ECOLE POLYTECHNIQUE DE BRUXELLES

MA1-ORO/MA2-IRIFS

Combinatorial Optimization - [INFO-F424]

Assortment Planning Problem Project

https://github.com/Mudezer/AP_IP

Awen Jacq-Bodet, 000601155

Elizabeth Gandibleux, 000601150

Loïc Bermudez, 440945

Renaud Chicoisne

Cristian Aguayo

2023 - 2024

Table des matières

1 Introduction

1.1 Introduction

The Assortment Planning Problem is a critical operational issue that arises in retail and inventory management industries such as retailing, airlines and consumer electronics. Given a set of product that are differentiated by price, quality and possibly other attributes, one has to determine the optimal mix of products to offer to customers in a retail setting to maximize the overall business performance. Such decisions become particularly important when different products are substitutable and customers exhibit a substitution behavior. For example a customer may a-priori prefer a product A to a product B, but may still be willing to buy product B if product A is not offered or not available anymore. The substitution behavior can be assortment-based (or static), i.e., unaffected by the availability of products and depends only on the specific assortment of products, or it can be stock-out-based (or dynamic), i.e., driven by stock-out events and/or availability of products. Due to its combinatorial nature, it has been shown that the APP is a NP-Hard problem, thereby giving rise to complex optimization models that are computationally challenging. As of today all the Assortment Planning Problem forms one of the core problem domains in revenue management and many variant of these problems have been studies extensively in the literature and is extensively used by many companies (for example Amazon, Alibaba Express, ...)[Vin].

1.2 Model Formulation

This project is focused on the static case, where a retailer want to determine which products to propose to its customer in order to maximize its expected revenue. The retailer will consider a set of product he can propose to its customers :

$$\mathscr{I} = \{1, ..., n\} \tag{1.1}$$

Selling a product i generates a net revenue of $r_i > 0$ for him. It is also assumed that the products are sorted in decreasing order of revenue $r_1 > r_2 > ... > r_n > 0$ and that selling nothing does come with a revenue $r_0 \geq 0$. Each product i will be purchased according to a particular probability coming from a discrete model, the

multinomial logit model written :

$$P_i(S) = \frac{e^{\mu_i}}{e^{\mu_0} + \sum_{j \in S} e^{\mu_j}} = \frac{e^{\mu_i}}{1 + \sum_{j \in S} e^{\mu_j}}, \quad \forall i \in \mathscr{I} \cup \{0\} \tag{1.2}$$

Where :

— The mean utility, a subjective indicator composed of a deterministic part and a stochastic part, reflecting the tendency for a customer to buy a certain product j belonging to the initial set $\mathscr{I}$:

$$(\mu_j)_{j \in S} \tag{1.3}$$

— The baseline utility, being the utility for the customers to buy nothing :

$$\mu_0 = 0 \tag{1.4}$$

— The set of alternatives **S** made available to the customers.

In fine, the objective is to determine the best subset of product **S** (i.e. the set of alternative) selected from the initial set $\mathscr{I}$ to maximize the net revenue given that the net revenues depend on the stochastic behaviour of the customers. The problem is posed as the following combinatorial optimization problem :

$$\text{(AP)} \quad \max_{S \subseteq \mathscr{I}} \left\{ r_0 \cdot P_0(S) + \sum_{i \in S} r_i \cdot P_i(S) \right\} \tag{1.5}$$

1.3 Objectives

This work has been split in four parts :

1. Finding a Primal and Dual Linear Programming model from the combinatorial form while studying the validity of the different models. It is also asked to explain the relationships between the dual and primal models.
2. Proving, through the Primal and Dual models found in the first part, that the AP can be solved through a algorithm in polynomial time. In addition, a polynomial time algorithm is to be proposed to solve the AP.
3. Finding a Mixed Integer Linear Program formulation from the Primal model by adding a new constraint thus making it dynamic (making it dependent of stock size), then comparing the effect of different values for the constraint by implementing it. It is also required to find a Lagrangian relaxation model and algorithm from an alternative model for the dynamic one then to study the complexity of the algorithm while building an heuristic out of it.
4. Testing and comparing all the models and algorithms from the previous parts on different instances

sizes. The comparison is based on the minimum, average and maximum optimal values and solution times.

1.4 Specifications

1.4.1 Experimental Environment

For the experimental implementations, the algorithms were implemented in the Julia language, using the Gurobi solver and available in the following github repository : `https://github.com/Mudezer/AP_IP` . The distribution 1.10 of Julia was used on an Apple MacBook Pro laptop which specifications are :

— CPU : 2.7 GHz Intel Core i7 (Four Cores)

— RAM : 16 Go 2133 MHz LPDDR3

All algorithms were implemented as single processes with no threads.

