To begin, we set our initial feasible assignment as $C_A = \{1, 3\}$, $C_B = \{2\}$. *Then, the underlying shortest paths graph is represented by Figure 2.2. In the initial assignment, we see that the shortest path to node A has length* 100, *and the shortest path to node B has length 120. Thus, we set* $\pi_A = 100$ *and* $\pi_B = 120$. *Since there is no negative cost cycle, the assignment is feasible, and has revenue* $2 \cdot 100 + 120 = 320$.

Next, we notice that Segment 1 *is the critical segment for edge A, while Segment* 2 *is the critical segment for edge* r_{BA}. *Thus, there are two potential reassignments we can make. Reassigning Segment 2 to node B produces a new underlying graph, represented in Figure 2.3. This assignment produces a revenue of* $3 \cdot 100 = 300$, *which is less profit than our original assignment produced. However, if we instead reassign Segment* 1 *to node* 0, *eliminating it, we do better.*

Consider the graph in Figure 2.4. The length of the shortest path to node A is 220 *and the length of the shortest path to node B is* 150. *Thus, we set* $\pi_A = 220$ *and* $\pi_B = 150$, *for a total profit of* 370. *Since this is a higher profit than the one realized in Figure 2.3, we officially reassign Segment 1 to the dummy node and iterate again. However, in the next iteration, we only have two possible reassignments: eliminating Segment 2 or eliminating Segment 3. Both produce a lower objective function value than the previous assignment, so the algorithm terminates and reports back the assignment in Figure 2.4 as its final solution.*

Dobson and Kalish claimed that Algorithm 1 runs in polynomial time, specifically $O(m^4 n)$. To

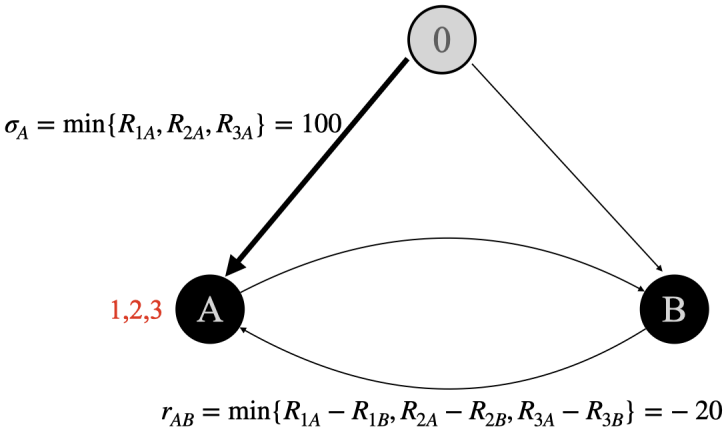

Figure 2.3: Temporarily reassigning Segment 2 to node A produces the above graph.

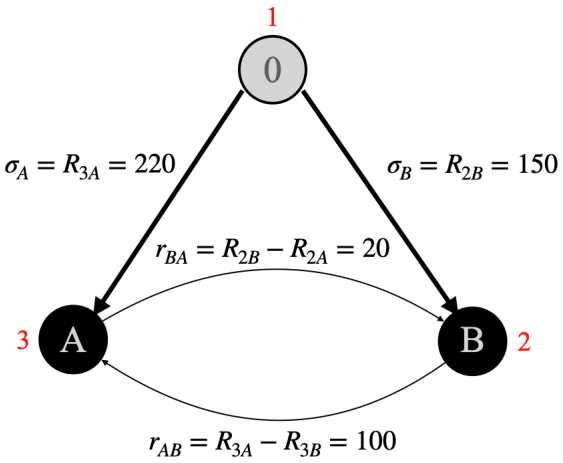

Figure 2.4: The assignment where Segment 1 is eliminated.

support their claim, they stated that every customer segment can be reassigned at most m times. However, Shioda et al. [51] presented a counterexample with 14 customers and 2 products where a segment is reassigned a total of six times, disproving the original claim of Dobson and Kalish.

Additionally, the work of Demirtaş [16] found that when the Dobson-Kalish algorithm is relaxed to instead select *any* reassignment that improves the objective function value, there exists an infinite family of instances of (1.4) that require $\Omega(n^2)$ customer reassignments before termination, even with just 3 products. The methodology used is worth a review; the author begins with a specific assignment and describes a specific series of $\Omega(n^2)$ reassignments. To force an instance to follow these reassignments, they construct a series of linear inequalities that the R_{ij} values must follow. Then, for large enough n, they show that the system of linear inequalities is feasible and define solutions that solve the system. While these instances are the best known family of worst-case complexity examples for the Dobson-Kalish algorithm (or more precisely Dobson-Kalish-*style* algorithms), whether or not there exists a polynomial time upper bound for its runtime is still an open question.

During implementation, there exists an additional optimization we can make. When making a reassignment of a segment from Product j to Product k, rather than recomputing the shortest paths for the entire graph, we only need to recompute on a specific subgraph:

Lemma 2.0.3 (Shioda, et al. 2007 [51])**.** *If Segment i^* is reassigned from Product j to Product k in* DK88*, then the prices of Product j and its children in the spanning tree may increase while all other prices remain the same.*

In particular, the only edge costs that can change are:

- Edges entering node k (whose costs may remain the same or decrease).
- Edges entering node j (whose costs may remain the same or increase).

Thus, the shortest path from node 0 to node k remains the same, and the shortest paths from node 0 to all nodes that are *not* children of j remain the same. Hence, we need only to update the shortest paths of Product j and its children - doing so, the rest of the directed graph remains identical to the previous assignment. This provides a nice improvement to the algorithm's efficiency.

Alternatively, instead of defining critical segments based only on edges in the shortest-path tree, one could consider *all* tight edges - that is, edges kj for which $\pi_j^* - \pi_k^* = \min_{i \in C_j}\{R_{ij} - R_{ik}\}$. This yields a larger set of potential customer reassignments, as there may be tight edges that are not in the tree. However, if we reassign a segment along an edge that is not in the tree, then Lemma 2.0.3 does not apply and we must recompute the entire shortest paths tree.

Another useful property of (1.4) is its natural upper bound. For any customer segment i, define $\bar{R}_i = \max_j\{R_{ij}\}$. Then,

Proposition 2.0.4. *The optimal value of* (1.4) *is at most* $\sum_{i=1}^n N_i \bar{R}_i$.

Proof. Each customer segment may purchase at most one product, and the most that any given customer segment is willing to pay for *any* product is $\bar{R}_i$. Thus, the amount of revenue that can be generated from any given customer segment i is at most $N_i\bar{R}_i$. Summing over all segments gives the upper bound $\sum_{i=1}^n N_i\bar{R}_i$. □

The last piece of the puzzle comes with the initial feasible assignment. The most common choice is the *Maximum Reservation Price Heuristic* (MaxR). While being a relatively simple heuristic,

Algorithm 2 Maximum Reservation Price Heuristic (MaxR)

1: **for all** $i \in I$ **do**
2: Set $\theta_{ij} = 1$ for $j \in \arg\max_k\{R_{ik}\}$, subject to proper tiebreaking.
3: **end for**
4: Solve the Shortest Paths Problem on the resulting underlying digraph.

Algorithm 2 does have a couple of nice properties:

Proposition 2.0.5. MaxR *has the following three properties:*

- *Every customer segment is assigned to one of their most preferred products.*
- MaxR *always produces a feasible assignment.*
- *If* $|C_j| = 1$ *for all* j *and for every segment* i*, their most preferred product* j *is unique, then* MaxR *is optimal for* (1.4).

Proof. The first statement is by definition of the algorithm, as in step 1 we explicitly assign Segment i to a segment j such that $j \in \arg\max_k\{R_{ik}\}$.

The second statement comes from the fact that for every segment i, their most preferred product j, and any other product k, we have $R_{ij} \geq R_{ik}$. Thus, since every segment is assigned to their most preferred product, every edge in the underlying shortest paths graph will be positive, and so there can be no negative directed cost cycle.

For the third statement, we notice that if $|C_j| = 1$ for all j and every segment has a unique most preferred product, the optimal set of prices for the assignment produced by MaxR is simply to set $\pi_j = \bar{R}_i$ for $i \in C_j$ for all j. These prices are feasible since no segment other than i gains a positive utility from purchasing Product j. Moreover, the revenue from these prices is $\sum_{n=1}^{i} N_i \bar{R}_i$, which is our upper bound for MUP from Proposition 2.0.4. □

Both [51] and [48] use MaxR as a starting heuristic for standard implementations of DK88. Later in this thesis, we present alternatives for initial feasible assignments.

2.1 Fixed Points and Reassignment Properties

As we saw above, the fact that computing optimal prices given a fixed assignment is a relatively easy problem allowed us to devise a very nice heuristic for MUP. Furthermore, it is an even simpler task to compute a corresponding assignment vector when given any set of prices - we simply assign every segment to the product maximizing its utility, subject to the tiebreaking rules mentioned in the introduction. We may capture this process notationally - given a feasible pricing vector, the operation $\mathcal{C}(\pi)$ returns the unique feasible assignment θ satisfying the inequalities in (1.4) as well as the tiebreaking rules. Similarly, the operation $\Pi(\theta)$ returns a (not necessarily unique) optimal pricing vector π such that π is feasible for θ. However, given π arbitrarily, it is not necessarily the case that $\Pi(\mathcal{C}(\pi)) = \pi$. Such a vector is known as a *fixed point*:

Definition 2.1.1. *A price vector π is a* fixed point *if $\Pi(\mathcal{C}(\pi)) = \pi$. An assignment θ is a fixed point if $\mathcal{C}(\Pi(\theta)) = \theta$. We also refer to a fixed point by the tuple (π, θ).*

The motivation for considering finding fixed points over single applications of Π is that it helps us find simple reassignments which improve the overall revenue. This is demonstrated in the following example:

Example 2.1.2. *Consider the simple instance:*

i	N_i	R_{i1}	R_{i2}
1	1	5	4
2	1	3	2

Table 2.2: Simple fixed point example

Let θ be the initial assignment that assigns Segment 1 to Product 2 and Segment 2 to Product 1. The optimal prices for this assignment are $\pi_1 = 3, \pi_2 = 2$ for a revenue of 5. However, $\mathcal{C}(\Pi)$ produces the assignment $\hat{\theta}$ that assigns both segments to Product 1, since Product 1 is more expensive. Then in $\hat{\pi} = \Pi(\hat{\theta})$, we have $\hat{\pi}_1 = 3$, and Product 2 is not offered, for a revenue of 6. Since $\mathcal{C}(\hat{\pi}) = \hat{\theta}$, we conclude that $(\hat{\pi}, \hat{\theta})$ is a fixed point.

Given that it is a simple way to improve revenue, one hopes that given any assignment, a fixed point can be computed efficiently. Thankfully, this is indeed the case:

Proposition 2.1.3 (Myklebust, et al. (2011) [48])**.** *Given any feasible assignment θ, a fixed point can be obtained in strongly polynomial time. In particular, there is an optimal solution to* (1.4) *that is a fixed point.*

Proof. Repeatedly apply $\mathcal{C}(\Pi)$ until we obtain a fixed point. Notice that if an application of $\mathcal{C}$ causes a segment i to switch from Product j to Product k, then $R_{ij} - \pi_j = R_{ik} - \pi_k$, since Segment i must have tied utilities for either. Furthermore, $\pi_k \geq \pi_j \Rightarrow R_{ik} \geq R_{ij}$, since the utility of Segment i with Product k must be equal to the utility of Segment i with Product j. Furthermore, we must have either $\pi_k > \pi_j \Rightarrow R_{ik} > R_{ij}$ or $k < j$, since $\mathcal{C}$ only ever reassigns Segment i from Product j to k if Product k is more expensive or if its index is smaller. Now, consider a single Segment i and order every product j in a list L, by descending order of R_{ij}, breaking ties by ascending order of j. Given the arguments above, $\mathcal{C}$ only ever reassigns Segment i from Product j to k if j proceeds k in L. Hence, for any individual segment i, there are at most m applications of $\mathcal{C}(\Pi)$ in which i switches products. Thus, the total number of operations of $\mathcal{C}(\Pi)$ before reaching a fixed point is bounded by $O(nm)$.

Incidentally, the above argument also shows that $\mathcal{C}$ can only ever *increase* the revenue of our current assignment. Thus, since $\Pi(\theta)$ gives us optimal prices for a particular assignment, repeated applications of $\mathcal{C}$ and Π can never *decrease* revenue. Hence, given some optimal solution, we may iterate $\mathcal{C}$ and Π on this solution to find a fixed point (π^*, θ^*). This does not decrease the revenue and hence (π^*, θ^*) is an optimal solution that is a fixed point. □

In addition to the properties of fixed points, due to the reliance of the Dobson-Kalish heuristic on updating to new assignments at each iteration, it is critical to understand some of the underlying

infrastructure with regards to feasible reassignments. These discussions are useful not only for making improvements to DK88, but also to other reassignment heuristics that have been discussed in the literature (such as GLOBAL-DK and CELL-PIERCE from [48]). To begin, recall from Proposition 2.0.1 that in the Price Setting Problem, every product j has a "critical segment" i^* such that $i^* \in \arg\min_{i \in C_j}\{R_{ij} - R_{ik}\}$. DK88 only considers reassigning Segment i from Product j to k if i is a critical segment of j and k is the parent node of j. It turns out that this intuition is necessary for producing feasible assignments:

Proposition 2.1.4 (Characterization of Feasible Single Reassignments)**.** *Given a feasible assignment, reassigning a customer i from j to $k \neq j$ produces a feasible assignment if and only if $R_{ij} - R_{ik}$ is the cost of the shortest path from k to j.*

Thus, if i is assigned to j but is *not* one of its critical segments, reassigning Segment i from j to its parent node k produces an infeasible assignment. While Proposition 2.1.4 provides a necessary and sufficient condition for the ability to reassign a particular customer, we also have two general properties that govern the relationship between two different feasible assignments:

Proposition 2.1.5 (Simultaneous Feasibility)**.** *Suppose θ is a feasible assignment and that a single reassignment produces another feasible assignment θ'. Then there exists π that is feasible for both θ and θ'.*

Lemma 2.1.6 (Weak Exchange Property)**.** *Let θ and θ' be distinct, feasible assignments. Let $(j_i : i \in \{1, 2, \ldots, n\})$ and $(k_i : i \in \{1, 2, \ldots, n\})$ be their respective vectors of purchases. Then there is some i^* such that $j_{i^*} \neq k_{i^*}$ and setting $j_{i^*} := k_{i^*}$ in θ produces a feasible assignment.*

The advantage of these properties is that it allows for algorithms that wish to pivot from assignment to assignment without necessarily moving along a monotone path. This is one of the main disadvantages of the Dobson-Kalish algorithm: it only considers reassignments that strictly increase the objective value. In fact, it is not always guaranteed that there exists a monotone path toward the optimal solution.

Example 2.1.7 (Myklebust, et al. (2011) [48])**.** *Consider an instance with three segments and a single product, where $N_1 = 3$, $N_2 = N_3 = 2$, $R_{11} = 2$, and $R_{21} = R_{31} = 1$. Let the initial assignment be $\theta := [1, 0, 0]$, where Segment 1 is assigned to Product 1 and Segments 2 and 3 are assigned to Product 0, the dummy product. The optimal price for θ is $\pi_1 = 2$, for a revenue of 6. The optimal assignment is $\theta^* = [1, 1, 1]$, $\pi_1^* = 1$, for a revenue of 7. However, there are only three possible single-segment reassignments from θ: assigning Segment 1 to Product 0, or assigning Segment 2 or 3 to Product 1. In the first reassignment, the revenue is decreased to 0, and in the second reassignment, the revenue is decreased to 5. Thus, one cannot make a series of single-segment reassignments from θ to θ^* without decreasing the objective value at some point.*

Hence, DK88 is only effective at searching for local maximizers, and if it terminates, has no ability to guarantee whether its final solution is a global or a local maximum. If one wishes to use a reassignment-based heuristic that is not constrained to always increase revenue at each assignment, then Proposition 2.1.5 and Lemma 2.1.6 guarantee a high degree of mobility throughout the assignment space. Fortunately, the Weak Exchange Property directly implies that at any moment, an optimal solution is not too far away:

Theorem 2.1.8. *Suppose θ and θ' are both feasible assignments that differ in the purchases of ℓ*

segments. Then there is a sequence of ℓ feasible single reassignments connecting θ to θ'.

Proofs of the above results can be found in Section 4 of [48]. While these facts are not directly useful for analyzing the Dobson-Kalish algorithm, understanding the underlying structure of reassignments is an important step towards devising new reassignment-based heuristics. Additionally, it turns out that some of these reassignment properties have connections to other areas of optimal pricing. This topic shall be revisited in Section 3.2 when we discuss Stackelberg Network Pricing Games.

2.2 Utility Tolerances

In 2007, Shioda, Tunçel, and Myklebust [51] presented an alternative formulation of (1.4) with the introduction of what we call *utility tolerances*. To eliminate any optimistic assumptions and to make the model more flexible under changes to the underlying data, we assume that the surplus of any product chosen by Segment i must exceed the surplus of any other product by some constant $\delta_i > 0$, which we may assume is given with the data. Thus, Segment i buys Product j if and only if

$$R_{ij} - \pi_j \geq R_{ik} - \pi_k + \delta_i, \ \forall k \neq j, \tag{2.4}$$

and

$$R_{ij} - \pi_j \geq CS_i + \delta_i. \tag{2.5}$$

Similarly to the original MUP formulation, we assume without loss of generality that $R_{ij} \geq CS_i + \delta_i$ for all i, j. This approach offers a few advantages. First, it allows for the elimination of the optimistic assumption that a customer segment will always purchase the most profitable product in the case of tiebreaks. Second, it adds certainty in the case of a tie between our price for a certain product and another competitors' price. Third, it allows the user of our model to make the choice between being more conservative (using large values of δ_i) or aggressive (using small values of δ_i) in their applications. Fourth, it protects solutions against small perturbations in the underlying data. Overall, the introduction of δ_i constants makes the model significantly more robust.

Incorporating the δ_i constants, the nonlinear Mixed Integer Program formulation of the optimal pricing problem can be expressed as:

$$\begin{aligned}
\max & \sum_{i=1}^{n}\sum_{j=1}^{m} N_i \pi_j \theta_{ij}, && && (2.6)\\
\text{s.t. } & (R_{ij} - \pi_j)\theta_{ij} \geq (R_{ik} - \pi_k + \delta_i)\theta_{ij}, && \forall i, j, \ \forall k \neq j,\\
& (R_{ij} - \pi_j)\theta_{ij} \geq (CS_i + \delta_i), && \forall i, j,\\
& \sum_{j=1}^{m} \theta_{ij} \leq 1, && \forall i,\\
& \pi_j \geq 0, && \forall j,\\
& \theta_{ij} \in \{0, 1\}, && \forall i, j.
\end{aligned}$$

Note that we can preprocess the data such that $R_{ij} \leftarrow \max(0, R_{ij} - CS_i - \delta_i)$ and $CS_i \leftarrow 0$ without changing the above problem. Like the first formulation of MUP, this allows us to simplify the

second line of constraints, while the first line remains effectively the same. Thus, we may instead consider the formulation:

$$\begin{aligned}
\max & \sum_{i=1}^{n}\sum_{j=1}^{m} N_i \pi_j \theta_{ij}, && && (2.7)\\
\text{s.t. } & (R_{ij}-\pi_j)\theta_{ij} \geq (R_{ik}-\pi_k+\delta_i)\theta_{ij}, && \forall i,j,\ \forall k \neq j,\\
& (R_{ij}-\pi_j)\theta_{ij} \geq 0, && \forall i,j,\\
& \sum_{j=1}^{m} \theta_{ij} \leq 1, && \forall i,\\
& \pi_j \geq 0, && \forall j,\\
& \theta_{ij} \in \{0,1\}, && \forall i,j.
\end{aligned}$$

To linearize the above model, we can introduce a continuous auxiliary variable p_{ij} such that:

$$p_{ij} = \begin{cases} \pi_j, & \text{if } \theta_{ij} = 1, \\ 0, & \text{otherwise.} \end{cases}$$

which we force with the constraints

$$\begin{aligned}
& p_{ij} \geq 0, \\
& p_{ij} \leq R_{ij}\theta_{ij}, \\
& p_{ij} \leq \pi_j, \\
& p_{ij} \geq \pi_j - \tilde{R}_j(1-\theta_{ij}),
\end{aligned}$$

for all i, j, where $\tilde{R}_j = \max_i\{R_{ij}\}$. This gives the linearized model:

$$\begin{aligned}
\max & \sum_{i=1}^{n}\sum_{j=1}^{m} N_i p_{ij}, && && (2.8)\\
\text{s.t. } & R_{ij}\theta_{ij} - p_{ij} \geq (R_{ik}-\pi_k+\delta_i)\theta_{ij}, && \forall i,j,\ \forall k \neq j,\\
& R_{ij}\theta_{ij} - p_{ij} \geq 0, && \forall i,j,\\
& \sum_{j=1}^{m} \theta_{ij} \leq 1, && \forall i,\\
& p_{ij} \leq \pi_j, && \forall i,j,\\
& p_{ij} \geq \pi_j - \tilde{R}_j(1-\theta_{ij}), && \forall i,j,\\
& \pi_j, p_{ij} \geq 0, && \forall i,j,\\
& \theta_{ij} \in \{0,1\}, && \forall i,j.
\end{aligned}$$

There is also an alternate form of (2.8) that reduces the number of constraints. For the first set of constraints, summing over all $j, j \neq k$ gives us:

$$\sum_{j\neq k}(R_{ij}\theta_{ij} - p_{ij}) \geq (R_{ik}+\delta_i)\left(\sum_{j\neq k}\theta_{ij}\right) - \pi_k,\ \forall i,k.$$

It is known that for the LP relaxation, the old and new first sets of constraints are not equivalent. However, in computational experiments the new formulation is more efficient [51]. Thus, we leave ourselves open to the opportunity to switch between them as desired. This gives an alternate formulation for the price setting subproblem as well. Fixing an assignment for (2.6), simplifying, and taking the dual gives a linear program identical to (2.3) where the variables in (2.2) are now:

$$r_{jk} := \min_{i \in C_j} \{R_{ij} - R_{ik} - \delta_i\}, \qquad \sigma_j := \min_{i \in C_j} \{R_{ij}\}. \tag{2.9}$$

Thankfully, the addition of utility tolerances still allows for the application of the Dobson-Kalish heuristic, due to the following lemma:

Lemma 2.2.1. *When $\delta_i = 0$ for every i, each reassignment in the Dobson-Kalish Reassignment Heuristic results in a feasible product-segment assignment.*

2.3 Approximation Algorithms

While the Maximum Utility Problem is $\mathcal{APX}$-hard, there exist several approximation results in the literature. In this section, we discuss the results of Guruswami, et al. [27], who introduced Algorithm 3 which we refer to simply as Guru.