2 From Combinatorial Optimization to Linear Programming

This section focuses on finding a Primal and Dual Linear Programming model from the combinatorial model.

2.1 Integer Programming Model

Due to the non-linearity and discrete nature of the combinatorial model, a first step is to find an integer programming model which allow an easier linearization into the desired linear programming.

2.1.1 Set of alternatives

In order to find the Integer Programming model, it is needed to introduce an integer variable to represent the inclusion/exclusion decision characteristic to the Assortment Planning Problem which is about *choosing which product to propose in the set of alternatives* while maximizing the expected revenue. Thus, we want to maximize the number of element to be taken from $\mathscr{I}$ to S.
We then introduce a new set of binary variables :

$$X = \{x_1, x_2, ..., x_n\} \quad \text{with} \quad x_i = \begin{cases} 1 & \text{product } i \text{ is assigned to set } S \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathscr{I} \tag{2.1}$$

These binary variables fully represent the decision of putting the product i into the set alternatives. Something to note, is that x_0 represent the fact of choosing nothing, therefore we have :

$$x_0 = \begin{cases} 1 & \text{, then } x_i = 0 \quad \forall i \in \mathscr{I} \\ 0 & \text{, otherwise} \end{cases} \tag{2.2}$$

2.1.2 Assortment Problem - Integer Programming

After the definition of the binary decision variables, the Integer Programming model is straightforward to find :

By using **1.2** in **1.5**, we get :

$$\max_{S\subseteq\mathscr{I}} \quad r_0 \cdot \frac{e^{\mu_0}}{e^{\mu_0}+\sum_{j\in S} e^{\mu_j}} + \sum_{i\in S} r_i \cdot \frac{e^{\mu_i}}{e^{\mu_0}+\sum_{j\in S} e^{\mu_j}} \tag{2.3}$$

Then, by using **1.4** we can simplify into :

$$\max_{S\subseteq\mathscr{I}} \quad r_0 \cdot \frac{1}{1+\sum_{j\in S} e^{\mu_j}} + \sum_{i\in S} r_i \cdot \frac{e^{\mu_i}}{1+\sum_{j\in S} e^{\mu_j}} \tag{2.4}$$

This expression can be unified this way :

$$\max_{S\subseteq\mathscr{I}} \quad \frac{r_0+\sum_{i\in S} r_i \cdot e^{\mu_i}}{1+\sum_{j\in S} e^{\mu_j}} \tag{2.5}$$

As previously said, we want to maximize the expected revenue produced by the set of alternatives which come back to maximizing the items we take from the initial set $\mathscr{I}$. We can then implement the new set of variables from **2.1**, which allow us to iterate over the set $\mathscr{I}$ instead of iterating over the set S.
Therefore, the sum $\sum_{i\in S} r_i \cdot e^{\mu_i}$ on the numerator can be replaced by the sum $\sum_{i=1}^{n} x_i r_i e^{\mu_i}$ which is equivalent as the variable x_i is simply an indicator of whether we take the product i to put it in S. And the sum $\sum_{j\in S} e^{\mu_j}$ can also be replace by the equivalent sum $\sum_{i=1}^{n} x_i e^{\mu_i}$ as we introduce the decision variable x_i

$$(AP-IP) \quad \max_{x\in\{0,1\}^n} \quad \frac{r_0+\sum_{i=1}^{n} x_i r_i e^{\mu_i}}{1+\sum_{i=1}^{n} x_i e^{\mu_i}} \tag{2.6}$$

2.1.3 Validity and Nonlinearity

The AP-IP model is a valid formulation for the Assortment Planning Problem because :

— The binary decision variables x_i correctly represent the inclusion or exclusion of products in the alternatives proposed to the customer. It is the sole way to get a subset from the initial one, also it is the sole way for the retailer to maximize the expected revenue as the others factors are out of his control

— The objective function is appropriate for maximizing the expected revenue from the alternatives. It takes into account the utility of each product and normalizes by the total utility, reflecting the probability of each product being chosen by the customer. Furthermore, it takes into account the case when nothing is bought.

— In term of constrains it is not easily seen, the objective function is assured to be non negative, in case of no product being purchased there is the r_0 term which correspond to the revenue of buying nothing and from the model formulation section, it is assumed that $r_0 \geq 0$, thus asserting non negativity.