Algorithm 3 Guru (Single-Price Heuristic)

1: Define $\bar{R}_i := \max_j \{R_{ij}\}$ for all $i \in \{1, \ldots, n\}$
2: Sort the customer segments such that $\bar{R}_1 \geq \bar{R}_2 \geq \cdots \geq \bar{R}_n$
3: Find index ℓ such that $\ell = \arg\max_k \left\{ \bar{R}_k \sum_{i=1}^{k} N_i \right\}$
4: Set $\pi_j = \bar{R}_\ell$ for all j

This algorithm is polytime, and in fact is an $O(\log(n))$-approximation in the special case that

$$\frac{\max_i N_i}{\min_j N_j} \leq c,$$

for some constant $c = O(1)$. However, when this ratio cannot be bounded by an absolute constant, we may construct instances such that the algorithm performs at $\Omega(n)$. Consider the following example:

Example 2.3.1. *Let $m := n$, and set the reservation prices as follows:*

$$R_{ij} := \begin{cases} 2^{(n-i)}, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Then for each $i \leq n$, set

$$N_i := 2^{i-1}.$$

To illustrate, consider the case where $m = n = 3$. Then,

$$\begin{aligned} R_{11} = 4, & N_1 = 1, \\ R_{22} = 2, & N_2 = 2, \\ R_{33} = 1, & N_3 = 4. \end{aligned}$$

So we set $\pi_j = 4$ for all j. Segments 1 and 2 end up buying nothing, and Segment 3 buys Product 3, for a total profit of 4. However, the optimal prices are:

$$\pi_1 = 1, \pi_2 = 2, \pi_3 = 4,$$

for a revenue of 12.

To show the approximation bound, note that we will always pick some ℓ such that $\pi_j = \bar{R}_\ell = 2^{n-\ell}$ for every j. Thus, the resulting revenue is:

$$\begin{aligned} 2^{n-\ell} \sum_{i=1}^{\ell} N_i &= 2^{n-\ell} \sum_{i=1}^{\ell} 2^{i-1} \\ &= 2^{n-\ell}(2^\ell - 1) \\ &< 2^n. \end{aligned}$$

However, the optimal profit is obtained by having every customer segment purchase their most desired product at maximum price, which can be done by setting $\pi_j = 2^{n-j}$ for all j. The resulting revenue is

$$\sum_{i-1}^{n} 2^{i-1} \cdot 2^{n-i} = n \cdot 2^{n-1}.$$

Thus, the approximation ratio of GURU is at least

$$\frac{n2^{n-1}}{2^n} = \frac{n}{2} = \Omega(n).$$

While GURU has a worst-case approximation ratio of $\Omega(n)$, the work of Myklebust, et al. [48] presents an alternative analysis that lowers the worst-case approximation bound in specific cases. To accomplish this analysis, we can formulate the search for worst-case instances as an optimization problem, treating the $\bar{R}_i$ and N_i values as variables rather than data. Hence, using our upper bound from Proposition 2.0.4, we may consider the following Nonlinear Program:

$$\begin{aligned} \max \sum_{i=1}^{n} N_i \bar{R}_i, & & & (2.10) \\ \text{s.t.} \quad \left(\sum_{k=1}^{i} N_k \right) \bar{R}_i \leq 1, & & \forall i \in I, & \quad (\star) \\ \bar{R}_{i+1} \leq \bar{R}_i, & & \forall i \in I \setminus \{n\}, & \\ \bar{R}_n \geq 0, & & & \\ N_i \geq 0, & & \forall i \in I. & \end{aligned}$$

The objective function corresponds to the maximum possible profit obtainable, in which every customer segment i buys a product j such that $\pi_j = \bar{R}_i$. The constraint $(\star)$ corresponds to the revenue of the GURU algorithm, which we set to be a maximum of 1. This way, we can solve the optimization problem (2.10) to find the largest gap possible between the revenue of GURU and our upper bound. Thus, if we find an improvement to the upper bound, we can use this technique to

gain an even better worst-case approximation ratio. If we take as input (i.e. fix) the N_i variables and compute the optimal $\bar{R}_i$ for the objective function in terms of N_i, we obtain that the value of (2.10) is:

$$\sum_{i=1}^{n} \frac{N_i}{\sum_{\ell=1}^{i} N_\ell}.$$

Similarly, the optimal values for N_i given the $\bar{R}_i$ variables as input is:

$$N_1 = \frac{1}{\bar{R}_1}, N_i = \frac{1}{\bar{R}_i} - \frac{1}{\bar{R}_{i-1}}.$$

This leads to the following value for (2.10):

$$n - \sum_{i=1}^{n-1} \frac{\bar{R}_{i+1}}{\bar{R}_i}.$$

Thus, given $N_i, \bar{R}_i$ variables as input, we can say that GURU has approximation ratio at most

$$\min \left\{ \sum_{i=1}^{n} \frac{N_i}{\sum_{\ell=1}^{i} N_\ell}, n - \sum_{i=1}^{n-1} \frac{\bar{R}_{i+1}}{\bar{R}_i} \right\}, \tag{2.11}$$

so GURU in fact has approximation ratio $\Theta(n)$. A positive consequence of expressing the worst-case bound in this way is that there exist special structural cases for the R_{ij} values that we can exploit to obtain better approximations. To illustrate this idea, we present the following theorem:

Theorem 2.3.2. *Let $n \geq 2$ and $k \in (0, n-1]$. Then if for all $1 \leq i \leq n-1$,*

$$\frac{\bar{R}_{i-1}}{\bar{R}_i} \geq 1 - \frac{k}{n-1}, \tag{2.12}$$

the approximation ratio of GURU *is at most $(k+1)$.*

Proof. Plugging the inequality (2.12) into (2.11), we see that:

$$\begin{aligned} n - \sum_{i=1}^{n-1} \frac{\bar{R}_{i+1}}{\bar{R}_i} &\leq n - \sum_{i=1}^{n-1} \left(1 - \frac{k}{n-1}\right) \\ &= n - (n-1)\left(1 - \frac{k}{n-1}\right) \\ &= k+1. \end{aligned}$$

□

Chapter 3

Related Problems

A common practice in combinatorial optimization when encountering an $\mathcal{NP}$-hard problem is to identify special structural qualities of an instance's data that can be exploited algorithmically. A classic example is the Travelling Salesman Problem, for which there exists no polynomial time α-approximation for any constant $\alpha \geq 1$ (if $\mathcal{P} \neq \mathcal{NP}$). However, in the *Metric Travelling Salesman Problem*, where the distances between points satisfy the triangle inequality, there exists a $1.5 - \epsilon$ approximation [39]. The Maximum Utility Problem is no exception to this phenomenon - already in this thesis, we have seen the algorithm Guru, which was shown in [27] to be an $O(\log(n))$-approximation when the ratio between maximum and minimum N_i values is bounded by a constant. The bulk of this chapter elaborates on that theme, giving a review of the structured settings under which the Maximum Utility Problem can be solved in polynomial time. In later sections, to expand upon the connection between combinatorial optimization, optimal pricing problems, and economic equilibria, we discuss a variety of leader-follower games, including Stackelberg Pricing Games and Bilevel Integer Linear Programs.

3.1 Structured Settings

First, we mention several special cases where an exact solution to the linearized Maximum Utility Problem (2.8) can be found in polynomial time [51]:

a. $n = 1$, in which case there is only one customer segment. Here we may set the price of all products equal to the highest reservation price of Segment 1.

b. $n = O(1)$, in which case the number of possible assignments is bounded by $m^n = m^{O(1)}$. Then for every assignment we can simply obtain its optimal profit or determine if it is infeasible by solving the underlying shortest paths problem. Among all feasible assignments, we simply choose the one with the highest profit.

c. $n \leq m$, in the special case described in the following lemma:

Lemma 3.1.1. *For $n \leq m$, the LP relaxation of (2.8) provides the integer optimal solution if each segment i can be assigned to a product j_i where:*

(a) $R_{ij_i} \geq R_{ik}, \ \forall k \in J,$

(b) $j_i \neq j_\ell,\ \forall i, \ell \in I,\ i \neq \ell,$

(c) $R_{ij_i} \geq R_{\ell j_i} + \delta_i,\ \forall i, \ell \in I,\ i \neq \ell.$

d. $m = 1$, due to the formulation of an alternative LP problem. Assume that Product 1 is the only product being sold. Then order the customer segments such that $R_{11} \geq R_{21} \geq \cdots \geq R_{n1}$. Then MUP has the following formulation:

$$\max \sum_{i=1}^{n} \left(\sum_{k=1}^{i} N_k \right) R_{i1} z_i, \tag{3.1}$$

$$\text{s.t. } \sum_{i=1}^{n} z_i = 1,$$

$$z_i \geq 0,\ \forall i,$$

which has an integer solution $z_{i^*} = 1$ when $i^* \in \arg\max_i \{R_{i1} \sum_{k=1}^{i} N_k\}$.

e. $m \leq n$, according to the conditions in the lemma:

Lemma 3.1.2. *Suppose C_j^* is the set of segments that purchase Product j in the solution of (3.1) and $C_j^* \bigcap C_k^* = \emptyset, \forall j, k \in \{1, \ldots, m\}, j \neq k$. Then C_j^* is the optimal segment-product assignment of (2.8) for $\delta_i \leq \min_{j=1,\ldots,m}\{\min_{l \in C_j^*}\{R_{lj}\} - R_{ij}\}$, where the upper bound is strictly positive. The optimal prices are $\pi_j^* = \min_{i \in C_j^*} R_{ij}$.*

To provide context to the above instances, we note that case **c** is a generalized version of the third property in Proposition 2.0.5, where for every customer segment their most desired product is unique and not desired as much by any other segment. Case **e** is a generalized version of case **d**, where we solve (3.1) for each product and happen to get disjoint C_j^* sets, which allows us to easily copmpute optimal prices.

The second setting we are interested in is when our reservation price matrix satisfies the *Extended Monge Property*:

Definition 3.1.3. *A matrix $R \in \mathbb{R}^{n \times m}$ satisfies the **Extended Monge Property** if the following three properties hold:*

a. *For all $1 \leq i < \ell \leq n$ and $1 \leq j < k \leq m$,*

$$R_{ij} + R_{\ell k} \geq R_{ik} + R_{\ell j}.$$

b. *For all $1 \leq i \leq n$ and $1 \leq j < m$,*

$$R_{ij} \geq R_{i,j+1}.$$

c. *For all $1 \leq i < n$ and $1 \leq j \leq m$,*

$$R_{ij} \geq R_{i+1,j}.$$

Assumption **a** is also equivalent to the statement

$$R_{ij} - R_{i,j+1} \geq R_{i+1,j} - R_{i+1,j+1} \text{ for all } 1 \leq i < n \text{ and } 1 \leq j < m,$$

which is generally referred to as the *Monge Property* [26]. This is a well-studied property of matrices that has been applied to a wide variety of combinatorial optimization problems over the past fifty years. Original applications of this structure were studied by Gaspard Monge in 1781 and A.J. Hoffman in 1961 [30], who both made connections to transportation problems. Since then, a large number of combinatorial optimization problems have been considered under the context of Monge matrices, including the traveling salesman problem, bipartite matching, and scheduling (see [12] and [11] for detailed surveys). On the other hand, assumptions **b** and **c** are quite strong. Taken together, we say that if a set of reservation prices satisfies the final two assumptions, then the set of prices is *Monotone Decreasing*. Under these three assumptions we are able to solve MUP exactly, due to the following theorem:

Theorem 3.1.4 (Günlük, 2008 [26])**.** *If reservation prices satisfy the extended Monge Property, then* (1.4) *can be solved by dynamic programming in* $O(nm^2)$ *time.*

A major building block for the authors' algorithm comes from the observation that under assumption **a**, any assignment vector cannot have any *crossings*. Intuitively, this means that whenever a customer segment i purchases a product j, all segments with an index at least i must also purchase a product with an index at least j. More formally,

Proposition 3.1.5 (Günlük, 2008 [26])**.** *Under assumption* ***a****, for any given price vector* π*, the corresponding assignment* $\theta(\pi)$ *has no crossings. In other words,*

$$\theta_{ik} + \theta_{\ell j} \leq 1 \text{ for all } k > j,\ \ell > i.$$

Proof. To prove this fact, we can simply show that an assignment invalidating the above inequality creates a negative directed cycle in the underlying shortest paths graph. Assume that for some i, j, $\ell > i$, and $k > j$, we have $\theta_{ik} = 1$ and $\theta_{\ell j} = 1$. Then in the digraph, we have that $r_{jk} \leq R_{\ell j} - R_{\ell k}$, and $r_{kj} \leq R_{ik} - R_{ij}$. But by assumption **a**,

$$r_{jk} + r_{kj} \leq R_{\ell j} - R_{\ell k} + R_{ik} - R_{ij} \leq 0.$$

Additionally, we can rule out the scenario where the bound is tight. In this case, Segment i is indifferent between Products j and k. Thus, our tiebreaking rule dictates that Segment i purchase the more expensive product. But if $\pi_k > \pi_j$, then

$$R_{ik} - \pi_k \leq R_{ij} - \pi_k < R_{ij} - \pi_j,$$

so Segment i prefers Product j to Product k. Hence, we may conclude

$$r_{jk} + r_{kj} < 0,$$

so there exists a negative directed cycle in the underlying shortest paths graph. □

This proposition implies that in any optimal assignment vector, the customer segments are partitioned into contiguous groups with the same assigned product. In particular, there exists some ℓ such that segments can be placed into groups $C_1, C_2, \ldots, C_\ell$, where every group i purchases product α_i (if in an assignment there exists segments that purchase nothing, then $\alpha_\ell = 0$, the dummy product). This fact serves as the basis for the following dynamic programming algorithm. For

notation, we define $N(a,b) := \sum_{i=a}^{b} N_i$. Similarly, we will initialize an $n \times m$ array Z of all zeroes in which to store our intermediate values. Then, we can define the algorithm.

Algorithm 4 Günlük, (2008)

for $j = m, m-1, \ldots, 1$ **do**
  $Z_{nj} = N(1,n)R_{nj}$
end for
for $i = n-1, \ldots, 2$ **do**
  for $j = m, \ldots, 1$ **do**
    $Z_{ij} = \max_{b \in \{j,\ldots,m\}} \{N(1,i)(R_{ij} - R_{ib}) + Z_{i+1,b}\}$
  end for
end for
$z^* = Z_{11} = \max_{b \in J} \{N_1(R_{11} - R_{1b}) + Z_{2b}\}$
return z^*

Note that Algorithm 4 requires that every customer segment purchase a product - however this technicality can be dealt with by adding a dummy product $m+1$ such that $R_{i,m+1} = 0$ for all segments i.

As a brief note we include two examples that provide some context to the assumptions made in [26]. In general, for the dynamic programming algorithm to work, it is sufficient but not necessary to make the assumption that the matrix R satisfies the extended Monge property. However, the properties are not individually strong enough. We demonstrate this through the following series of statements.

Remark 3.1.6. *Assumptions* ***b*** *and* ***c*** *are not sufficient to require all optimal solutions to have no crossings.*

Example 3.1.7. *We demonstrate Remark 3.1.6 with the following instance:*

i	N_i	R_{i1}	R_{i2}
1	1	6	5
2	1	4	1
3	1	3	1
4	100	$1+\varepsilon$	1

Table 3.1: A matrix that is not Monge but whose optimal assignment has no crossings

Note that the R_{ij} matrix is not Monge, as $R_{11} - R_{12} < R_{21} - R_{22}$. If we let $\varepsilon \leq 1.01$, we see that the optimal solution is for segments 1 and 4 to purchase Product 2, while segments 2 and 3 purchase Product 1, with $\pi_1 = 3$ and $\pi_2 = 1$. The resulting revenue is 107. However, the most profit we can get from an assignment with no crossings is 106. *This can be achieved with $\pi_1 = 2, \pi_2 = 1$, which has segments* $1, 2$, *and 3 purchasing Product 1 and Segment* 4 *purchasing Product 2.*

Remark 3.1.8. *The Extended Monge Property is not necessary for all optimal solutions to have no crossings. Again, consider the next example:*

Example 3.1.9. *We can see that the following instance is not Monge:*

i	N_i	R_{i1}	R_{i2}
1	1	5	4
2	50	3	1

Table 3.2: Another matrix that is not Monge but whose optimal assignment has no crossings

However, the optimal solution is to set $\pi_1 = 3$ and to not offer Product 2, leading to a revenue of 153 with both customer segments purchasing Product 1.

Moving on from Monge matrices, the last structured instance we are interested in comes when we are interested in finding a *complete* allocation of segments to products. In this particular setting, we assume that each product j has c_j available copies. Moreover, we assume that $N_i = 1$ for every customer segment and that the total number of available products equals the number of customer segments - that is:

$$n = \sum_{j=1}^{m} c_j.$$

This is not a very meaningful restriction, as any general scenario can be modified to satisfy the above constraint simply by adding additional "dummy products" or customer segments with reservation prices of 0. The more restrictive requirement, then, is that for every assignment θ, we require every customer segment to purchase exactly one product - that is:

$$\sum_{j=1}^{m} \theta_{ij} = 1 \ \forall i.$$

Then, every single assignment θ is in fact a permutation $\phi : N \to N$ (where N is the set of customer segments) where $\phi(j)$ is the index of the segment that purchases Product j. Thus, the problem of solving the envy-free pricing model reduces to pricing the individual products in N, finding an envy-free (i.e. each segment is assigned to a product that maximizes their utility) permutation, and its corresponding perfect matching $M = \{(\phi(i), i) : i \in N\}$. If we let Γ denote the set of all permutations of N and $FR(\phi)$ denote the feasible region of envy-free prices induced by ϕ, then we may reduce MUP to the following optimization problem:

$$\begin{aligned} \max_{\phi \in \Gamma} \max_{\pi \in FR(\phi)} & \sum_{i=1}^{n} \pi_i, & \\ \text{s.t. } & \pi_i \leq R_{\phi(i),i} & \forall i \in N, \\ & \pi_i - \pi_j \leq R_{\phi(i),i} - R_{\phi(i),j} & \forall i, j \in N, \\ & \pi_i \geq 0. & \end{aligned}$$

The resulting problem is called Envy-Free Perfect Matching, which has an exact solution, via reduction to perfect matching:

Theorem 3.1.10 (Arbib, Karaşhan, Pinar, 2017 [3])**.** Envy-Free Perfect Matching *can be solved exactly in $O(n^3)$ time.*

Another commonly studied variant of the ubiquitous pricing problem is the MAX-BUYING (and its cousin MIN-BUYING) problem. In this version, rather than determining product-segment assignments directly via the segments' reservation prices, we instead simply seek to find a pricing vector and corresponding *allocation* of products to segments that is feasible, subject to certain constraints. In particular, we say that Product j is feasible for Segment i if $R_{ij} \geq \pi_j$. Then, any assignment θ is *feasible* if

$$\theta_{ij} = 1 \Rightarrow R_{ij} \geq \pi_j.$$

That is, every product-customer pairing is feasible. In addition, in the MAX-BUYING model, each segment purchases the most expensive product among those available, and in the MIN-BUYING model each segment purchases the cheapest. There is also the RANK-BUYING model, where each segment has a preference order among all products, and purchases the product with the highest preference among those that are feasible given the current pricing vector. Then, the envy-free pricing model reduces to simply finding the most profitable assignment θ and corresponding prices π subject to the above constraints.