— The non-linearity of the model is easily shown as the numerator $r_0 + \sum_{i=1}^{n} x_i r_i e^{\mu_i}$ is a linear function of x_i and the denominator $1 + \sum_{i=1}^{n} x_i e^{\mu_i}$ also is a linear function of x_i making the fraction a fraction of objective function and so non linear.

The AP-IP model is nonlinear implying more complex algorithms and requiring higher computational costs. This non linearity is due to the normalization of each potential revenue by the total utility of the set of alternatives. In fact, a programming model is nonlinear if the objective function include any non linear terms such as products of variables, powers, exponential,... In this case, each variable is divided by a sum including other variables which is the analog to a product, thus making it non linear.

2.2 Linear Integer Program

This section is about linearizing the Integer Program model.

2.2.1 New variables

A simple way to make the AP-IP linear would be to introduce new variables to make disappear the division making the previous model non linear. Therefore, we will consider two new variables to be introduced in the formulation :

$$y_0 := \frac{1}{1 + \sum_{j=1}^{n} x_j e^{\mu_j}} \tag{2.7}$$

$$y_i := \frac{x_i e^{\mu_i}}{1 + \sum_{j=1}^{n} x_j e^{\mu_j}} \quad , \forall i \in \mathscr{I} \tag{2.8}$$

There are some links we can do between the new variables and the data we have so far :
Firstly, We know that **1.2** is the probability of buying the product i from the set of alternatives S. Secondly, it is precised that the product i belong to the set $\mathscr{I} \cup \{0\}$, this set takes two general cases : the case when nothing is purchased indicated by $\{0\}$, and the case when a product is purchased indicated by the set of all possible product $\mathscr{I}$. Also we can extend the set $\mathscr{I}$ by adding the 0th element being the fact of taking nothing Hence we have two possible cases of probability :

1. The case when nothing is selected, thus $i = 0$. We can then write **1.2** this way :

$$P_0(S) = \frac{e^{\mu_0}}{e^{\mu_0} + \sum_{j \in S} e^{\mu_j}} \tag{2.9}$$

   By using the fact that the mean utility of taking nothing is equal to 0 (**1.4**) and as e^0 is equal to 1, we can rewrite the expression this way :

$$P_0(S) = \frac{1}{1 + \sum_{j \in S} e^{\mu_j}} \tag{2.10}$$

   In the same way as in the previous section, we can introduce the new variables from **2.1**, the expres-

sion would become :

$$P_0(S) = \frac{1}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} = y_0 \tag{2.11}$$

2. The case when the product i is selected. We can then write **1.2** this way :

$$P_i(S) = \frac{e^{\mu_i}}{e^{\mu_0} + \sum_{j \in S} e^{\mu_j}} \tag{2.12}$$

By using the fact that the mean utility of taking nothing is equal to 0 (**1.4**) and as e^0 is equal to 1 , we can rewrite the expression this way :

$$P_i(S) = \frac{e^{\mu_i}}{1+\sum_{j \in S} e^{\mu_j}} \tag{2.13}$$

In the same way as in the previous section, we can introduce the new variables from **2.1**, the expression would become :

$$P_i(S) = \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} = y_i \quad , \forall i \in \mathscr{I} \tag{2.14}$$

Through both development, we have found that y_0 and y_i are both probabilities, respectively the probability of taking nothing and the probability of taking the product i.

We can even make a link between y_0 and y_i. Let's start by taking **2.8** :

$$y_i = \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} \quad , \forall i \in \mathscr{I} \tag{2.15}$$

If we look at the numerator, the new variable x_i has two possibles values : either 0, if the product is not taken in the set of alternatives or 1 if the product is taken in the set of alternatives. This lies then in two possibilities :

— When $x_i = 0$:

$$y_i = \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}}|_{x_i=0} \tag{2.16}$$

Thus giving :

$$y_i = 0 \tag{2.17}$$

— When $x_i = 1$:

$$y_i = \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}}|_{x_i=1} \tag{2.18}$$

Thus giving :

$$y_i = \frac{1 \times e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} = \frac{1}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} \times e^{\mu_i} \tag{2.19}$$

By using **2.7**, we can replace the fraction by y_0 :

$$y_i = y_0 \times e^{\mu_i} \forall i \in \mathscr{I} \tag{2.20}$$

By developing both cases we have found two possible values for the probability y_i : either 0 or $y_0 e^{\mu_i}$, $\forall i \in \mathscr{I}$

2.2.2 Linearization

We start by rewriting **2.6** :

$$\text{(AP-IP)} \quad \max_{x\in\{0,1\}^n} \quad r_0 \times \frac{1}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} + \sum_{i=1}^{n} r_i \times \frac{x_i e^{\mu_i}}{\sum_{j=1}^{n} x_j e^{\mu_j}} \tag{2.21}$$