Another possible assumption for the MAX-BUYING problem is the addition of a price ladder. Under this framework, given an ordering $\{1, \ldots, j \ldots, m\}$ of products, we require that for any pricing vector π,

$$\pi_1 \geq \ \cdots \geq \pi_m.$$

As many models in auction theory, optimal pricing, and mechanism design assume, we can also impose *Limited Supply* on the products, so that every product j has some maximum number of copies C_j that can be sold.

Variations of the overall model can be studied using combinations of these assumptions. An example is the work of Fernandes and Schouery in 2013 [21], who provided polynomial time algorithms for both Max-Buying with Limited Supply (MAX-NPL-LS) and Max-Buying with Limited Supply and a Price Ladder (MAX-PL-LS), with ratios of $\frac{e}{e-1}$ and $2+\varepsilon$, respectively [21]. For other variations, there is extensive research on both approximations and hardness bounds [1][10]. We refer the reader to the cited material for further information.

3.2 Stackelberg Network Pricing Games

In analyzing the Maximum Utility Problem, we are modelling a scenario where multiple parties must take some action - in this case, one party is setting prices, and the customer segments are solving a maximization problem to determine which product to purchase. The resulting optimal solution is a type of equilibria, where neither party can improve their utility or revenue given the actions of the other. In a more general sense, the type of underlying problem that the customers solve and the structure under which prices are set determine the form of the optimization problem that we need to study. Thus, if we *change* the underlying optimization problem that the customers must solve, we dramatically alter the structure of the overall problem. As a warm-up to this idea, we may consider the combinatorial setting. Assume we have a *leader* pricing items in some universe U, and customers (referred to as *followers*) which seek to purchase some feasible subset of elements that maximize a given utility function. Such a framework is referred to as a *Stackelberg Network Pricing Game.*

To demonstrate a simple instance in the graph setting, we define the following Stackelberg Network Pricing Game, which we refer to simply as STACK: let $G = (V, E)$ be a multigraph with $E = E_p \sqcup E_f$. We have two types of players in the game, one *leader* and one (or multiple) *followers*. We call E_p the *priceable* edges and E_f the *fixed-price* edges, with $m = |E_p|$. For every $e \in E_f$ there is a fixed cost $c(e) \geq 0$. For every $e \in E_p$, the leader may define a price $p(e) \geq 0$. Each follower $i = 1, \ldots, k$ has a set $\mathcal{S}_i \subset 2^E$ of *feasible subgraphs*. The weight $w(S)$ of a subgraph S is simply the sum of the edges in the subgraph, which may be either fixed or priceable. Conversely, the *revenue* $r(S)$ of the leader from subgraph S is the sum of the priceable edges in S. We seek to find the pricing function p^* for the leader that generates the maximum revenue, i.e.,

$$p^* \in \arg\max_p \sum_{i=1}^{k} r(S_i^*(p)),$$

where

$$S_i^*(p) = \arg\min_{S \in \mathcal{S}_i} \{w(S)\} \text{ for each } i, \text{ given } p.$$

We note that, given prices for E_p, it is not necessarily the case that followers can find a subgraph of minimum weight in polynomial time (e.g. compute $\arg\min_{S \in \mathcal{S}_i} \{w(S)\}$), nor is it the case that $\mathcal{S}_i$ must have any discernible structure at all. However, in the literature we discuss, these caveats will be assumed.

Even in the single-follower case, this problem is known to be both $\mathcal{NP}$ [43] and $\mathcal{APX}$ [36]-hard. However, there happens to be a simple, single-price algorithm that achieves a nice approximation ratio [9]. Assuming there exists a feasible subgraph for the follower not containing any of the priceable edges and that each follower can find a subset of minimum cost in polynomial time, given a set of prices, let the cheapest of these have cost c_0. Then we have the following algorithm:

Algorithm 5 STACK Single-Price Algorithm

1: **for** $j = 0, \ldots, \lceil c_0 \rceil$ **do**
2: Let $p_j = (1 + \epsilon)^j$.
3: Assign p_j to all priceable edges and record $r(p_j)$.
4: **end for**
5: **return** $p \in \arg\max_{p_j} r(p_j)$.

Theorem 3.2.1 (Briest, Hoefer, Krysta, 2012 [9])**.** *Given any $\epsilon > 0$, the single-price algorithm is a $(1 + \epsilon)H_m$-approximation, where H_m is the m^{th} harmonic number. For* STACK *with k followers, the single-price algorithm computes a $(1 + \epsilon)(H_k + H_m)$-approximation.*

Similarly, in the case where the followers have weighted demands d_j, we can alternately define the revenue for the leader as

$$\sum_{j=1}^{k} d_j \sum_{e \in S_j \cap E_p} p(e),$$

for which the single-price algorithm also has a provable approximation ratio, which does not rely on the number of followers:

Theorem 3.2.2 (Briest, Hoefer, Krysta, 2012 [9]). *The single-price algorithm computes an* $(1+\epsilon)m^2$*-approximation with respect to the optimal revenue for* StackVC *with multiple weighted followers.*

Of course, the above results are quite general, and can be improved when we consider special cases of underlying combinatorial problems. One nice example is the well-studied vertex cover problem on bipartite graphs. In this scenario, we assume that the *vertices* are priceable, rather than edges, and that the followers seek to compute a minimum-cost vertex cover given a set of prices for edges. This problem is known as StackVC, which can be solved exactly in the single-follower case (Note that V_p refers to the set of priceable edges):

Theorem 3.2.3 (Briest, Hoefer, Krysta, 2012 [9]). *If for a bipartite graph* $G = (A \cup B, E)$ *we have* $V_p \subseteq A$*, then there is an* $O(n^6)$ *algorithm computing an optimal price function* p^* *for* StackVC*.*

Of course, [9] is only an introduction to the rich variety of Stackelberg Network pricing problems. There are many new results in the field over the past decade, including:

- An $O(\log m / \log \log m)$ polytime approximation scheme for the tollbooth problem on trees [23].
- A PTAS for the tollbooth problem on paths, also known as the *highway problem* [24].
- An $O(n^4)$ algorithm for StackVC, representing an improvement over Theorem 3.2.3 [6].
- An extension of the results of [9], where instead we let the follower minimize *any* continuous function (subject to a few restrictions) [8].

Interestingly, there exists a combinatorial connection between Stackelberg Pricing Games and the Price Setting Problem from Chapter 2. Recall the notion of simultaneous feasibility from Proposition 2.1.5: If θ and θ' are feasible assignments that differ by a single reassignment, then there exists a pricing vector π that is feasible for both θ and θ'. There turns out to be an analogous result for StackMST, where the underlying optimization problem solved by the followers is the Minimum Spanning Tree Problem (see [44] for the current state of the art). Here, we say that a tree T is *feasible* if there exists a feasible price vector for which T is an MST. Then for StackMST, we can say:

Theorem 3.2.4 (Myklebust, Sharpe, Tunçel, 2011 [48]). *Suppose* T, T' *are feasible trees for an instance of* StackMST*, and suppose* $|T \setminus T'| = 1$*. Then there exists a price vector* π *for the edges such that* T, T' *are both minimum-cost spanning trees with respect to* π*.*

3.3 Bilevel Mixed Integer Linear Programs

Expanding on the idea of leader-follower games, we can consider a highly generalized scenario known as *Bilevel Linear Programming*. This setting encompasses two-stage decision problems, with a leader and follower, where some or all of the decision variables are integer. This setting is highly adaptable and has been studied in many different settings and applications, including vehicle routing and transportation. Bilevel formulations are also especially well-suited for modeling pricing problems [19] [43]. For a detailed history of the topic and examples of its applications, we refer the reader to the survey [52].

To demonstrate an example of a bilevel model, we review the problem studied in [41]. In any bilevel program, the leader first decides the variables z, then the follower chooses their own variables x.

The goal of the leader is to optimize over (x, z) subject to joint constraints. Additionally, when x has multiple optimal solutions, we choose one that has the optimal outcome for the leader, similar to our established tiebreaking conventions. Given these assumptions, we can define the following *Bilevel Linear Program*:

$$\inf_{x,z} \ c^T x + e^T z, \tag{3.2}$$

$$\text{s.t. } Cx + Dz \leq p, \ z \in \mathbb{R}^d, \tag{3.3}$$

$$x \in \arg\min_{x'} \{g^T x' : Ax' \leq Bz + u, \ x' \in \mathbb{R}^n\}. \tag{3.4}$$

The follower's (or *lower level* problem (3.4)) is an example of a *parametric linear program* with the right hand parameterized by z. Unsurprisingly, this problem is also known to be $\mathcal{NP}$-Hard - in fact, if z is allowed to be continuous, it is not even guaranteed that the set of all *bilevel feasible solutions*

$$\mathcal{F} := \{(x, z) \in \mathbb{R}^{n+d} : (3.3), (3.4)\}$$

is closed [56]. Thus, we must express the objective function (3.2) as an infimum rather than a minimum. However, if z is integer and the instance is feasible, we may replace "inf" with "min" [56]. Other types of bilevel programs exist, in the same way that many different types of linear programs exist. For example, we may formulate a *Bilevel Mixed Integer Linear Program* (BMILP) where some or all of the leader and follower's variables must be integer. In particular, consider the BMILP

$$\inf_{x,z} \ c^T x + e^T z, \tag{3.5}$$

$$\text{s.t. } Cx + Dz \leq p, \ z \in \mathbb{R}^d, \ z_i \in \{0, 1\}, \ \forall i \in I,$$

$$x \in \arg\min_{x'} \{g^T x' : Ax' \leq Bz + u, \ x' \in \mathbb{R}^n\},$$

where I is an indexing set indicating the subset of z variables that must be binary. We would like to point out that (3.5) is closely related to the Maximum Utility Problem formulation (1.4). While MUP cannot itself be expressed as a BMILP, as the prices are not required to be set after an assignment is determined, the motions of fixing assignments and solving the Price Setting Problem resemble these bilevel actions.

Ultimately, as (3.2) and (3.5) are incredibly general, much of the research on these formulations has been done on special cases. In the paper [41], the authors are interested in a specific BMILP where the leader's variables $z \in \mathbb{R}^d$ are continuous, the *follower's* variables $x \in \mathbb{Z}^n$ are integer, and all other data are integer:

$$\inf_{x,z} \ c^T x + e^T z, \tag{3.6}$$

$$\text{s.t. } Cx + Dz \leq p, \ z \in \mathbb{R}^d,$$

$$x \in \arg\min_{x'} \{g^T x' : Ax' \leq Bz + u, \ x' \in \mathbb{Z}^n\}.$$

They also assume that the set $\mathcal{F}$ corresponding to (3.6) is bounded. Similarly, if we assume that for every $z \in \text{proj}_z P$, where

$$\text{proj}_z P = \{z \in \mathbb{R}^d : (x, z) \in P \text{ for some } x \in \mathbb{R}^n\},$$

the lower level optimization problem (3.4) has a bounded feasible region, then:

Theorem 3.3.1 (Köppe, Queyranne, Ryan, 2010 [41])**.** *Given an instance of the Bilinear Mixed Integer Linear Program* (3.6)*, there exists an algorithm which:*

a. *decides if the instance is feasible;*

b. *if it is feasible, decides if the infimum is attained; and*

c. *if so, finds an optimal solution.*

The algorithm runs in polynomial time when the follower's dimension n is fixed.

Their results are also extended to the case when the leader's variables are required to be integral, i.e. $z \in \mathbb{Z}^d$. In this case, they give an algorithm that runs in polynomial time when the total dimension $d + n$ is fixed.

3.4 Bilinear Mixed Integer Programs

There exists one further generalization we can make beyond our bilevel mixed integer linear program formulation. While the bilevel formulation implies a scenario where leaders and followers act separately, we can envision a scenario where two parties act simultaneously. This phenomenon can be modeled through what we refer to as a *bilinear* constraint:

Definition 3.4.1. *A* Bilinear Constraint *is an inequality of the form*

$$\pi^T M \theta \leq \kappa, \tag{3.7}$$

where M is an $n \times m$ matrix, $\pi \in \mathbb{R}^n, \theta \in \mathbb{R}^m$, and $\kappa \in \mathbb{R}$. The values for M and κ are received as data, while π and θ are variables. Moreover, M is subdivided in the following way:

$$M := \begin{bmatrix} \gamma & q^T \\ r & Q \end{bmatrix},$$

where γ is a single entry, $q \in \mathbb{R}^{n-1}$, $r \in \mathbb{R}^{m-1}$, and $Q \in \mathbb{R}^{(m-1)\times(n-1)}$.

Now that we have defined a bilinear constraint, consider the vectors

$$\bar{\pi} = \begin{bmatrix} 1 \\ \pi \end{bmatrix}, \quad \bar{\theta} = \begin{bmatrix} 1 \\ \theta \end{bmatrix}.$$

Then, multiplying $\bar{\pi}$ and $\bar{\theta}$ by M, we have the following:

$$\bar{\pi} M \bar{\theta} = \begin{bmatrix} 1 & \pi^T \end{bmatrix} M \begin{bmatrix} 1 \\ \theta \end{bmatrix} = \gamma + r^T \pi + q^T \theta + \pi^T Q \theta$$

$$= \gamma + \sum_{i=1}^{m} r_i \pi_i + \sum_{j=1}^{n} q_j \theta_j + \sum_{i=1}^{m} \sum_{j=1}^{n} Q_{ij} \pi_i \theta_j. \tag{3.8}$$

We can see that this bilinear constraint format gives us a tremendous amount of modeling power. The inequality format (3.7) can be used to model not only any linear inequality, but also any degree-two inequalities, where π and θ entries are multiplied. We may do this as long as the matrix M is designed properly and the linear equations $\pi_1 = 1, \theta_1 = 1$ are included. In fact, (3.7) can even

be used to model *quadratic* inequalities - simply apply the constraint $\pi_i = \theta_j$ for some i, j. Then we can model the variables π_i^2 or θ_j^2 by setting the entry Q_{ij} in some constraint matrix M to be nonzero. In particular, notice that we can also use bilinear constraints to model binary integrality constraints, since for any variable θ_i,

$$\theta_i \in \{0, 1\} \iff \theta_i^2 - \theta_i = 0.$$

As a consequence, we can also use this technique to model any bounded integer variable, as given some integer variable $0 \leq x \leq N$, we can express it as the sum of $\lfloor \log(N) \rfloor$ binary variables [58]. Putting all of these observations together, we can formulate a *Bilinear Mixed Integer Program*:

$$\begin{aligned} \max \pi^T M^0 \theta, & \qquad (3.9) \\ \text{s.t. } \pi^T M^i \theta \leq \kappa_i, \ i \in \{1, \ldots, \ell\}, & \\ \theta \in \mathbb{R}^n, & \\ \pi \in \mathbb{R}^m, & \end{aligned}$$

where M^i is an $m \times n$ matrix, π and θ are column vectors, and $\kappa \in \mathbb{R}^\ell$.

The formulation (3.9) is incredibly general. Already, we have seen that it can be used to model any linear, quadratic, or mixed integer program with bounded integer variables. We may also use this framework to examine general multi-decision optimization problems where agents solve a variety of underlying combinatorial problems, such as perfect matching, minimum spanning tree, or shortest paths, while also allowing participants to act simultaneously, rather than in sequence as they do in bilevel programs. As a matter of fact, our bilinear mixed integer programming formulation is at least as general as our formulation for bilevel programs. In the following theorem, we give a proof for the bilevel linear programming formulation (3.2). However, the proof is easily modified to include the formulation (3.5), by including binary constraints for the members of I.

Theorem 3.4.2. *Let* BLP *be an instance of* (3.2) *where the bilevel feasible solution set $\mathcal{F}$ is closed. Then* BLP *can be formulated as a bilinear mixed integer program.*

First, we prove a useful lemma:

Lemma 3.4.3. *There exists a system of bilinear constraints whose feasible solution set fully characterizes the set of optimal solutions to the lower level problem* (3.3) *of* BLP.

Proof. We recall the lower-level optimization problem (3.3) of BLP:

$$\begin{aligned} \min g^T x, & \qquad (3.10) \\ \text{s.t. } Ax \leq Bz + u, & \qquad (3.11) \\ x \in \mathbb{R}^n. & \end{aligned}$$

Then, notice that if the upper level variables z are treated as constants, the entire right hand side of (3.11) becomes a constant as well. Thus, we may express it as a column vector v, and the problem becomes:

$$\begin{aligned} \min g^T x, & \qquad (3.12) \\ \text{s.t. } Ax \leq v, & \qquad (3.13) \\ x \in \mathbb{R}^n. & \end{aligned}$$

Because (3.12) is a linear program, its dual exists and can be expressed as:

$$\max v^T y, \tag{3.14}$$
$$\text{s.t. } A^T y \geq g, \tag{3.15}$$
$$y \in \mathbb{R}^m.$$

Now, notice that if we force both sets of constraints (3.13) and (3.15) at the same time, then the set of feasible x and y values simply characterize the set of all feasible primal and dual solutions to (3.12) and (3.14). However, if we *also* force the constraint

$$g^T x - v^T y = 0, \tag{3.16}$$

then by the Strong Duality Theorem, the *only* solutions that are feasible all of (3.12), (3.14) and (3.16) are the values x^* and y^* that are *optimal* for (3.12) and (3.14). Furthermore, this solution x^*, y^* also optimizes (3.10), assuming z is fixed. Thus, we can formulate the system

$$\begin{aligned} g^T x - v^T y &= 0, \\ Ax &\leq v, \\ A^T y &\geq g, \\ x \in \mathbb{R}^n, y &\in \mathbb{R}^m, \end{aligned}$$

whose feasible solution set fully characterizes the set of all optimal solutions to the lower-level problem (3.10). Now, all we need to do is write the above system using bilinear constraints. For every constraint, we will define a matrix that is in the same form as in Definition 3.4.1. Then, we simply define the entries of the matrix so that when the expansion in (3.8) takes place, our desired constraints appear.

To do this, we must first compress x and y into a single vector $w = [w_0, x, y]$ of length $n + m + 1$. Thus, we have that $x = [w_1, \ldots, w_n]$ and $y = [w_{n+1}, \ldots, w_{n+m}]$. Now, to model the equality constraint, we create a matrix M^*, where $r^* = [g, -v]$ and the entries of M^* are 0 everywhere else. For the set of constraints $Ax \leq v$, for every row $1 \leq i \leq m$ of A, we create a matrix M^i such that $r^i = [A_i, 0^m]$ and $\gamma^i, q^i, Q^i = 0$. Similarly, for the constraints $A^T y \geq g$, we create a matrix M^j for every row $1 \leq j \leq n$ of A^T such that $r^j = [0^n, A_j^T]$ and $\gamma^i, q^i, Q^i = 0$. Then, using the variables z from the upper level constraints, we can write the new set of lower level constraints as:

$$\begin{aligned} w^T M^* z &= 0, && \\ w^T M^i z &\leq v_i, && \forall i \in \{1, \ldots, m\}, \\ w^T M^j z &\geq g_j, && \forall j \in \{1, \ldots, n\}, \\ w_0, z_0 &= 1, && \\ w &\in \mathbb{R}^{n+m+1}. && \end{aligned} \tag{3.17}$$

We note that using this framework, an assumption that the lower level problem (3.10) is feasible or bounded is not necessary. In the case that the primal for the lower level problem (3.12) is infeasible and the dual (3.14) is unbounded, then the system (3.17) is also infeasible, given fixed z. The same is true if the dual is infeasible and the primal unbounded. Thus, the system (3.17), along with the upper level constraints, fully characterizes the set of all feasible solutions for the bilevel program (3.2). □

Now, we can complete the proof of Theorem 3.4.2.

Proof. Let our bilevel program BLP be:

$$\begin{aligned} \min_{x,z}\ & c^T x + e^T z, \\ \text{s.t. } & Cx + Dz \le p,\ z \in \mathbb{R}^d, \\ & x \in \arg\min_{x'}\{g^T x' : Ax' \le Bz + u,\ x' \in \mathbb{R}^n\}. \end{aligned}$$

All that remains to do is model the objective function and upper level constraints. For the objective function, we define a matrix M^0 of size $(n+m+1) \times (d+1)$, where

$$r^0 = [c, 0^m], q^0 = e, \gamma^0 = 0, Q^0 = 0.$$

Then, for each upper level constraint $1 \le k \le \ell$, we create a matrix M^k, where

$$r^k = [C_k, 0^m], q^k = D_k, \gamma^k = 0, Q^k = 0.$$

Then, we can write BLP as:

$$\begin{aligned} \min_{w,z}\ & w^T M^0 z, & \\ \text{s.t. } & w^T M^k z \le p_\ell, & \forall k \in \{1,\dots,\ell\}, \\ & w^T M^i z \le v_i, & \forall i \in \{1,\dots,m\}, \\ & w^T M^j z \ge g_j, & \forall j \in \{1,\dots,n\}, \\ & w^T M^* z = 0, & \\ & w_0, z_0 = 1, & \\ & w \in \mathbb{R}^{n+m+1}, z \in \mathbb{R}^{d+1}. & \end{aligned}$$

□

To conclude the chapter, for completeness and to further demonstrate the modeling power of bilinear mixed integer programs, we present an example where we write the Maximum Utility Problem in terms of our BMIP formulation (3.9).

Example 3.4.4. *First, we use an alternative formulation for* MUP *from [16], with some re-indexing:*

$$\max \sum_{i=1}^{n}\sum_{j=1}^{m+1} N_i \pi_j \theta_{ij}, \tag{3.18}$$

$$s.t. \sum_{j=1}^{m+1} (R_{ij} - \pi_j)\theta_{ij} \ge R_{ik} - \pi_k, \qquad \forall i,k,$$

$$\sum_{j=1}^{m+1} \theta_{ij} = 1, \qquad \forall i, \tag{3.19}$$

$$\pi_1 = 0,$$

$$\pi_j \ge 0, \qquad \forall j,$$

$$\theta_{ij} \in \{0,1\}, \qquad \forall i,j, \tag{3.20}$$

where there are n customer segments, $m+1$ products (including the dummy product, now indexed from 1 to $m+1$), π is a row vector of length $m+1$ and θ is a row vector of length $n(m+1)$. Here we abuse notation, where θ_{ij} refers to the $i(j-1)+j^{th}$ entry in the row vector θ. Our next step is to define the matrices M for each constraint and the objective function. Let's begin with M^0. We have:

$$M^0 = \begin{bmatrix} \gamma^0 & (q^0)^T \\ r^0 & Q^0 \end{bmatrix},$$

where $\gamma^0, q^0, r^0 = 0$ and $M^0_{ij} = N_i$ for all $2 \le i \le m+1, 2 \le j \le n(m+1)$.

Our next task is to define the matrix M^{ik} for every product k. Let:

$$M^{ik} = \begin{bmatrix} \gamma^{ik} & (q^{ik})^T \\ r^{ik} & Q^{ik} \end{bmatrix}.$$

For a given $1 \le j \le n(m+1)$, define $\ell = j \pmod{m+1}$. Then we have the following:

$$\begin{aligned}
\gamma^{ik} &= 0, \\
r^{ik}_j &= \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{otherwise,} \end{cases} \\
q^{ik}_j &= \begin{cases} R_{i\ell}, & \text{if } (i-1)(m+1) < j \le i(m+1), \\ 0, & \text{otherwise,} \end{cases} \\
Q^{ik}_{j_1 j_2} &= \begin{cases} -1, & \text{if } j_1 \equiv j_2 \pmod{m+1},\ (i-1)(m+1) < j \le i(m+1), \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

For constraint (3.19), for every i, we have the matrix M^i, where $\gamma^i, r^i, Q^i = 0$ and

$$q^i_j := \begin{cases} 1, & \text{if } (i-1)(m+1) < j \le i(m+1), \\ 0, & \text{otherwise.} \end{cases}$$

We then have the nonnegativity constraint on π and the $\{0,1\}$ constraint on θ. Finally, we set $\pi_1 = 0$ to represent the dummy node, and additionally $\pi_0 = 1, \theta_0 = 1$ so that the expansion in (3.8) is present. Thus, we can construct our bilinear program as in (3.9):

$$\begin{aligned}
&\max \pi^T M^0 \theta, \\
&s.t.\ \pi^T M^{ik} \theta \ge R_{ik}, && \forall i, k, \\
&\qquad \pi^T M^i \theta = 1, && \forall i, \\
&\qquad \pi_1 = 0, \\
&\qquad \pi_0, \theta_0 = 1, \\
&\qquad \pi \ge 0, \\
&\qquad \theta \in \{0,1\}^{n(m+1)}.
\end{aligned}$$

Chapter 4

Worst-Case Algorithmic Analyses on Practical Instances

In Chapter 3, we reviewed the existing literature that discusses structural scenarios under which the Maximum Utility Problem can be solved exactly. This chapter continues said theme, presenting additional potential structures our data may take and investigating the algorithmic consequences of these instances. The papers [17], [48], [16], and [51], which study the general formulation (1.4) of the Maximum Utility Problem, all extend the majority of their efforts to the case when N_i and R_{ij} can take any uniformly random value in $\mathbb{Z}^+$. However, this assumption is far too general in practice. As an example, consider the worst-case approximation bound for Guru. The instance presented in Example 2.3.1 sets $N_i = 2^{i-1}$ for all i. However, for values of n that are present in industry applications (ex. $n \geq 1000$), the values of N_i in this example easily exceed the number of atoms in the universe, and the range of R_{ij} values is similarly exponential. In practical applications, though, the ratio between $\bar{R}_1$ and $\bar{R}_n$ can often be bounded by a reasonably small constant. To cite some specific price ranges that appear in real data (prices are as of July 2022, in CAD):

- The **Toronto Blue Jays** [33] single-game ticket prices available on ticketmaster.ca typically range from \$20 through \$130.
- **Arc'teryx** [31] offers rain jackets priced between \$120 for basic necessities and $\$1,000$ for the most technical gear.
- **Toyota** [34] offers typical passenger vehicles at base prices ranging from $\$19,450$ (Corolla) through $\$48,290$ (Tundra).
- **Redfin.com** [32] lists 2+ bedroom/2+ bathroom homes in Kitchener, ON for prices between $\$390,000$ and $\$3,800,000$. However, individual customer price ranges for homes should be considered to be much more restricted than the minimum and maximum price of available homes.

While not a comprehensive review, these examples show that in practice, a multitude of instances will have data ranges that can be bounded by a reasonably small constant. The consequences of this are explored in Subsection 4.3.2.

Another complication within practical applications is that the number of customer segments and

products n and m may be significantly larger than the actual number of buyers and items. Given a small number of potential buyers, one may wish to model the behavior of multiple consumers through clustering them into several segments, or to model the ability to purchase multiple products or multiple copies of products by duplicating segments. Additionally, a large number of products may be used to simulate offering different bundles of products. Consider the example of a tourism company [55]. They offer only a modest number of single items, such as 20 destination cities, 7 different departure/return days, and 8 different hotel choices per destination. The company may choose to offer 3-day, 7-day, 14-day, etc. vacation packages, with transportation tickets, accommodations, and other bells and whistles included. Moreover, these packages may be offered for each combination of individual items. In this case, given the vast number of possibilities for the content of a package, the number of product bundles can easily be in the thousands or even millions. Thus, many of the algorithms and heuristics for the Maximum Utility Problem are intended for use on these large-scale instances. In Section 4.1, we see some examples of this principle in practice, reviewing the types of data instances that have been generated in the literature. Then, the bulk of this chapter discusses a variety of applicable settings under which we derive new algorithmic ideas. While for each considered scenario we detail some reasons why they are interesting and potentially applicable in economic environments, we note that it is not known whether these cases commonly arise specifically for the Maximum Utility Problem. This is also the case for the polytime scenarios mentioned in Chapter 3.

4.1 Data Generation

A key focus throughout this thesis is discussing the types of data patterns that are likely to be present in practical applications. As such, we use this section to briefly discuss the data that has been generated for computational experiments in the literature. Ideally, any company utilizing these models also has access to extensive data mining techniques they can use to determine the reservation prices, the structure of the customer segments, and the utility tolerances. This includes personal information such as demographics, location, and purchasing patterns, to name a few categories. One needs look no further than targeted advertising to see a modern example of intricate data collection being used in direct-to-consumer optimization models. For example, Demand-Side Platforms (DSP) are companies that partner with advertisers to deliver content towards consumers. Given large amounts of customer data and products to advertise, the DSP participates in online auctions to bid on slots that consumers can see. As there is significant competition for these slots, the question of how much to bid given the parameters of each auction gives rise to an optimization problem [25].

Absent full sets of data, we must have a method for filling in the missing attributes. Possible choices include using *conjoint analysis* (see [40] for an example) or multinomial logit models [37]. Another commonly used implementation is to probabilistically generate customer preferences given some set of raw customer data, computing mean and variance from known attributes and using filling in the R_{ij} values according to some distribution [54]. In the literature that discusses the Maximum Utility Problem explicitly, there are several different test datasets that are generated. The papers [50], [48], and [51] all use data generated from historical vacation package sales, in partnership with a company in tourism. While this data is protected, these papers also generate random datasets:

- [51] generates test cases for each pair $(n, m) \in \{2, 5, 10, 20, 40, 60, 80, 100\}^2$, with uniformly random values of R_{ij} between 29 and 210, and values of N_i between 500 and 799.
- [48] generates a more aggressive dataset of instances with the same R_{ij} value range, but with $m \in \{5000, 10000, 20000, 40000\}$ and $n \in \{200, 400, 600, 100\}$.
- The original paper by Dobson and Kalish [17] uses three different "product-classes", each describing a different possible real-world scenario (for example, in the first we assume that each segment has the same order of preference among all the products). For each scenario, they generate 40 different random matrices with 5 segments and 4 products.

Finally, [50] provides an explicit methodology for directly generating some of the R_{ij} values used in their computational experiments. Using historical data, the fraction f_{ij} of customers from each segment i that purchased each product j, and the price p_{ij} they paid for it, is known. Then, to estimate R_{ij} for each i and j, they assume each segment acts according to a *share-of-surplus* [42] model. Letting B_i be the set of products purchased by Segment i, we have the approximation

$$f_{ij} \approx \frac{R_{ij} - p_{ij}}{\sum_{k \in B_i}(R_{ik} - p_{ik})}.$$

To generate approximate R_{ij} values for $j \in B_i$ for all i and fit the model to the p_{ij} data, they use least-squares regression, and suggest using the techniques mentioned above for filling in data to define the remaining R_{ij} entries.

4.2 Multiplicative Utility Tolerances

As we saw in Section 2.2, the introduction of utility tolerances adds a significant amount of robustness to the model. Expanding on this concept, rather than having a single utility tolerance for each customer segment, we may instead define a constant $\delta_{ijk} > 0$ for every segment i and pair of products j, k. The additional customization allows for more flexibility during implementation, as we now have the ability to model interactions between products. Additionally, the δ_{ijk} parameters are useful for representing the preferences of Segment i among all products. While this simple change can be made and implemented in the model (2.6), here we present an alternative formulation where we model δ as a multiplicative scalar instead of as an additive scalar.

Beyond flexibility, one particular advantage of using utility tolerances as multiplicative scalars rather than additive scalars that it allows the δ values to easily be defined by a percentage of a segment's reservation prices rather than by an additive constant. One disadvantage, however, is that the model assumes a higher degree of data collection (in that we assume, when solving the model, that the δ values are data that must be collected by the user, and not determined during direct implementation). However, for any given i, it is possible to simply derive the δ_{ijk} values as a function or as a consequence of Segment i, without taking into consideration any aspects of the products j or k. In this way, we do not necessarily require any more data collection than in the model (2.6).

To define the new model, we must first redefine the customer segment's purchasing requirements. Instead of the inequalities (2.4) and (2.5), we now assume that Segment i buys Product j if and only if

$$R_{ij} - \pi_j \geq R_{ik}\delta_{ijk} - \pi_k, \ \forall k \neq j$$

and

$$R_{ij} - \pi_j \geq CS_i \delta_{ij0}, \ \forall i, j,$$

where 0 again represents the "dummy" product. Preprocessing such that $R_{ij} \leftarrow \max(0, R_{ij} - CS_i\delta_{ij0})$ and $CS_i \leftarrow 0$, the problem (2.8) becomes:

$$\begin{aligned}
\max & \sum_{i=1}^{n}\sum_{j=1}^{m} N_i p_{ij}, && && (4.1)\\
\text{s.t. } & R_{ij}\theta_{ij} - p_{ij} \geq (R_{ik}\delta_{ijk} - \pi_k)\theta_{ij}, && \forall i, j, \ \forall k \neq j,\\
& R_{ij}\theta_{ij} - p_{ij} \geq 0, && \forall i, j,\\
& \sum_{j=1}^{m} \theta_{ij} \leq 1, && \forall i,\\
& p_{ij} \leq \pi_j, && \forall i, j,\\
& p_{ij} \geq \pi_j - \tilde{R}_j(1 - \theta_{ij}), && \forall i, j,\\
& \pi_j, p_{ij} \geq 0, && \forall i, j,\\
& \theta_{ij} \in \{0, 1\}, && \forall i, j.
\end{aligned}$$

The LP (2.1) becomes:

$$\begin{aligned}
\max & \sum_{j=1}^{m} M_j \pi_j, && && (4.2)\\
\text{s.t. } & \pi_j - \pi_k \leq \min_{i \in C_j}\{R_{ij} - R_{ik}\delta_{ijk}\}, && \forall j \in B, \forall k \neq j,\\
& \pi_j \leq \min_{i \in C_j}\{R_{ij}\}, && \forall j \in B.
\end{aligned}$$

The dual of (4.2) is identical to (2.3), where the variables in (2.2) are changed to:

$$r_{jk} := \min_{i \in C_j}\{R_{ij} - R_{ik}\delta_{ijk}\}, \qquad \sigma_j := \min_{i \in C_j}\{R_{ij}\}.$$

Note that the value inside the brackets of σ_j can be turned into $R_{ij}\delta_{ij0}$ by preprocessing such that $R_{ij} \leftarrow \frac{R_{ij}}{\delta_{ij0}} - CS_i\delta_{ij0}$.

4.3 Logarithmic Scaling in Data

Now that we have established a framework for considering special structures of the underlying reservation price data, we can discuss the algorithmic consequences in some specific instances. This section involves two different methods of scaling - in the first, we consider a rounding scheme where we round our data to the nearest power of c for some $c \in \mathbb{R}^+$. In the second, we study a more general case where $\bar{R}_i$ values are uniformly separated on a logarithmic scale.

The motivation for considering rounding to a nearest power is connected to several areas of combinatorial optimization and operations research. From combinatorial optimization, we have the

technique of *capacity scaling*. In some problems, we can often arrive at efficient solutions by acting in p different stages, where in each stage we only examine elements with a cost function bounded by some function of 2^p; for example, an algorithm for integral-capacity maximum flow that, in each successive stage p, only considers augmenting paths of width at least 2^p. Another example of scaling is the celebrated *Successive-Scaling Algorithm* of Edmonds and Karp [20] for finding minimum-cost flows, which relies on solving several successively scaled down instances of a problem to produce a polynomial-time algorithm. In the subfield of operations research known as *Inventory Management*, the notion of *power-of-two* policies refers to models where product orders take place at time intervals that scale by some factor of 2. Popular examples of power-of-two policies include scheduling on production machines and multi-staged decision making in manufacturing plants (see Chapter 3 of [47]).

These examples motivate our study of rounding and later considering the application where $\bar{R}_i$ values scale logarithmically. The reason we are interested in this second scenario is related to our discussion of how R_{ij} values can be generated in the first place - in practice, it can be common to have large amounts of reservation price data defined by a relatively small number of parameters (see, for instance, the discussion in Section 6.2 of [48]). As such, we do not always have a complete definitive set of R_{ij} values. In order to obtain our matrix of values, then, we require techniques for generating reservation prices based off the small number of parameters at out disposal. As discussed in Section 4.1, there exist several popular methods to fill these values in given limited data. Thus, it is important to study instances of the Maximum Utility Problem where the reservation price matrix has some discernible structure - if there exists a harmony between these structural properties, efficient techniques for them, and R_{ij} data generation, then such techniques will be highly useful in applications.

4.3.1 An Attempt at Rounding

Consider an instance of (4.1) with n customer segments, m products, and reservation prices R_{ij}. Then, consider the alternative set of reservation prices, where for every R_{ij} and $c \in \mathbb{R}^+$, we set:

$$R'_{ij} = \begin{cases} 0, & \text{if } R_{ij} < \frac{1}{c}, \\ c^k, & \text{if } R_{ij} \geq \frac{1}{c}, \end{cases}$$

where

$$k = \arg\min_{x \in \mathbb{N}} \left(|c^x - R_{ij}| \right).$$

i.e. the value of R_{ij} is rounded to the nearest power of c, or 0 if it is sufficiently small. One obvious benefit of considering these alternate reservation prices is the possibility of a simpler and easier to analyze algorithm. The difficulty, of course, is in relating the modified solution to the solution of the original problems.

We first observe that if there are n customers and m products, then the number of possible values of R_{ij} is upper bounded by nm. Thus, when fixing an assignment θ, there are $O(n^2m^2)$ possible values for the edge prices r_{jk} in the underlying shortest paths graph, as each r_{jk} must be the difference between reservation prices $R_{ij} - R_{ik}$ for some customer segment i. This immediately implies a bounded number of objective values, which implies that we can bound the number of iterations of the Dobson-Kalish algorithm. However, using the rounding scheme is not necessary for this fact, and we can prove a much more general version:

Theorem 4.3.1. *Suppose that for all i, j, k, the input data N_i, R_{ij}, and δ_{ijk} are rational for* (4.1)*. Then the Dobson-Kalish Algorithm has a pseudopolynomial running time.*

We will return to the proof Theorem 4.3.1 at the end of this section, after finishing our discussion of rounding. Ultimately, this simple rounding scheme is not a clearly beneficial technique in that it fails to retain key structural aspects of the original instance. The following example shows there may be a large gap between "true" and "modified" profit:

Example 4.3.2. *Let $m = 1$ and $c = 2$. Then for every customer segment $i \leq n$, let $R_{i1} = 0.5 - \epsilon$ for $\epsilon > 0$. Then $R'_{i1} = 0$, for every i, and the modified profit will always be 0. However, the optimal price for Product 1 with original reservation prices is $0.5 - \epsilon$, giving a profit of $n(0.5 - \epsilon)$, so the gap is $\Omega(n)$.*

In fact, the profit gap extends to the case where the R_{ij} values are integer. For the rounding scheme where we round to the nearest power of 2 for each R_{ij}, consider hgggggthe instance of one customer and two products. Let $R_{11} = 3, R_{12} = 2^k$ for any k. Then $R'_{11} = 2, R'_{12} = 2^k$. Then we let $\pi = [2, 2^k]$. Then assigning Segment 1 to Product 2 is feasible with alternate reservation prices, (and in fact is the assignment obtained by running the MaxR heuristic), but infeasible with true reservation prices. Let $\theta(\pi)$ denote the "true" assignment with the vector π, and $\theta'(\pi)$ denote the assignment with modified reservation prices. Then the difference in profit between $\theta'(\pi)$ and $\theta(\pi)$ is $2^k - 2$, which we can make as large as we want, scaling k.

This counterexample can be achieved within the Dobson-Kalish algorithm explicitly. Add an additional customer segment with $R_{21} = 2, R_{22} = 0$. With rounded reservation prices, the MaxR heuristic will assign Segment 1 to Product 2, and Segment 2 to Product 1. The initial shortest paths solution will give Product 1 a price of 2, and Product 2 a price of 2^k. This is optimal (since Segment 2 will never buy Product 2), so the algorithm terminates. However, given $\pi = [2, 2^k]$, both segments will in fact purchase Product 1, giving a difference in profit of $2^k - 2$, which scales with k.

These results suggest we will need a more nuanced rounding approach that can predict or mitigate the reassignment of customer segments once the algorithm terminates. It is likely that any simple rounding scheme will run into a version of this problem.

Returning to the proof of Theorem 4.3.1, we must first tackle the following Lemma:

Lemma 4.3.3. *Assume that for an instance of* (4.1) *all input data is integer. Let $SP(m)$ denote the computational time it takes to solve one shortest paths problem on a graph with $O(m)$ edges. Then the runtime of the Dobson-Kalish Algorithm is at most*

$$O\left(SP(m) \cdot n \cdot \sum_{i=1}^{n} N_i \bar{R}_i\right).$$

Proof. Consider the possible values for edges in the underlying shortest-paths problem (4.2):

$$r_{jk} := \min_{i \in C_j} \{R_{ij} - R_{ik}\delta_{ijk}\}, \qquad \sigma_j := \min_{i \in C_j} \{R_{ij}\}.$$

Because the values of R_{ij} and δ_{ijk} are all integer, the values of r_{jk} and σ_j must also be integer. Then when solving (4.2) during an iteration of DK88, the resulting price of every product must also be an

integer, as their prices are defined by the shortest path from the dummy node to the product node. Thus, for any potential solution encountered by DK88, the objective value $\sum_{i=1}^{n}\sum_{j=0}^{m} N_i\pi_j\theta_{ij}$ must be integer, since prices and N_i values are integer.

Furthermore, due to steps 12 and 13 of Algorithm 1, the Dobson-Kalish heuristic *only* iterates if it is able to find a new assignment that strictly improves the objective value. Since every solution considered by DK88 has an integer-valued objective value, it follows that after each iteration the value of the current best solution increases by at least 1. Because every solution has a value at least 0 and at most $\sum_{i=1}^{n} N_i\bar{R}_i$, there can be at most $\sum_{i=1}^{n} N_i\bar{R}_i$ iterations before DK88 terminates. Finally, DK88 solves $O(n)$ shortest paths problems which each take time $SP(m)$ to solve. Thus, DK88 has a runtime at most

$$O\left(SP(m)\cdot n\cdot\sum_{i=1}^{n} N_i\bar{R}_i\right),$$

as desired. □

Now, we can prove Theorem 4.3.1.

Proof. Let R_{ij}, N_i, and $\delta_{ijk} \in \mathbb{Q}^+$ for all i, j, k. Let q be the least common multiple of the denominators of all numbers in the data, which exists as all data are rational. Then, if we multiply all data values by q, the resulting new data values will all be integer. Applying Lemma 4.3.3, we see that the new instance will have a running time of

$$O\left(SP(m)\cdot n\cdot q^2\sum_{i=1}^{n} N_i\bar{R}_i\right).$$

The inclusion of q^2 in the runtime bound is due to the scaling of data values. If we multiply all R_{ij} values and N_i values by q, then the new resulting upper bound for total revenue is $\sum_{i=1}^{n} qN_iq\bar{R}_i = q^2\sum_{i=1}^{n} N_i\bar{R}_i$. However, since is q^2 is bounded by a polynomial function of the size of the input data, the runtime is still pseudopolynomial. □

Corollary 4.3.4. *Assume that in an instance of* (4.1)*, for all i, j, k, the data values R_{ij}, N_i, and δ_{ijk} are integer multiples of some $\gamma \in \mathbb{Z}^+$. Then the runtime of the Dobson-Kalish Algorithm is*

$$O\left(SP(m)\cdot n\cdot\frac{\sum_{i=1}^{n} N_i\bar{R}_i}{\gamma}\right).$$

Proof. Since all data are integer multiples of γ, all edge costs must also be integer multiples of γ. Thus, after each iteration of DK88, the value of the current best solution must go up by at least γ, and the runtime bound follows. □

4.3.2 Uniformly Logarithmic Separation in Data

As we saw above, when rounding to nearest powers of c, it is difficult to retain the original structure of any given instance. However, in the particular instance where out original data follows a logarithmic structure, we can avoid rounding entirely. We recall the motivation for considering this scenario from above - often times, we may be given a few parameters with which to estimate

our data, and are responsible for using various techniques to fill out remaining entries. One such possible scenario is that where we have data that tells us the maximum and minimum reservation prices for the *entire* set of customers - in other words, we know the general range of values that our customer's reservation prices lie in. As we saw in the introduction to this chapter, it is not uncommon in practical applications for this ratio to be relatively small (e.g. a factor of 10).

Formally, say that we have n customer segments and we assume without loss of generality that $\bar{R}_1 \geq \dots \bar{R}_n$, and only $\bar{R}_1 = u$ and $\bar{R}_n = \ell$ are known. Are there patterns our data may take that we can exploit? It turns out that if the $\bar{R}_i$ values follow a uniformly logarithmic scale, we can use Algorithm 3 (GURU) to get a good approximation ratio. Hence, we motivate the following three definitions:

Definition 4.3.5. *For $\alpha \leq 1$, we say that a set J of n customer segments is* α-dense *if*

$$\bar{R}_n \geq \alpha \bar{R}_1.$$

Definition 4.3.6. *For $\beta \leq 1$, we say that a set of n customer segments is β-uniformly dense if for all $\bar{R}_i, \bar{R}_{i+1}$, where $\bar{R}_i \geq \bar{R}_{i+1}$,*

$$\frac{\bar{R}_{i+1}}{\bar{R}_i} \geq \beta.$$

Definition 4.3.7. *For any customer segment i, let $S_i = \arg\max_j R_{ij}$ be the set containing all of their most preferred products. Then, let σ_i be the smallest j such that $j \in S_i$.*

To demonstrate the effectiveness of uniformly logarithmic scaling, let $\alpha = \frac{\ell}{u}$ and assume that for all i, the $\bar{R}_i$ values are uniformly spaced in a logarithmic scale, i.e. $\bar{R}_i = u\beta^{i-1}$ for the constant

$$\beta = \alpha^{\frac{1}{n-1}}.$$

Next, recall Theorem 2.3.2: if $\frac{\bar{R}_{i+1}}{\bar{R}_i} \geq 1 - \frac{k}{n-1}$, then the approximation ratio of GURU is $(k+1)$. Notice that in our current dataset, we have:

$$\frac{\bar{R}_{i+1}}{\bar{R}_i} = \frac{u\beta^i}{u\beta^{i-1}} = \beta.$$

Thus we satisfy the conditions necessary for a $(k+1)$-approx, for appropriate k. Having a robust terminology for these ratios is helpful, as it allows us to vary n and α and determine the resulting approximation ratio. To demonstrate, tables are presented for various values of α and n between 1/25 to 1/50 and 100 to 1000, respectively, with the resulting β and k values included, where k is (a rounded approximation of) the smallest number that satisfies the above inequality. Note that the resulting approximation ratio is $k+1$:

α	1/25	1/30	1/35	1/40	1/45	1/50	
β	0.96801	0.96623	0.96472	0.96342	0.96228	0.96126	(n=100)
k	3.167	3.343	3.492	3.621	3.731	3.836	

α	1/25	1/30	1/35	1/40	1/45	1/50	
β	0.98395	0.98305	0.98229	0.98163	0.98105	0.98053	(n=200)
k	3.193	3.372	3.524	3.655	3.770	3.874	

α	1/25	1/30	1/35	1/40	1/45	1/50	
β	0.99357	0.99321	0.99290	0.99263	0.99240	0.99219	(n=500)
k	3.209	3.390	3.543	3.675	3.792	3.897	

α	1/25	1/30	1/35	1/40	1/45	1/50	
β	0.99678	0.99660	0.99645	0.99631	0.99620	0.99609	(n=1000)
k	3.214	3.395	3.549	3.682	3.799	3.904	

Table 4.1: Smallest values of k for fixed α and n

For smaller α, the result is even more pronounced - for $\alpha = \frac{1}{5}$ and $n = 1000$, for example, we achieve an approximation ratio of roughly 2.6. In general, by plugging the identity for β into the approximation ratio (2.11), we see that when a set of n customer segments is α-dense and β-uniformly dense for $\beta = \alpha^{\frac{1}{n-1}}$, GURU has approximation ratio at most $n(1 - \alpha^{\frac{1}{n-1}}) + \alpha^{\frac{1}{n-1}}$. Another, more useful representation, is the following:

Theorem 4.3.8. *When a set of n customer segments is α-dense and β-uniformly dense for $\beta = \alpha^{\frac{1}{n-1}}$,* GURU *has a worst-case approximation ratio at most* $\ln(\frac{1}{\alpha}) + 1$.

Proof. Since the set of customer segments is $\alpha^{\frac{1}{n-1}}$-bounded, we have:

$$\alpha^{\frac{1}{n-1}} \geq 1 - \frac{k}{n-1},$$

which rearranges to

$$(n-1)\left(\alpha^{\frac{1}{n-1}} - 1\right) \geq -k.$$

We seek to find a value of k such that the inequality is always satisfied - since then by Theorem 2.3.2, GURU achieves a $k+1$-approximation. The limit of the left hand side as $n \to \infty$ is $\ln(\alpha)$, and since the left hand side is a decreasing function, we have that

$$(n-1)\left(\alpha^{\frac{1}{n-1}} - 1\right) \geq \ln(\alpha) \geq -k$$

for all $n \in \mathbb{N}$. Thus, fixing α, the inequality is always satisfied when $k \geq -\ln(\alpha) = \ln(\frac{1}{\alpha})$, as desired. □

Applying the exp() function to both sides also gives the following fact:

Corollary 4.3.9. *For fixed k,* GURU *achieves a $(k+1)$-approximation when a set of customer segments is e^{-k}-dense and β-uniformly dense for $\beta = \alpha^{\frac{1}{n-1}}$.*

In practice, this is quite powerful. For example, to obtain a guaranteed 3-approximation using GURU, we only require that our data is $\frac{1}{7}$-dense. This assumption is not too prohibitive in practice, as we saw in the examples presented during the introduction to this chapter.

By identifying even more additional structure in the data, when present, we can guarantee large amounts of revenue increases by applying the Π function to our assignments. A specific example of this follows below.

Theorem 4.3.10. *Let $n \leq m$ and our data be β-uniformly dense for some $\beta \leq 1$. Additionally, assume that for all i, $N_i = 1, |S_i| = 1$, and that for all segments i, i', $S_i \neq S_{i'}$. Furthermore, assume that there exists a constant $c \leq \beta$ such that for all i, for all $j \neq \sigma_i$, $R_{ij} \leq c \cdot \bar{R}_i$. Let θ be the assignment returned by* GURU*, and let ℓ be the customer segment such that for every product j, $\pi_j = \bar{R}_\ell$. Then, the function $\Pi(\theta)$ increases the revenue by at least*

$$(1-c)\bar{R}_1 \frac{1-\beta^{\ell-1}}{1-\beta}.$$

Proof. Consider B, the set of customer segments assigned to a product in θ. We first note that for all i in B, $\bar{R}_i \geq \bar{R}_\ell$, and in particular $B = \{1, \ldots, k\}$. Next, note that in θ, every segment i purchases Product σ_i since all products have the same price and each segment seeks to maximize its utility. Since σ_i is unique for each i and $S_i \neq S_{i'}$ for all i, no two segments purchase the same product. Thus, we can reindex the products such that $\sigma_i = i$.

Second, notice that since $\bar{R}_i \geq \bar{R}_{i-1} \cdot \beta$ for all i, we note that for all i, we also have $\bar{R}_i \geq \bar{R}_1 \cdot \beta^{i-1}$. Third, notice that in the underlying shortest paths digraph, for every node $1 \leq i \leq \ell - 1$, every edge entering node i has cost at least $\bar{R}_i - R_{ij} \geq \bar{R}_i - c \cdot \bar{R}_i = (1-c)\bar{R}_i$. Thus, the shortest path to node i is $\min\{\bar{R}_i, \bar{R}_\ell + (1-c)\bar{R}_i\}$, since the single-price algorithm sets $\pi_i = \bar{R}_\ell$ for all i. However, since $c \geq \beta$, we have that for all $i < \ell, \bar{R}_i \cdot c \geq \bar{R}_\ell$ and thus the shortest path to every node is at most $\bar{R}_i$. So we use the latter min term, and see that the function $\Pi(\theta)$ increases the cost of node i by at least $(1-c)\bar{R}_i$. Thus, the total amount of price increase over all the segments is:

$$\begin{aligned}
\sum_{i=1}^{\ell-1}(1-c)\bar{R}_i &\geq \sum_{i=1}^{\ell-1}(1-c)\bar{R}_1\beta^{i-1} \\
&= (1-c)\bar{R}_1 \sum_{i=1}^{\ell-1}\beta^{i-1} \\
&= (1-c)\bar{R}_1 \frac{1-\beta^{\ell-1}}{1-\beta}.
\end{aligned}$$

□

Remark 4.3.11. *When $\ell = n$, the bound in the above proof improves to $(1-c)\bar{R}_1 \frac{1-\alpha}{1-\beta}$.*

Proof. The following simple algebraic steps are sufficient:

$$\begin{aligned}
(1-c)\bar{R}_1 \frac{1-\beta^{n-1}}{1-\beta} &= (1-c)\bar{R}_1 \frac{1-(\alpha^{\frac{1}{n-1}})^{n-1}}{1-\beta} \\
&= (1-c)\bar{R}_1 \frac{1-\alpha}{1-\beta}.
\end{aligned}$$

□

While requiring a large amount of assumptions to become a provable statement, the most important one that contributes to the generation of increased revenue is the gap between $\bar{R}_i$ and Segment i's other valuations, for all i. In practice what this suggests is that GURU "underperforms" when customer segments have products that they highly prefer over all others, since it is unable to capture the potential gains when solving the underlying price-setting problem.

4.4 Low-Rank Matrices

In this section, we present a proof that the Maximum Utility Problem can be solved exactly when the matrix of reservation prices is rank 1. More broadly, the setting where data matrices are of low rank is one that has been considered in a wide variety of applications and optimization problems. For example, the methodology of *Factor Analysis* is a commonly used technique in statistics that is used to gain a representation of the correlation between a set of variables in terms of a small number of unseen factors. While it had its origins in psychometrics, it has since been applied to numerous fields, including econometrics, machine learning, and biology [2]. One connection it shares with optimization is with the *Rank-Constrained Factor Analysis problem*, in which a covariance matrix Σ of a set of random vectors $\mathbf{x}$ is decomposed into the sum of a nonnegative diagonal matrix and a positive semidefinite matrix of low rank [7].

Another area where low-rank matrices are concerned is the *Nonnegative Matrix Factorization* problem, in which one nonnegative matrix is factored into two new matrices, also nonnegative, and typically of low rank. The method is useful for identifying the addition of noise in datasets [49] and has been applied to many different scientific settings including mass spectronomy [18], audio processing [22], and imaging [35]. In relation to these problems, there is also *Nonnegative Low-Rank Matrix* approximation, which seeks to approximate large-scale datasets using products of low-rank matrices [53].

Relating the discussion of low-rank matrices to optimal pricing and economic equilibria, consider that often times large-scale instances of pricing problems can be driven by a very few number of factors (see, for instance, Subsection 4.3.1 of [38], or the introduction to this section). Due to the difficulty of estimating consumer preferences, one possibility for characterizing a large-scale dataset is through the multiplication of low-rank matrices that are defined by these "driving factors", as well as the addition of statistical noise [48]. This discussion motivates the study of the Maximum Utility Problem within the context of low-rank matrices, and the contribution of this thesis towards that conversation is a brief note regarding solutions to matrices of rank 1. It turns out that the algorithm presented in [26] can solve these exactly. The theorem statement and proof follows:

Theorem 4.4.1. *Instances of* MUP *with a rank-1 matrix of reservation prices can be solved exactly using Algorithm 4*

To prove this fact, it suffices to show that rank 1 matrices satisfy the Extended Monge Property. Recall that this condition is satisfied when all three of the following conditions hold:

a. For all $1 \leq i < \ell \leq n$ and $1 \leq j < k \leq m$,

$$R_{ij} + R_{\ell k} \geq R_{ik} + R_{\ell j}.$$

b. For all $1 \leq i \leq n$ and $1 \leq j < m$,

$$R_{ij} \geq R_{i,j+1}.$$

c. For all $1 \leq i < n$ and $1 \leq j \leq m$,

$$R_{ij} \geq R_{i+1,j}.$$

For any rank 1 matrix R, we can rearrange it in the necessary way and apply the dynamic programming-based Algorithm 4, which solves these instances exactly due to Theorem 3.1.4. Thus, we may validate Theorem 4.4.1 by proving the following lemma:

Lemma 4.4.2. *If a nonnegative matrix is rank 1, its rows and columns can be rearranged such that the new matrix satisfies the Extended Monge Property.*

Proof. Let A be a nonnegative matrix of rank 1, and let x and y be nonnegative vectors such that $xy^T = A$. Consider new vectors x' and y' that contain the elements of x and y in nonincreasing order - that is, $\{x'_1, \ldots, x'_n\} = \{x_1, \ldots, x_n\}$ and $x'_1 \geq x'_2 \geq \cdots \geq x'_n$, and similar for y'. Then the new matrix $A' = x'y'^T$ is simply the matrix A with several rows and columns rearranged. Additionally, the ij^{th} entry in A' is $A'_{ij} = x'_i \cdot y'_j$. Thus,

$$\begin{aligned} A'_{ij} - A'_{i,j+1} = x'_i y'_j - x'_i y'_{j+1} = x'_i(y'_j - y'_{j+1}) &\geq x'_{i+1}(y'_j - y'_{j+1}) \\ &= A'_{i+1,j} - A'_{i+1,j+1}, \qquad \text{(Assumption } \mathbf{a}\text{)} \end{aligned}$$

and

$$A'_{ij} = x'_i y'_j \geq x'_{i+1} y'_j = A'_{i+1,j}, \qquad \text{(Assumption } \mathbf{b}\text{)}$$

and

$$A'_{ij} = x'_i y'_j \geq x'_i y'_{j+1} = A'_{i,j+1}, \qquad \text{(Assumption } \mathbf{c}\text{)}$$

as desired. □

4.5 MaxR$^+$

We conclude the chapter with a new heuristic that takes inspiration from both Guru and MaxR. Notice that if we take the initial single-price vector π, then among all customer segments i for which $i \in C_j$ for some j (i.e. segments that purchase some product), Segment i will purchase some product in S_i, as all products are the same price and products in S_i will maximize the surplus of Segment i. This defines an assignment θ, which we can improve through continued iterations of Π and $\mathcal{C}$ until a fixed point is found. Performing these operations on the best single-price vector is precisely the algorithm $\overline{\text{Guru}}$ from Myklebust, et al. [48]. Our algorithm blends $\overline{\text{Guru}}$ and MaxR by considering a larger set of potential assignments than MaxR does, and performing one iteration of $\Pi(\theta)$ for each assignment.

The main idea of the new algorithm is to test MaxR on each subset of customer segments $\{1\}, \{1,2\}, \{1,2,3\}, \ldots, \{1,2,\ldots,n\} \subseteq I$, whereas the original MaxR heuristic only considers the full set of customer segments. As such, this new algorithm is called MaxR$^+$. The advantage of this approach is that we are able to rule out scenarios where low-valued segments drive down the overall price of products. For example, in an instance with two customer segments of size $N_i = 1$ and a

single product with $R_{11} = 100$ and $R_{21} = 1$, the original MaxR heuristic will produce a revenue of 2, while our new enhanced heuristic will be able to consider the assignment $\theta_{11} = 1, \theta_{21} = 0$ which has an optimal revenue of 100. However, this simple approach can be improved in the case when S_i contains multiple elements for some i.

Example 4.5.1. *Consider the following simple instance:*

i	N_i	R_{i1}	R_{i2}
1	1	100	1
2	100	1	1

Table 4.2: An example of an edge case where MaxR is not optimal

In this example, there are two elements of S_2. In the MaxR *heuristic, if Segment 2 selects Product 1, then the maximum revenue we can achieve is 101. However, if Segment 2 selects Product 2, then the maximum revenue we can achieve is 200.*

To optimize around this edge case, we need to introduce a few extra pieces of notation. During the algorithm, we will explicitly create a unique assignment θ^{ij} for every segment/product pair i, j such that $j \in S_i$. Now, let $p(\theta^{ij})$ denote the revenue of an assignment θ^{ij} with prices $\Pi(\theta^{ij})$. At the end of iteration i, we let τ_i denote the product j' such that $j' \in \arg\max_{j \in S_i}\{p(\theta^{ij})\}$. In the case of a tie for j', we resort to our conventional tiebreakers. Then, we fix τ_i as the product that Segment i will always purchase in initial assignments during subsequent iterations. Putting all of this together, we are ready to present the algorithm MaxR$^+$:

Algorithm 6 MaxR$^+$ (Enhanced Single-Price Heuristic)

Define $\bar{R}_i := \max_j\{R_{ij}\}$ for all $i \in \{1, \ldots, n\}$.
Sort the customer segments such that $\bar{R}_1 \geq \bar{R}_2 \geq \cdots \geq \bar{R}_n$.
for all $1 \leq i \leq n$ **do**
  for all $j \in S_i$ **do**
    Initialize $\theta^{ij}_{\ell k} = 0$ for all ℓ, k.
    for all $1 \leq \ell < i$ **do**
      Set $\theta^{ij}_{\ell \tau_\ell} = 1$
    end for
    Set $\theta^{ij}_{ij} = 1$
    for all $i + 1 \leq k \leq n$ **do**
      if $\bar{R}_k = \bar{R}_i$ **then**
        Set $\theta^{ij}_{kk'} = 1$ for the element $k' \in S_k$ with the smallest index.
      end if
    end for
    Record the resulting assignment θ^{ij}, price vector $\Pi(\theta^{ij})$, and objective value $p(\theta^{ij})$.
  end for
  Set $\tau_i = \arg\max_{j \in S_i}\{p(\theta^{ij})\}$.
end for
return The assignment $\theta^{i^* j^*}$ such that $\{i^*, j^*\} = \arg\max_{i, j \in S_i}\{p(\theta^{ij})\}$.

We note that steps 10-14 are for another specific edge case: if there are two segments i and i' such that $\bar{R}_i = \bar{R}_{i'}$, then including one of them while excluding the other in θ may lead to an infeasible assignment. Explicitly including i' in θ eliminates this possibility. Now, we can prove that the algorithm is well-defined. This amounts to showing that it only ever considers feasible assignments:

Proposition 4.5.2. *For all $i, j \in S_j$, the assignment θ^{ij} is feasible.*

Proof. It suffices to show that there exists a set of prices for the current assignment that are feasible. Consider using the pricing vector π such that $\pi_j = \bar{R}_i$ for all j. Then if we apply π to θ^{ij}, every segment i' such that $\bar{R}'_i \geq \bar{R}_i$ will be assigned to one of their most preferred products. They will also purchase a product that maximizes their utility, since all products have the same price. Moreover, all other segments purchase nothing since there is no product with a price at most their maximum reservation price. Thus, π is a feasible set of prices for θ^{ij} and hence θ^{ij} is a feasible assignment. □

Now that we have shown Algorithm 6 is well-defined, we may also note some special properties of MaxR$^+$. In particular, these are all properties inherited from either MaxR or Guru, and while MaxR$^+$ is indeed at least as profitable as Guru, we cannot necessarily say the same for MaxR. The reason is due to the fact that the algorithms have different tiebreaking mechanisms. However, all of the properties we know to be true for MaxR are also true for MaxR$^+$.

Theorem 4.5.3. MaxR$^+$ *has the following properties:*

- ***a.*** MaxR$^+$ *is strongly polytime.*
- ***b.*** *In the assignment produced by* MaxR$^+$*, every customer segment is assigned to a member of S_i.*
- ***c.*** *If for all i, $|S_i| = 1$ and for every other segment i', $R_{i\sigma_i} > R_{i'\sigma_i}$, then* MaxR$^+$ *is optimal.*
- ***d.*** MaxR$^+$ *generates at least as much revenue as* Guru*.*
- ***e.*** *If $\frac{\max_i N_i}{\min_j N_j} \leq c$ for some constant $c = O(1)$, then* MaxR$^+$ *is a $O(\log(n))$-approximation algorithm.*
- ***f.*** *If $n \geq 2$ and $k \in (0, n-1]$, and if for all $1 \leq i \leq n-1$, $\frac{\bar{R}_{i-1}}{\bar{R}_i} \geq 1 - \frac{k}{n-1}$, then the approximation ratio of* MaxR$^+$ *is at most $(k+1)$.*
- ***g.*** *If $I = \{1, \ldots, n\}$ is α-dense and β-uniformly dense for $\beta = \alpha^{\frac{1}{n-1}}$,* MaxR$^+$ *has a worst-case approximation ratio at most $\ln(\frac{1}{\alpha}) + 1$.*

Proof. To prove Property **a**, notice that the bottleneck in Algorithm 6 is computing the price vector $\Pi(\theta^{ij})$ for each i, j. As this can be done efficiently with the Bellman-Ford-Moore algorithm and we only need to do so polynomially many times, MaxR$^+$ is strongly polytime.

Property **b** is immediate, as in steps 7, 9, and 12 of Algorithm 6, we only set $\theta^{ij}_{\ell k} = 1$ for some ℓ, k if $k \in S_\ell$.

For property **c**, see that the assignment $\theta^{n\sigma_n}$ is precisely the assignment that assigns every segment i to the unique element of S_i. Moreover, since for every pair of segments i, i', we have $R_{i\sigma_i} > R_{i'\sigma_i}$,

the optimal set of prices for $\theta^{n\sigma_n}$ is to set $\pi_{\sigma_i} = \bar{R}_i$ for all i. This leads to a revenue of $\sum_{i=1}^{n} N_i \bar{R}_i$ which by Proposition 2.0.4 is the upper bound for MUP. Since MaxR$^+$ considers $\theta^{n\sigma_n}$ as a potential assignment, it must achieve this revenue.

To prove Properties **e**, **f**, and **g**, it suffices to Prove Property **d**. Property **e** is shown to be true for Guru in [27], while properties **f** and **g** correspond to Theorems 2.3.2 and 4.3.8, respectively. If we show that MaxR$^+$ always produces a revenue at least as large as the revenue generated by Guru, then the resulting worst-case approximation bounds follow.

Recall that during step 3 of Guru, we find an index ℓ such that $\ell = \arg\max_k \left\{ \bar{R}_k \sum_{i=1}^{k} N_i \right\}$. During the ℓ^{th} step of the arg max calculation, Guru effectively considers the pricing vector π^ℓ such that $\pi^\ell_j = \bar{R}_\ell$ for all $j \in J$. In its corresponding assignment $\mathcal{C}(\pi^\ell)$, then, all customer segments $1 \leq i \leq \ell$ are assigned to some product in S_i. Notice that this is a property shared by *any* assignment $\theta^{\ell j}$ such that $j \in S_\ell$. Under the pricing vector π^ℓ, then, $\theta^{\ell j}$ generates at least as much revenue as $\mathcal{C}(\pi^\ell)$, since all segments $i \leq \ell$, as well as all segments ℓ' such that $\bar{R}_\ell = \bar{R}_{\ell'}$, purchase a product and pay $\bar{R}_\ell$ for it. Thus, MaxR$^+$ either returns the assignment $\theta^{\ell j}$, which generates at least as much revenue as the assignment $\mathcal{C}(\pi^\ell)$ returned by Guru, or returns an even more profitable assignment. Hence, the revenue generated by MaxR$^+$ is at least the revenue generated by Guru. Properties **e**, **f**, and **g** follow. □

The upside of this starting heuristic is that it considers a large number of assignments that include the ones given by Guru, and has the same general properties as both Guru and MaxR. Thus, we may expect it to perform better than these heuristics. The downside is a longer runtime, as well as the fact that we might end up with a smaller set of customer segments purchasing products, or a smaller number of products being offered. With DK88, this means a much smaller set of assignments can be possibly considered since the heuristic has no method of reviving eliminated segments or products. Overall, as with MaxR and Guru, the utility of Algorithm 6 is not necessarily a product of its provable approximation ratios but as a good initial assignment for the Dobson-Kalish algorithm or other reassignment heuristics.

Chapter 5

Upper Bound Improvements

In Chapter 4, we were able identify specific assumptions to the underlying data for which new approximation ratios for existing and original algorithms could be found. However, in most of these instances, the value that we use to upper bound the optimal profit is the same as from Proposition 2.0.4: $\sum_i N_i \bar{R}_i$. While there exist many instances where this upper bound is achieved, it is also highly inefficient in others. For example, in any instance where the most preferred product for every segment is unique and the naive upper bound is tight, it is always possible to achieve the optimal revenue through use of MaxR:

Proposition 5.0.1. *Assume that for all i, $|S_i| = 1$ and that the optimal value of (1.4) is $\sum_{i=1}^{n} N_i \bar{R}_i$. Then* MaxR *achieves the optimal revenue.*

Proof. For every Segment i, there is only one product in J for which they are willing to pay $\bar{R}_i$ - precisely the product in S_i. Thus, the only possible assignment that can achieve a revenue of $\sum_{i=1}^{n} N_i \bar{R}_i$ is the assignment that assigns every customer segment to their most preferred product. By Proposition 2.0.5, MaxR produces this assignment. □

Conversely, we can consider a case where the naive upper bound is guaranteed to perform poorly - the $m = 1$ case. Since there is only one product, if $\bar{R}_i$ values are distinct, then it is only possible for one customer segment to pay their $\bar{R}_i$ value - and some segments may not purchase the product at all in an optimal solution. Thus, the naive upper bound severely overestimates the total revenue that can be produced. However, in this case, we *also* have an algorithm that is optimal - Guru.

Proposition 5.0.2. *When $m = 1$,* Guru *provides an optimal solution to (1.4).*

Proof. From Section 3.1, we know that when $m = 1$, then the linear program (3.1) computes an optimal solution to (1.4). Moreover, there exists an integer-valued solution z_{i^*} when

$$i^* \in \arg\max_k \left\{ \bar{R}_k \sum_{i=1}^{k} N_i \right\}.$$

Notice that i^* is the same index returned by Guru that we use to set the price for Product 1. Thus, Guru is optimal. □

In a sense, then, we already have algorithms for some extreme cases - both where the upper bound performs poorly and where it represents the optimal revenue. However, there is a vast middle ground of instances where neither our algorithms nor our naive upper bound have been shown to be efficient or accurate. Hence, the goal of this chapter is to further our understanding of the upper bound and attempt to improve it in some specific instances. First, in Section 5.1 we provide an introduction to upper bound analysis by examining a small subcase. Then, in Section 5.2 we introduce the LP relaxation of (4.1), its dual, and use both of them to investigate possibilities for upper bounds in a slightly more general case.

5.1 Introduction to Upper Bound Analysis

Overall, there is some evidence to show that when the number of customer segments is relatively small, especially relative to the number of products, the Maximum Utility Problem may be easier to solve. There exist a few special cases where we already have exact solutions - the $n = O(1)$ case, and the case outlined in Lemma 3.1.1. Additionally, when $N_i = 1$ for all i, we have seen that the approximation ratio of Guru is at most $O(n \log n)$. Finally, as we can offer at most n products in any given assignment, the initial feasible assignment used for any application of DK88 will lead to an underlying shortest paths graph with at most $n + 1$ nodes (due to the dummy node). Thus, if n is relatively small, then the shortest paths graph will also be relatively small and DK88 is likely to have a shorter runtime.

Since we already have many good options for scenarios where $n \leq m$ it will be useful to study improvements to the upper bound in the *converse* situation - when $n \geq m$. First, as a warmup, we explicitly derive an improved upper bound for a small subcase.

5.1.1 The 2 Segments, 2 Products Case

Consider a pricing vector π and corresponding assignment θ where the profit equals $\sum_{i=1}^{n} N_i \bar{R}_i$. Then for all customer segments i, if $\theta_{ij} = 1$, we must have

$$\bar{R}_i - \pi_{\sigma_i} \geq R_{ij} - \pi_j, \ \forall j \neq \sigma_i.$$

Moreover, if the upper bound is achieved, then $\pi_{\sigma_i} = \bar{R}_i$ for all i. Thus, for all i and $j \neq i$,

$$\bar{R}_i \geq R_{j\sigma_i}. \tag{5.1}$$

Now, let us study the case where for exactly one pair of segments, (5.1) is invalidated. In particular, assume that for all $3, \ldots, n$, for all $j \in \{1, \ldots, m\}$,

$$\bar{R}_i \geq R_{j\sigma_i}.$$

Let $\bar{R}_1 > \bar{R}_2$ without loss of generality. Let $\sigma_2 = 2$. Assume that:

$$\bar{R}_2 < R_{12}.$$

Thus, we assume that the inequality (5.1) is false for Segment 2 in particular. Now, we shall attempt to derive a new upper bound for the amount of revenue we can receive from segments 1 and 2. There are two specific cases we need to consider. In the first, $R_{12} = \bar{R}_1$ (so $\sigma_1 = 2$). In

Example 5.1.1.

i	R_{i1}	R_{i2}
1	5	6
2	2	4

i	R_{i1}	R_{i2}
1	8	20
2	9	15

Table 5.1: Examples where $R_{12} = \bar{R}_1$

this case, the values could look something like the instances in Example 5.1.1 (note that the N_i values are arbitrary). If $R_{11} \geq R_{21}$, then we should never offer Product 1, since both $R_{12} \geq R_{11}$ and $R_{22} \geq R_{21}$. So, the optimal value is

$$\max\{N_1\bar{R}_1, \bar{R}_2(N_1 + N_2)\}.$$

On the other hand, if as in the second example $R_{11} < R_{21}$, then if Segment 1 buys Product 2, we can offer Product 1 to Segment 2 without losing any profit. Thus, the optimal profit would be

$$\max\{N_1\bar{R}_1 + N_2R_{21}, \bar{R}_2(N_1 + N_2)\}.$$

In the second case, we have $R_{11} = \bar{R}_1$. For instance:

Example 5.1.2.

i	R_{i1}	R_{i2}
1	7	5
2	2	4

Table 5.2: An example when $R_{11} = \bar{R}_1$

Again, the N_i are arbitrary. In this scenario, Segment 2 will never purchase Product 1, as this would lead to a profit of $(N_1 + N_2)R_{21} \leq (N_1 + N_2)\bar{R}_2$. Next, notice that if Segment 1 buys Product 1 and Segment 2 buys Product 2, we will always set $\pi_2 = \bar{R}_2$ and thus the highest we can set π_1 is $\bar{R}_1 - (R_{12} - \bar{R}_2)$. Thus, the profit we gain in this scenario is at least $N_1(\bar{R}_1 - R_{12}) + \bar{R}_2(N_1 + N_2) > \bar{R}_2(N_1 + N_2)$, so in particular we notice that we will not have both segments purchasing Product 2. Of course, it is also possible that Segment 1 alone purchases Product 1, leading to a profit of $N_1\bar{R}_1$. So, the optimal profit is:

$$\max\{N_1\bar{R}_1, N_1(\bar{R}_1 - R_{12}) + \bar{R}_2(N_1 + N_2)\}.$$

Combining the cases together, we can construct an upper bound of:

$$\max\{N_1\bar{R}_1 + N_2R_{21}, N_1(\bar{R}_1 - R_{12}) + \bar{R}_2(N_1 + N_2)\}. \tag{5.2}$$

Lastly, we may note a useful structural fact:

Proposition 5.1.3. *In the $n = m = 2$ case, Segment 1 will always purchase a member of S_i in any optimal solution.*

Proof. Without loss of generality, we assume that $R_{11} > R_{12}$. This gives us two cases:

- $R_{21} = \bar{R}_2$.
  Then the optimal value is either $\max\{N_1\bar{R}_1, R_{21}(N_1+N_2)\}$ or $\max\{N_1\bar{R}_1 + N_2R_{22}, R_{21}(N_1+N_2)\}$, and in both scenarios Segment 1 purchases Product 1.
- $R_{22} = \bar{R}_2$.
  Then if $R_{12} < R_{22}$, both segments simply purchase their most desired product. Otherwise, the optimal profit is $\max\{N_1\bar{R}_1, N_1(\bar{R}_1 - R_{12}) + R_{22}(N_1+N_2)\}$. Again, all scenarios have Segment 1 purchasing Product 1.

□

5.2 Linear Programming-Derived Upper Bounds

Another obvious starting point for computing a better upper bound is primal/dual analysis. As optimal dual solutions upper bound optimal primal solutions, finding feasible dual solutions will allow us to improve our overall upper bound for MUP. We begin with the linear programming relaxation of (4.1):

$$\max \sum_{i=1}^{n}\sum_{j=1}^{m} N_i p_{ij}, \tag{5.3}$$

$$\begin{aligned}
\text{s.t. } \sum_{j\neq k}(R_{ij}\theta_{ij} - p_{ij}) &\geq R_{ik}\left(\sum_{j\neq k}\delta_{ijk}\theta_{ij}\right) - \pi_k, && \forall i,k,\\
R_{ij}\theta_{ij} - p_{ij} &\geq 0, && \forall i,j,\\
\sum_{j=1}^{m}\theta_{ij} &\leq 1, && \forall i,\\
p_{ij} &\leq \pi_j, && \forall i,j,\\
p_{ij} &\geq \pi_j - \tilde{R}_j(1-\theta_{ij}), && \forall i,j,\\
\theta_{ij}, \pi_j, p_{ij} &\geq 0, && \forall i,j.
\end{aligned}$$

The next step is to write down the dual. As an intermediate step, we rewrite (5.3) to a more palatable form (for our purposes):

$$\max \sum_{i=1}^{n}\sum_{j=1}^{m} N_i p_{ij},$$

$$\begin{aligned}
\text{s.t. } \sum_{j\neq k}(\theta_{ij}(R_{ik}\delta_{ijk} - R_{ij}) + p_{ij}) - \pi_k &\leq 0, && \forall i,k, && (\omega)\\
\tilde{R}_j\theta_{ij} - p_{ij} + \pi_j &\leq \tilde{R}_j, && \forall i,j, && (\gamma)\\
\sum_{j=1}^{m}\theta_{ij} &\leq 1, && \forall i, && (y)\\
p_{ij} - R_{ij}\theta_{ij} &\leq 0, && \forall i,j, && (\sigma)\\
p_{ij} - \pi_j &\leq 0, && \forall i,j, && (\beta)\\
\theta_{ij}, p_{ij}, \pi_j &\geq 0, && \forall i,j.
\end{aligned}$$

Finally, the dual:

$$\min \sum_{i=1}^{n} y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} \tilde{R}_j \gamma_{ij}, \tag{5.4}$$

$$\text{s.t.} \sum_{k\neq j} \omega_{ik}(R_{ik}\delta_{ijk} - R_{ij}) - R_{ij}\sigma_{ij} + y_i + \tilde{R}_j\gamma_{ij} \geq 0, \qquad \forall i,j, \qquad (\theta)$$

$$\sum_{k\neq j} \omega_{ik} + \sigma_{ij} + \beta_{ij} - \gamma_{ij} \geq N_i, \qquad \forall i,j, \qquad (p_{ij})$$

$$\sum_{i=1}^{n} (\gamma_{ij} - \beta_{ij} - \omega_{ij}) \geq 0, \qquad \forall j, \qquad (\pi_j)$$

$$\omega, \gamma, y, \sigma, \beta \geq 0. \tag{5.5}$$

We wish to compute feasible solutions to (5.4). To kick off our analysis, we test on a small example.

Example 5.2.1. *Let $n = m = 1$, with $R_{11} = 10$ and $N_1 = 5$. Then the dual simplifies to:*

$$\begin{aligned} \min\, & y_1 + 10\gamma_{11}, \\ s.t.\ & -10\sigma_{11} + y_1 + 10\gamma_{11} \geq 0, \\ & \sigma_{11} + \beta_{11} - \gamma_{11} \geq 5, \\ & \gamma_{11} - \beta_{11} - \omega_{11} \geq 0, \\ & \omega, \gamma, y, \sigma, \beta \geq 0. \end{aligned}$$

An optimal solution is $y_1 = 50$, $\sigma_{11} = 5$, with objective value 50.

Notice that the objective value in Example 5.2.1 is precisely the same as the naive upper bound that we used in our worst-case analysis of Guru. In fact, it turns out that for any instance of (5.4), there exists a simple solution that achieves this same upper bound.

Proposition 5.2.2. *A feasible solution to* (5.4) *is given by $y_i = \bar{R}_i N_i$, $\sigma_{ij} = N_i$. The resulting objective value is equal to the trivial revenue upper bound from Proposition 2.0.4.*

The π_j constraints are satisfied automatically, and the p_{ij} constraints are satisfied as $\sigma_{ij} = N_i$ for all i, j. The θ constraints are satisfied since $y_i = \bar{R}_i N_i \geq R_{ij} N_i = R_{ij}\sigma_{ij}$. Finally, the objective value is equal to $\sum_i y_i = \sum_i \bar{R}_i N_i$.

However, for even slightly more complicated examples, the dual LPs can have nontrivial optimal solutions. Take the following example, with $\delta_{ijk} = 0$ for all i, j, k:

Example 5.2.3.

i	N_i	R_{i1}	R_{i2}
1	2	4	2
2	1	2	1

Table 5.3: A simple instance with a nontrivial optimal dual solution

(5.4) has the following optimal solution for the instance given by Table 5.3 (nonzero values only are given):

$y_1 = 4, y_2 = 0.5, \gamma_{21} = 0.75, \gamma_{22} = 0.5, \omega_{12} = 0.5, \sigma_{11} = 0.75, \sigma_{12} = 2, \sigma_{21} = 1.75, \sigma_{22} = 1.5, \beta_{11} = 0.75,$

with objective value = 8.5.

There are also examples where the dual variables achieve larger values, like in the following scenario:

Example 5.2.4.

i	N_i	R_{i1}	R_{i2}
1	2	4	2
2	1	2	1
3	2	3	6

Table 5.4: An example with relatively large optimal dual variables

The corresponding objective value is 20, given by $\pi_1 = 4, \pi_2 = 6$. This is achieved with dual variables:

$$y_1 = 4, y_3 = 6, \gamma_{21} = 1, \gamma_{22} = 1, \omega_{11} = 1, \omega_{12} = 2, \omega_{21} = 2, \omega_{22} = 6, \omega_{31} = 2, \omega_{32} = 1, \beta_{11} = 1, \beta_{32} = 1.$$

We would like to connect this with our discussion from Section 5.1.1. Let's return to Example 5.1.1, with additional N_i values:

i	N_i	R_{i1}	R_{i2}
1	5	8	20
2	10	9	15

Table 5.5: The second instance in Example 5.1.1, with N_i values

The crude estimate $\max\{N_1\bar{R}_1 + N_2R_{21}, N_1(\bar{R}_1 - R_{12}) + \bar{R}_2(N_1 + N_2)\}$ yields an upper bound of $\max\{190, 225\} = 225$, which can be achieved by setting $\pi_1 = \infty, \pi_2 = 15$.

The optimal dual solution, computed using Gurobi, is 233.78. Note this is only slightly lower than the naive upper bound, which has an objective value of 250.

Unfortunately, there also exists large families of R_{ij} data where the crude upper bound $\sum_{i=1}^{n} N_i\bar{R}_i$ is the optimal value of the primal, but where GURU performs optimally with a smaller revenue.

Theorem 5.2.5. *Let $m \geq 2$, and let the matrix of reservation prices be β-uniformly dense for $\beta = 0.5$ and assume that for all i, j, $R_{ij} = \bar{R}_i$. Then the objective value $\sum_{i=1}^{n} N_i\bar{R}_i$ is optimal for the dual.*

Proof. It suffices to find a solution to the primal with objective value $\sum_{i=1}^{n} N_i\bar{R}_i$. First, for all $j \geq 3$, set $\pi_j = 0$ and $p_{ij}, \theta_{ij} = 0$ for all i. Note that any constraint involving a product $j \geq 3$ is automatically satisfied, so for the remainder of the proof we only consider segments where $j \leq 2$.

Our solution is as follows: set $\theta_{11} = \theta_{12} = 0.5$, and set $p_{11} = p_{12} = \pi_1 = \pi_2 = \frac{\bar{R}_1}{2}$. For the other

variables, for all i, the values are:

$$\begin{aligned}\theta_{i1} :=& \frac{2^{i-2}}{2^{i-1}-1}\\ \theta_{i2} :=& 1-\theta_{i1}\\ p_{i1} :=& \frac{\bar{R}_1}{2^i-2}\\ p_{i2} :=& \bar{R}_i - p_{i1}\end{aligned}$$

We can easily see that this satisfies all primal constraints. Moreover, by definition $p_{i1}+p_{i2}=\bar{R}_i$. Thus, the objective function has value $\sum_{i=1}^{n} N_i\bar{R}_i$, as desired. □

Lemma 5.2.6. GURU *performs optimally on the instances described in Theorem 5.2.5*

Proof. Let π be an arbitrary pricing vector. All customer segments are indifferent between each product, so if for any j,k, $\pi_j > \pi_k$, all segments will prefer Product k over Product j. In particular, this implies that either $\pi_j = \pi_k$ for all j,k, or all customer segments will purchase the same product. Thus, the assignment $\theta(\pi)$ is equivalent to the assignment $\theta(\pi^*)$, where π^* is the pricing vector where every product is given the price $\min_j\{\pi_j\}$. However, by definition, GURU finds the optimal pricing vector given that all products have the same price. So in this case, GURU is optimal. □

We can demonstrate the result of Theorem 5.2.5 through a simple example.

Example 5.2.7.

i	N_i	R_{i1}	R_{i2}
1	1	100	100
2	1	50	50

Table 5.6: An example where the solution from Theorem 5.2.5 is outperformed by GURU

Here, the objective value of the primal using the solution from Theorem 5.2.5 is 150, which is equal to the upper bound from Proposition 2.0.4. However, the optimal value is 100, achieved by setting $\pi_1 = 100, \pi_2 = 100$. GURU achieves this since it considers setting $\pi_1, \pi_2 = \bar{R}_1 = 100$ as a potential pricing vector.

Through observation and testing examples, it appears that Theorem 5.2.5 is true regardless of the value of β. However, a general formula for primal variables that achieves the desired objective value has not yet been found. Overall, the "moral" from this approach is that the primal is less effective in establishing upper bounds when customer segments are "indifferent" across the line of prices (in that the difference between their highest and lowest reservation prices is small). In the following subsection, we will see another example where this phenomenon occurs.

5.2.1 Linear Programming Upper Bounds for the 2 Product Case

Now that we have developed some practice analyzing the upper bound with the primal and dual, we can return to a generalized version of the task from Section 5.1.1. Unfortunately, even the 2×3 case is not as simple. There exist instances where Segment 1 does *not* always purchase their

i	N_i	R_{i1}	R_{i2}
1	1	10	8
2	5	8	3
3	100	1	2

Table 5.7: An instance where the highest valued segment does not purchase a member of S_i

most desired product. For example: in Table 5.7, the optimal solution has $\pi_1 = 7, \pi_2 = 2$ with $\theta_{12} = 1, \theta_{21} = 1$, and $\theta_{32} = 1$.

Next, we study a lengthier example that helps us glean more insights into the underlying structure of the dual. While this example ultimately does not provide inspiration for a general improved upper bound, it does provide a general framework for how improved upper bounds could be found, especially when more structure to the data is assumed.

Example 5.2.8. *We consider the example from [51] which disproved the original runtime claims from [17]. Note we have fixed $\epsilon = 1$. The trivial upper bound has value 4620, the LP relaxation gives*

i	N_i	R_{i1}	R_{i2}
1	1	0	1
2	1	1	2
3	1	4	5
4	1	9	10
5	1	8	9
6	1	17	18
7	90	12	13
8	1	5	2
9	1	10	7
10	1	9	6
11	1	18	17
12	1	13	10
13	90	26	23
14	10	101	100

Table 5.8: A large instance

value 4480, and the single-price algorithm gives value 2600. The best integer solution the author can find is $\pi_1 = 16, \pi_2 = 13$ which gives a revenue of 2769. The dual variables in the optimal solution are presented in Table 5.9, from which we glean many insights. First, we notice that the dual variables for the first 13 customer segments follow a strict formula. Let $\hat{R}_i := \min_j\{R_{ij}\}$. Then the dual variables for customer segments $1 \leq i \leq 13$ satisfy the following conditions:

$$y_i = \hat{R}_i N_i, \quad \gamma_{ij} = \begin{cases} 0, & \text{if } R_{ij} = \hat{R}_i, \\ \frac{N_i(R_{ij} - \hat{R}_i)}{\bar{R}_j - R_{ij}}, & \text{if } R_{ij} = \bar{R}_i, \end{cases} \quad \sigma_{ij} = \begin{cases} N_i, & \text{if } R_{ij} = \hat{R}_i, \\ \gamma_{ij} + N_i, & \text{if } R_{ij} = \bar{R}_i, \end{cases} \quad \beta_{ij} = \omega_{ij} = 0. \tag{5.6}$$

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}
0	1	4	9	8	17	1080	2	7	6	17	10	2070	761

	Product 1	Product 2
γ_{1j}	0	0.010101010101...
γ_{2j}	0	0.010204081632...
γ_{3j}	0	0.010526315789...
γ_{4j}	0	0.0111111111111...
γ_{5j}	0	0.010989010989...
γ_{6j}	0	0.012195121951...
γ_{7j}	0	1.0344827586...
γ_{8j}	0.03125	0
γ_{9j}	0.03296703296...	0
γ_{10j}	0.0326086956...	0
γ_{11j}	0.0120481927...	0
γ_{12j}	0.0340909090...	0
γ_{13j}	3.6	0
γ_{14j}	0	0
ω_{14j}	1.2776163..	0

	Product 1	Product 2
σ_{1j}	1	1.0101010101...
σ_{2j}	1	1.0102040816...
σ_{3j}	1	1.0105262631...
σ_{4j}	1	1.0111111111...
σ_{5j}	1	1.0109890109...
σ_{6j}	1	1.0121951219...
σ_{7j}	90	91.03448275...
σ_{8j}	1.03125	1
σ_{9j}	1.0329670...	1
σ_{10j}	1.0326086...	1
σ_{11j}	1.0120481...	1
σ_{12j}	1.0340909...	1
σ_{13j}	93.6	90
σ_{14j}	7.5346...	7.62277...
β_{14j}	2.4653...	1.099609...

Table 5.9: The optimal dual variables for the instance in Table 5.8

We can see that for the first 13 customer segments, all dual constraints hold with equality. Note that the contribution each segment pays to the overall minimization problem is actually worse *than the contribution they pay in the naive upper bound. For example, for Segment 13, we have* $N_{13}\bar{R}_{13} = 90 \cdot 13 = 1170$. *However, with this set of dual variables, their contribution to the objective function is* $\hat{R}_{13}N_{13} + \frac{90(13-12)}{100-13} \cdot \tilde{R}_2 = 1183$. *This means that all of the contribution to a better solution is coming from Segment 14. Thus, we wish to obtain an algebraic representation of Segment 14's dual variables. If we fix the dual variables involving all other segments and look at only the constraints involving Segment 14, we are left with a small LP:*

$$\begin{aligned}
\min y_{14} + \tilde{R}_1\gamma_{14,1} + \tilde{R}_2\gamma_{14,2}, \\
y_{14} - R_{14,1}\sigma_{14,1} \geq 0, \\
y_{14} + \omega_{14,1}(R_{14,1} - R_{14,2}) - R_{14,2}\sigma_{14,2} \geq 0, \\
\sigma_{14,1} + \beta_{14,1} \geq N_{14}, \\
\sigma_{14,2} + \beta_{14,2} + \omega_{14,1} \geq N_{14}, \\
\sum_{i=1}^{13} \gamma_{i1} - \omega_{14,1} - \beta_{14,1} \geq 0, \\
\sum_{i=1}^{13} \gamma_{i2} - \beta_{14,2} \geq 0,
\end{aligned}$$

which achieves optimality with the values in the table above. In this particular instance, there exists

a feasible solution with all the constraints at equality. As such, we can find algebraic representations of the values of the variables using Gaussian Elimination. The values are:

$$\begin{aligned}
y_{14} &= \frac{N_{14}(R_{14,1}+R_{14,2}) - R_{14,1}\sum_{i=1}^{13}\gamma_{i1} - R_{14,2}\sum_{i=1}^{13}\gamma_{i2}}{2},\\
\sigma_{14,1} &= \frac{N_{14}(R_{14,1}+R_{14,2}) - R_{14,1}\sum_{i=1}^{13}\gamma_{i1} - R_{14,2}\sum_{i=1}^{13}\gamma_{i2}}{2R_{14,1}},\\
\sigma_{14,2} &= \frac{N_{14}(3R_{14,1}-R_{14,2}) - R_{14,1}\sum_{i=1}^{13}\gamma_{i1} - 2R_{14,1}\sum_{i=1}^{13}\gamma_{i2} + R_{14,2}\sum_{i=1}^{13}\gamma_{i2}}{2R_{14,1}},\\
\beta_{14,1} &= \frac{N_{14}(R_{14,1}-R_{14,2}) + R_{14,1}\sum_{i=1}^{13}\gamma_{i1} + R_{14,2}\sum_{i=1}^{13}\gamma_{i2}}{2R_{14,1}},\\
\beta_{14,2} &= \sum_{i=1}^{13}\gamma_{i2},\\
\omega_{14,1} &= \frac{N_{14}(R_{14,2}-R_{14,1}) + R_{14,1}\sum_{i=1}^{13}\gamma_{i1} - R_{14,2}\sum_{i=1}^{13}\gamma_{i2}}{2R_{14,1}}.
\end{aligned}$$

Thanks to a definitive representation of the dual variables, we can also obtain an algebraic representation of the objective value. The contribution of the γ variables to the objective function is:

$$\begin{aligned}
R_{14,1}\sum_{i=1}^{13}\gamma_{i1} + R_{14,2}\sum_{i=1}^{13}\gamma_{i2} &= \sum_{i=1}^{13}\left(\tilde{R}_1\frac{N_i(R_{i1}-\hat{R}_i)}{\tilde{R}_1 - R_{i1}} + \tilde{R}_2\frac{N_i(R_{i2}-\hat{R}_i)}{\tilde{R}_2 - R_{i2}}\right)\\
&= \sum_{i=1}^{13}\left(\tilde{R}_{\sigma_i}\frac{N_i(\bar{R}_i-\hat{R}_i)}{\tilde{R}_{\sigma_i}-\bar{R}_i}\right).
\end{aligned}$$

Plugging everything in to the objective function, we achieve a value of

$$\begin{aligned}
&\sum_{i=1}^{13}N_i\hat{R}_i + \frac{N_{14}(\hat{R}_{14}+\bar{R}_{14})}{2} + \sum_{i=1}^{13}\sum_{j=1}^{2}\gamma_{ij}\tilde{R}_j - \frac{1}{2}\sum_{i=1}^{13}\left(\tilde{R}_{\sigma_i}\frac{N_i(\bar{R}_i-\hat{R}_i)}{\tilde{R}_{\sigma_i}-\bar{R}_i}\right)\\
&= \sum_{i=1}^{13}N_i\hat{R}_i + \frac{N_{14}(\hat{R}_{14}+\bar{R}_{14})}{2} + \frac{1}{2}\sum_{i=1}^{13}\left(\tilde{R}_{\sigma_i}\frac{N_i(\bar{R}_i-\hat{R}_i)}{\tilde{R}_{\sigma_i}-\bar{R}_i}\right).
\end{aligned}$$

Generalizing the objective value from Example 5.2.8, if we assume that $m = 2$, $R_{11} > R_{i1}$, and $R_{12} > R_{i2}$ for all i, and $R_{11} \geq R_{12}$, then we can represent the above objective value as:

$$\frac{N_1(\tilde{R}_1+\tilde{R}_2)}{2} + \sum_{i=2}^{n}\left(N_i\hat{R}_i + \frac{\tilde{R}_{\sigma_i}N_i(\bar{R}_i-\hat{R}_i)}{2(\tilde{R}_{\sigma_i}-\bar{R}_i)}\right), \tag{5.7}$$

which is a more aggressive upper bound than the one from Proposition 2.0.4. However, this derivation does not work for every instance where one segment "dominates" all others - the above calculations are true for the particular example we looked at, however are not true in general.

Example 5.2.9. *Consider the following instance:*

i	N_i	R_{i1}	R_{i2}
1	1	100	50
2	1	12	10
3	1	7	23
4	1	11	10

Table 5.10: Example 5.2.9

This instance has a dual solution with objective value approximately 119.85 and a naive upper bound of 146. However, the formula (5.7) *gives a objective value of approximately 118.51, which is impossible. This is because the* ω_{11} *value obtained using the dual values from the system of linear equalities is actually negative, and thus the resulting solution is not feasible. If we instead attempt to solve the smaller LP model (i.e. relax the equalities into inequalities):*

$$\begin{aligned} \min\ & y_1, \\ s.t.\ & y_1 + (R_{12} - R_{11})\omega_{12} - R_{11}\sigma_{11} \geq 0, \\ & y_1 + (R_{11} - R_{12})\omega_{11} - R_{12}\sigma_{12} \geq 0, \\ & \sigma_{11} + \beta_{11} + \omega_{12} \geq N_1, \\ & \sigma_{12} + \beta_{12} + \omega_{11} \geq N_1, \\ & \sum_{i=2}^{n} \gamma_{i1} - \omega_{11} - \beta_{11} \geq 0, \\ & \sum_{i=2}^{n} \gamma_{i2} - \omega_{12} - \beta_{12} \geq 0. \end{aligned} \tag{5.8}$$

we get the following solution:

$$y_1 = 66.974, \sigma_{11} = 0.3734, \sigma_{12} = 1, \omega_{11} = 0, \omega_{12} = 0.592592, \beta_{11} = 0.033963, \beta_{12} = 0.$$

Plugging this into the overall objective value for the solution using the dual variables from (5.6) *for customer segments* 2 *through* 4 *gives a solution with value approximately* 127*, which is worse than the solution we obtain by solving the dual.*

Like the other approaches we have made toward establishing better upper bounds, this general method seems to be most effective when there is a large gap between the $\bar{R}_i$ and $\tilde{R}_j$ values, and ineffective when they are close. For example:

Example 5.2.10. *Consider changing the* R_{ij} *values slightly in Example 5.2.9, so that the* $\bar{R}_i$ *and* $\tilde{R}_j$ *values are closer.*

i	N_i	R_{i1}	R_{i2}
1	1	45	25
2	1	12	10
3	1	15	23
4	1	11	10

Table 5.11: An altered version of Table 5.10

This new instance has a naive upper bound of 91 and an optimal dual solution of 80.454. However, fixing the variables for segments $2-4$ *as in* (5.6) *gives us the following variable values:*

y_2	y_3	y_4
10	15	23

	Product 1	Product 2
γ_{2j}	0.0606	0
γ_{3j}	0	4
γ_{4j}	0.0294	0

	Product 1	Product 2
σ_{2j}	1.0606	1
σ_{3j}	1	5
σ_{4j}	1.0294	1

Table 5.12: Fixing variables as in (5.6)

Solving the resulting LP for the remaining variables gives the following solution for (5.8)*:*

$$y_1 = \sigma_{11} = \sigma_{12} = \omega_{12} = 0, \omega_{11} = 0.09, \beta_{11} = 1, \beta_{12} = 0.91.$$

This gives an objective value of roughly 139.05 for the solution, clearly worse than even our naive upper bound.

Overall, this strategy of fixing "basic" variables for smaller, less valuable customer segments and solving the resulting smaller LP has some promise. However, in order for this strategy to succeed, we must have more insight into the structure of the dual and how the variables interact with each other. Additionally, assuming structure in the data (such as uniformly logarithmic scaling, as in Subection 4.3.2), could provide scenarios where provable improvements to the upper bound are easier to come by.

5.3 An Exactly Solvable Subproblem

We can also attempt to look at some general (yet still somewhat constrained) variations of the original formulation (1.4). Take $n = 3$, $m = 2$, and fix $R_{12} = R_{31} = 0$. Then, we can write the

following Nonlinear Mixed Integer Program:

$$\begin{aligned}
\max \sum_{i=1}^{3}\sum_{j=1}^{2} N_i \pi_j \theta_{ij}, \\
\text{s.t. } \theta_{11}(\bar{R}_1 - \pi_1) &\geq 0, \\
\theta_{32}(\bar{R}_3 - \pi_3) &\geq 0, \\
\theta_{21}(R_{21} - \pi_2) &\geq 0, \\
\theta_{22}(R_{22} - \pi_2) &\geq 0, \\
\theta_{21}(R_{21} - \pi_2) &\geq R_{22}\theta_{21} - \pi_2, \\
\theta_{22}(R_{22} - \pi_2) &\geq R_{21}\theta_{22} - \pi_1, \\
\theta_{21} + \theta_{22} &= 1, \\
\theta_{11}, \theta_{21}, \theta_{22}, \theta_{32} \in \{0,1\}, \pi_1, \pi_2 &\geq 0.
\end{aligned}$$

We can eliminate the variable θ_{22} and the constraint $\theta_{21} + \theta_{22} = 1$ by setting $\theta_2 := \theta_{21}$ and replacing all mentions of θ_{22} with $(1 - \theta_{21})$:

$$\begin{aligned}
\max \sum_{i=1}^{3}\sum_{j=1}^{2} N_i \pi_j \theta_{ij}, \\
\text{s.t. } \theta_1(\bar{R}_1 - \pi_1) &\geq 0, \\
\theta_3(\bar{R}_3 - \pi_2) &\geq 0, \\
\theta_2(R_{21} - \pi_2) &\geq 0, \\
(1 - \theta_2)(R_{22} - \pi_2) &\geq 0, \\
\theta_2(R_{21} - \pi_2) &\geq R_{22}\theta_2 - \pi_2, \\
(1 - \theta_2)(R_{22} - \pi_2) &\geq R_{21}(1 - \theta_2) - \pi_1, \\
\theta_1, \theta_2, \theta_3 \in \{0,1\}, \pi_1, \pi_2 &\geq 0.
\end{aligned} \tag{5.9}$$

For the nonlinear model, it is possible to solve exactly in the following way: for an assignment $\theta \in \{0,1\}^3$, let P^θ be the convex hull of the associated LP given by fixing the assignment θ for (5.9). Then $\mathcal{P} := \bigcup_{\theta \in \{0,1\}^3} P^\theta$ is also convex, and moreover the optimal solution to (5.9) will be an extreme point of $\mathcal{P}$, easily found by standard convex optimization techniques. The advantage of a brute-force approach is that the search space is quite small - $\mathcal{P}$ is the union of at most 8 convex hulls in relatively low-dimension space. It remains a possibility that we could break a larger problem down into many small pieces, each resembling this general formulation. By solving each of them exactly and stitching the solutions back together in an intelligent way, it may be possible to construct an algorithm or heuristic with a better approximation ratio or more efficient runtime.

For completeness, we also include the linearization of (5.9) and its dual:

$$
\begin{aligned}
\max \sum_{i=1}^{3}\sum_{j=1}^{2} N_i p_{ij}, & \\
\text{s.t. } \theta_2 R_{21} - p_{21} &\geq \theta_2 R_{22} - \pi_2, \\
(1-\theta_2) R_{22} - p_{22} &\geq (1-\theta_2) R_{21} - \pi_1, \\
\theta_1 R_{11} - p_{11} &\geq 0, \\
\theta_2 R_{21} - p_{21} &\geq 0, \\
(1-\theta_2) R_{22} - p_{22} &\geq 0, \\
\theta_3 R_{32} - p_{32} &\geq 0, \\
\theta_1 &\leq 1, \\
\theta_2 &\leq 1, \\
\theta_3 &\leq 1, \\
p_1 &\leq \pi_1, \\
p_{21} &\leq \pi_1, \\
p_{22} &\leq \pi_2, \\
p_3 &\leq \pi_2, \\
p_{11} &\geq \pi_1 - R_{11}(1-\theta_1), \\
p_{21} &\geq \pi_1 - \bar{R}_2(1-\theta_2), \\
p_{22} &\geq \pi_2 - \bar{R}_2\theta_2, \\
p_{32} &\geq \pi_2 - R_{32}(1-\theta_3), \\
\theta_i, p_{ij}, \pi_j &\geq 0.
\end{aligned}
$$

$$
\begin{aligned}
\min y_1 + y_2 + y_3 + \bar{R}_1\gamma_{11} + \bar{R}_2\gamma_{21} + \bar{R}_2\gamma_{22} + \bar{R}_3\gamma_{32}, & \\
\text{s.t. } -R_{11}\sigma_{11} + y_1 + \bar{R}_1\gamma_{11} &\geq 0, \\
\omega_{22}(R_{22} - R_{21}) - R_{21}\sigma_{21} + y_2 + \bar{R}_2\gamma_{21} &\geq 0, \\
\omega_{21}(R_{21} - R_{22}) - \sigma_{22} + y_2 + \bar{R}_2\gamma_{22} &\geq 0, \\
-R_{32}\sigma_{32} + y_3 + \bar{R}_3\gamma_{32} &\geq 0, \\
\sigma_{11} + \beta_{11} - \gamma_{11} &\geq N_1, \\
\omega_{22} + \sigma_{21} + \beta_{21} - \gamma_{21} &\geq N_2, \\
\omega_{21} + \sigma_{22} + \beta_{22} - \gamma_{22} &\geq N_2, \\
\sigma_{32} + \beta_{32} - \gamma_{32} &\geq N_3, \\
\gamma_{11} + \gamma_{21} - \beta_{11} - \beta_{21} - \omega_{21} &\geq 0, \\
\gamma_{22} + \gamma_{32} - \beta_{22} - \beta_{32} - \omega_{22} &\geq 0, \\
\beta, \sigma, y, \omega, \gamma &\geq 0.
\end{aligned}
$$

Chapter 6

Conclusion and Future Research Directions

In this thesis, we have reviewed the historical background of the Maximum Utility Problem as well as the general array of algorithms and heuristics that have been applied to it in the literature. In addition, we have developed new analytic techniques for MUP, providing numerous avenues for proving both best and worst-case approximation ratios. To achieve this, we identified key scenarios in which the problem can be solved exactly and for which there exist algorithms with good approximation ratios, building significantly off the research of previous authors. We also analyzed both the primal and dual of the linear relaxation of the model, testing and working out solutions for many different examples. These exercises increased our overall understanding of the primal and dual, and offer promising ideas for future improvements to a generalized upper bound for MUP.

Overall, there remain significant open questions. We have yet to discover a definitive answer as to whether the original 1988 Dobson-Kalish algorithm is polytime, and it remains to be seen if there are other structured cases under which MUP can be solved directly or approximated well. Our research presents multiple opportunities for improvements on the latter - it is highly possible that stronger analysis on both the primal relaxation and dual can significantly improve the worst-case upper bound. Also, while we have shown that the Maximum Utility Problem can be solved exactly when the reservation price matrix is rank 1, we have not presented any additional arguments for a more general low-rank case. Since the rank 1 case itself is not highly applicable, improved algorithms for the low-rank case will be highly beneficial in applications of the model. Additionally, there may be specific structured cases for the underlying datasets that allow for better algorithms than the ones presented in this thesis. Furthermore, we have not provided computational experiments for the MaxR$^+$ algorithm presented in Chapter 4 - such testing will provide a better assessment of its effectiveness. We hope that future researchers find this thesis helpful towards these goals.

Bibliography

[1] Gagan Aggarwal, Tomás Feder, Rajeev Motwani, and An Zhu. "Algorithms for multi-product pricing". In: *Automata, languages and programming*. Vol. 3142. Lecture Notes in Comput. Sci. Springer, Berlin, 2004, pp. 72–83. DOI: 10.1007/978-3-540-27836-8_9. URL: https://doi.org/10.1007/978-3-540-27836-8_9.

[2] T.W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley Series in Probability and Statistics. Wiley, 2003. ISBN: 9780471360919. URL: https://books.google.ca/books?id=Cmm9QgAACAAJ.

[3] C. Arbib, O. E. Karaşan, and M. Ç. Pınar. "On envy-free perfect matching". In: *Discrete Appl. Math.* 261 (2019), pp. 22–27. ISSN: 0166-218X. DOI: 10.1016/j.dam.2018.03.034. URL: https://doi.org/10.1016/j.dam.2018.03.034.

[4] Sanjeev Arora. "Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems". In: *J. ACM* 45.5 (1998), pp. 753–782. ISSN: 0004-5411. DOI: 10.1145/290179.290180. URL: https://doi.org/10.1145/290179.290180.

[5] Giorgio Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. 1st. Berlin, Heidelberg: Springer-Verlag, 1999. ISBN: 3540654313.

[6] Mourad Baıou and Francisco Barahona. "Stackelberg Bipartite Vertex Cover and the Preflow Algorithm". In: *Algorithmica* 74.3 (2016), pp. 1174–1183. ISSN: 0178-4617. DOI: 10.1007/s00453-015-9993-x. URL: https://doi.org/10.1007/s00453-015-9993-x.

[7] Dimitris Bertsimas, Martin S. Copenhaver, and Rahul Mazumder. "Certifiably Optimal Low Rank Factor Analysis". In: (2016). arXiv: 1604.06837 [stat.ME].

[8] Toni Böhnlein, Stefan Kratsch, and Oliver Schaudt. "Revenue maximization in Stackelberg pricing games: beyond the combinatorial setting". In: *Math. Program.* 187.1-2, Ser. A (2021), pp. 653–695. ISSN: 0025-5610. DOI: 10.1007/s10107-020-01495-0. URL: https://doi.org/10.1007/s10107-020-01495-0.

[9] Patrick Briest, Martin Hoefer, and Piotr Krysta. "Stackelberg network pricing games". In: *Algorithmica* 62.3-4 (2012), pp. 733–753. ISSN: 0178-4617. DOI: 10.1007/s00453-010-9480-3. URL: https://doi.org/10.1007/s00453-010-9480-3.

[10] Patrick Briest and Piotr Krysta. "Buying cheap is expensive: approximability of combinatorial pricing problems". In: *SIAM J. Comput.* 40.6 (2011), pp. 1554–1586. ISSN: 0097-5397. DOI: 10.1137/090752353. URL: https://doi.org/10.1137/090752353.

[11] Rainer E. Burkard. "Monge properties, discrete convexity and applications". In: *European Journal of Operational Research* 176.1 (2007), pp. 1–14. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2005.04.050. URL: https://www.sciencedirect.com/science/article/pii/S0377221705008702.

[12] Rainer E. Burkard, Bettina Klinz, and Rüdiger Rudolf. "Perspectives of Monge properties in optimization". In: *Discrete Applied Mathematics* 70.2 (1996), pp. 95–161. ISSN: 0166-218X. DOI: https://doi.org/10.1016/0166-218X(95)00103-X. URL: https://www.sciencedirect.com/science/article/pii/0166218X9500103X.

[13] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 1998, pp. x+355. ISBN: 0-471-55894-X.

[14] George B. Dantzig. "Linear programming". In: vol. 50. 1. 50th anniversary issue of Operations Research. 2002, pp. 42–47. DOI: 10.1287/opre.50.1.42.17798. URL: https://doi.org/10.1287/opre.50.1.42.17798.

[15] Erik D. Demaine, Uriel Feige, Mohammadtaghi Hajiaghayi, and Mohammad R. Salavatipour. "Combination can be hard: approximability of the unique coverage problem". In: *SIAM J. Comput.* 38.4 (2008), pp. 1464–1483.

[16] Derya Demirtas. "Worst-Case Complexity Analyses for the Dobson-Kalish Optimal Pricing Algorithm and its Relatives". MA thesis. Aug. 2010.

[17] Gregory Dobson and Shlomo Kalish. "Positioning and Pricing a Product Line". In: *Marketing Science* 7.2 (1988), pp. 107–210. DOI: 10.1287/mksc.7.2.107. URL: https://doi.org/10.1287/mksc.7.2.107.

[18] Rémi Dubroca, Christophe Junor, and Antoine Souloumiac. "Weighted NMF for high-resolution mass spectrometry analysis". In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. 2012, pp. 1806–1810.

[19] Jean-Pierre Dussault, Patrice Marcotte, Sébastien Roch, and Gilles Savard. "A smoothing heuristic for a bilevel pricing problem". In: *European Journal of Operational Research* 174.3 (2006), pp. 1396–1413. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2004.07.076. URL: https://www.sciencedirect.com/science/article/pii/S0377221705004947.

[20] Jack Edmonds and Richard M. Karp. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems". In: *J. ACM* 19.2 (Apr. 1972), pp. 248–264. ISSN: 0004-5411. DOI: 10.1145/321694.321699. URL: https://doi.org/10.1145/321694.321699.

[21] Cristina G. Fernandes and Rafael C. S. Schouery. "Approximation algorithms for the max-buying problem with limited supply". In: *Algorithmica* 80.11 (2018), pp. 2973–2992. ISSN: 0178-4617. DOI: 10.1007/s00453-017-0364-7. URL: https://doi.org/10.1007/s00453-017-0364-7.

[22] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. "Nonnegative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis". In: *Neural computation* 21 (Oct. 2008), pp. 793–830. DOI: 10.1162/neco.2008.04-08-771.

[23] Iftah Gamzu and Danny Segev. "A sublogarithmic approximation for tollbooth pricing on trees". In: *Math. Oper. Res.* 42.2 (2017), pp. 377–388. ISSN: 0364-765X. DOI: 10.1287/moor.2016.0803. URL: https://doi.org/10.1287/moor.2016.0803.

[24] Fabrizio Grandoni and Thomas Rothvoß. "Pricing on Paths: A PTAS for the Highway Problem". In: *SIAM Journal on Computing* 45.2 (2016), pp. 216–231. DOI: 10.1137/140998846. eprint: https://doi.org/10.1137/140998846. URL: https://doi.org/10.1137/140998846.

[25] Paul Grigas, Alfonso Lobos, Zheng Wen, and Kuang-Chih Lee. "Optimal Bidding, Allocation, and Budget Spending for a Demand-Side Platform with Generic Auctions". In: *SSRN Electronic Journal* (Jan. 2021). DOI: 10.2139/ssrn.3841306.

[26] Oktay Günlük. "A pricing problem under Monge property". In: *Discrete Optim.* 5.2 (2008), pp. 328–336. ISSN: 1572-5286. DOI: 10.1016/j.disopt.2006.06.005. URL: https://doi.org/10.1016/j.disopt.2006.06.005.

[27] Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. "On profit-maximizing envy-free pricing". In: *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM, New York, 2005, pp. 1164–1173.

[28] Venkatesan Guruswami and Euiwoong Lee. "Nearly optimal NP-hardness of unique coverage". In: *SIAM J. Comput.* 46.3 (2017), pp. 1018–1028.

[29] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. Seventh. New York, NY, USA: McGraw-Hill, 2001.

[30] A. J. Hoffman. "On simple linear programming problems". In: *Proc. Sympos. Pure Math., Vol. VII*. Amer. Math. Soc., Providence, R.I., 1963, pp. 317–327.

[31] https://arcteryx.com/ca/en/c/mens/shell-jackets. In: (2022). [Website; last accessed July 29, 2022].

[32] https://www.redfin.ca/on/kitchener. In: (2022). [Website; last accessed July 27, 2022].

[33] https://www.ticketmaster.ca/event/10005B5CBFDE3D6A?tfl=Toronto_Blue_Jays-Schedule-Toronto_Blue_Jays:_Schedule:_Schedule-web-x0-unknown-unknown&_ga=2.74952855.401471260.1658934757-203497847.1658256134. In: (2022). [Website; last accessed July 27, 2022].

[34] https://www.toyota.ca/toyota/en/. In: (2022). [Website; last accessed July 27, 2022].

[35] Sen Jia and Yuntao Qian. "Constrained Nonnegative Matrix Factorization for Hyperspectral Unmixing". In: *IEEE Transactions on Geoscience and Remote Sensing* 47.1 (2009), pp. 161–173. DOI: 10.1109/TGRS.2008.2002882.

[36] Gwenaël Joret. "Stackelberg network pricing is hard to approximate". In: *Networks* 57.2 (2011), pp. 117–120. ISSN: 0028-3045. DOI: 10.1002/net.20391. URL: https://doi.org/10.1002/net.20391.

[37] Wagner A. Kamakura and Gary J. Russell. "A Probabilistic Choice Model for Market Segmentation and Elasticity Structure". In: *Journal of Marketing Research* 26.4 (1989), pp. 379–390. ISSN: 00222437. URL: http://www.jstor.org/stable/3172759 (visited on 04/09/2023).

[38] Mehdi Karimi, Somayeh Moazeni, and Levent Tunçel. "A utility theory based interactive approach to robustness in linear optimization". In: *J. Global Optim.* 70.4 (2018), pp. 811–842. ISSN: 0925-5001. DOI: 10.1007/s10898-017-0581-2. URL: https://doi.org/10.1007/s10898-017-0581-2.

[39] Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. "A (Slightly) Improved Approximation Algorithm for Metric TSP". In: *CoRR* abs/2007.01409 (2020). arXiv: 2007.01409. URL: https://arxiv.org/abs/2007.01409.

[40] Rajeev Kohli and Vijay Mahajan. "A Reservation-Price Model for Optimal Pricing of Multiattribute Products in Conjoint Analysis". In: *Journal of Marketing Research* 28.3 (1991), pp. 347–354. ISSN: 00222437. URL: http://www.jstor.org/stable/3172870 (visited on 04/09/2023).

[41] M. Köppe, M. Queyranne, and C. T. Ryan. "Parametric integer programming algorithm for bilevel mixed integer programs". In: *J. Optim. Theory Appl.* 146.1 (2010), pp. 137–150. ISSN: 0022-3239. DOI: 10.1007/s10957-010-9668-3. URL: https://doi.org/10.1007/s10957-010-9668-3.

[42] Ursula G. Kraus and Candace Arai Yano. "Product line selection and pricing under a share-of-surplus choice model". In: *European Journal of Operational Research* 150.3 (2003). Financial Modelling, pp. 653–671. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(02)00522-2. URL: https://www.sciencedirect.com/science/article/pii/S0377221702005222.

[43] Martine Labbé, Patrice Marcotte, and Gilles Savard. "A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing". In: *Management Science* 44 (Dec. 1998), pp. 1608–1622. DOI: 10.1287/mnsc.44.12.1608.

[44] Martine Labbé, Miguel A. Pozo, and Justo Puerto. "Computational comparisons of different formulations for the Stackelberg minimum spanning tree game". In: *International Transactions in Operational Research* 28.1 (2021), pp. 48–69. DOI: https://doi.org/10.1111/itor.12680. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12680.

[45] Arthur Lee and Bruce Xu. "Classifying Approximation Algorithms: Understanding the APX Complexity Class". In: *CoRR* abs/2111.01551 (2021). arXiv: 2111.01551. URL: https://arxiv.org/abs/2111.01551.

[46] Jeffrey McGill and Garrett van Ryzin. "Revenue Management: Research Overview and Prospects". In: *Transportation Science* 33 (May 1999), pp. 233–256. DOI: 10.1287/trsc.33.2.233.

[47] John A. Muckstadt and Amar Sapra. *Principles of inventory management*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2010, pp. xviii+339. ISBN: 978-0-387-24492-1. DOI: 10.1007/978-0-387-68948-7. URL: https://doi.org/10.1007/978-0-387-68948-7.

[48] T. G. J. Myklebust, M. A. Sharpe, and L. Tunçel. "Efficient heuristic algorithms for maximum utility product pricing problems". In: *Comput. Oper. Res.* 69 (2016), pp. 25–39. ISSN: 0305-0548. DOI: 10.1016/j.cor.2015.11.013. URL: https://doi.org/10.1016/j.cor.2015.11.013.

[49] Lipeng Ning, Tryphon T Georgiou, Allen Tannenbaum, and Stephen P Boyd. "Linear models based on noisy data and the Frisch scheme". In: *siam REVIEW* 57.2 (2015), pp. 167–197.

[50] R. Shioda, L. Tunçel, and B. Hui. "Applications of deterministic optimization techniques to some probabilistic choice models for product pricing using reservation prices". In: *Pac. J. Optim.* 10.4 (2014), pp. 767–808. ISSN: 1348-9151.

[51] R. Shioda, L. Tunçel, and T. G. J. Myklebust. "Maximum utility product pricing models and algorithms based on reservation price". In: *Comput. Optim. Appl.* 48.2 (2011), pp. 157–198. ISSN: 0926-6003. DOI: 10.1007/s10589-009-9254-5. URL: https://doi.org/10.1007/s10589-009-9254-5.

[52] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. "A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications". In: *IEEE Transactions on Evolutionary Computation* 22.2 (2018), pp. 276–295. DOI: 10.1109/TEVC.2017.2712906.

[53] Guang-Jing Song and Michael K. Ng. "Nonnegative low rank matrix approximation for nonnegative matrices". In: *Appl. Math. Lett.* 105 (2020), pp. 106300, 7. ISSN: 0893-9659. DOI: 10.1016/j.aml.2020.106300. URL: https://doi.org/10.1016/j.aml.2020.106300.

[54] Kalyan T. Talluri and Garret J. Ryzin. *The Theory and Practice of Revenue Management*. Springer New York, 2005. DOI: https://doi.org/10.1007/b139000. URL: https://link.springer.com/book/10.1007/b139000.

[55] Levent Tunçel. "Optimization Based Approaches to Product Pricing". Proc. IV. International Conference on Business, Management and Economics. 2008. URL: https://www.math.uwaterloo.ca/~ltuncel/publications/icbme2008.pdf.

[56] L. Vicente, G. Savard, and J. Judice. "Discrete linear bilevel programming problem". In: *J. Optim. Theory Appl.* 89.3 (1996), pp. 597–614. ISSN: 0022-3239. DOI: 10.1007/BF02275351. URL: https://doi.org/10.1007/BF02275351.

[57] G. W. "Revenue Management: Hard-Core Tactics for Market Domination". In: *Cornell Hotel and Restaurant Administration Quarterly* 38.2 (1997), pp. 17–17. DOI: 10.1177/001088049703800218. eprint: https://doi.org/10.1177/001088049703800218. URL: https://doi.org/10.1177/001088049703800218.

[58] H. Paul Williams. *Logic and integer programming*. Vol. 130. International Series in Operations Research & Management Science. Springer, New York, 2009, pp. xii+155. ISBN: 978-0-387-92279-9.