By using the new variables from **2.7** and **2.8**, we can replace each fraction in the expression. Also maximizing the amount of product we put in the set of alternatives is equivalent to maximizing the probability of customers to purchase the products in the set :

$$\max_{y, y_0 \geq 0} \quad r_0 \times y_0 + \sum_{i=1}^{n} r_i \times y_i = r_0 y_0 + \sum_{i=1}^{n} r_i y_i \tag{2.22}$$

Then by using the conclusion obtained through **2.11, 2.14, 2.17, 2.20**, we can derive a two constrains :

1. As y_0 and $y_i, \forall i \in \mathscr{I}$ are probabilities belonging to the same set (i.e. probability of taking nothing from **S** or taking the product i from **S**), we can say that the sum of all probabilities is equal to one :

$$y_0 + \sum_{i=1}^{n} y_i = 1 \tag{2.23}$$

2. y_i can take two possible values : either 0 or $y_0 e^{\mu_i}$, thus giving :

$$y_i \in \left\{0, y_0 e^{\mu_i}\right\} \quad , \forall i \in \mathscr{I} \tag{2.24}$$

Finally, by assembling all three facts we found a Linear Integer Program model of the problem :

$$\text{(AP-IPL)} \quad \max_{y, y_0 \geq 0} \quad r_0 y_0 + \sum_{i=1}^{n} r_i y_i \tag{2.25}$$

$$\text{s.t.} \quad y_0 + \sum_{i=1}^{n} y_i = 1 \tag{2.26}$$

$$y_i \in \left\{0, y_0 e^{\mu_i}\right\} \quad , \forall i \in \mathscr{I} \tag{2.27}$$

2.3 Primal and Dual relaxations

2.3.1 Continuous relaxation

An easy relaxation for (AP-IPL) would be on the constraint on the values of y_i (**2.27**, thus giving the following model :

$$\text{(AP-L)} \quad \max_{y, y_0 \geq 0} \quad r_0 y_0 + \sum_{i=1}^{n} r_i y_i \tag{2.28}$$

$$\text{s.t.} \quad y_0 + \sum_{i=1}^{n} y_i = 1 \tag{2.29}$$

$$y_i \leq y_0 e^{\mu_i} \quad , \forall i \in \mathscr{I} \tag{2.30}$$

First, we can rewrite the AP-L problem to make the reasoning easier :

$$\text{(Primal)} \quad \begin{cases} \max \quad G = r_0 y_0 + \sum_{i=1}^{n} r_i y_i \\ y_0 + \sum_{i=1}^{n} y_i = 1 \\ -y_0 e^{\mu_i} + y_i \leq 0 \quad , \forall i \in \mathscr{I} \\ y \geq 0 \quad , y_0 \geq 0 \end{cases} \tag{2.31}$$

Then we define the dual variables, one for each constraints :

$$\begin{cases} y_0 + \sum_{i=1}^{n} y_i = 1 \quad [\pi_0] \\ -y_0 e^{\mu_i} + y_i \leq 0 \quad [\pi_i] \end{cases} \tag{2.32}$$

With the dual variables we can form the Dual objective function. As the primal is a maximization problem the dual is then a minimization one, where the dual variables coefficients are the constraining coefficient of the primal constraints. Something to note is that as the objective function being primal or dual, consider all the products in the set **S**, we must add the sum from 1 to n before the second dual variable :

$$\min \quad G' = \pi_0 \times 1 + \sum_{i=1}^{n} \pi_i \times 0 = \pi_0 \tag{2.33}$$

We can then form the dual constraints derived from the coefficient in the primal problem :

— From the first coefficient of the primal problem and the coefficient of the corresponding variable we have :

$$\pi_0 - \pi_i \times e^{\mu_i} \leq r_0 \quad \forall i \in \mathscr{I} \tag{2.34}$$

As in the dual objective function we iterate over all products in $\mathscr{I}$, thus giving us :

$$\pi_0 - \sum_{i=1}^{n} \pi_i e^{\mu_i} \geq r_0 \tag{2.35}$$

— From the second coefficient of the primal problem and the coefficient of the corresponding variable we have :

$$\sum_{i=1}^{n} \pi_0 + \pi_i \geq \sum_{i=1}^{n} r_i \Leftrightarrow n \times \pi_0 + \pi_i \geq \sum_{i=1}^{n} r_i \tag{2.36}$$

This constraint can be fragmented in n easier constraints, hence resulting in :

$$\pi_0 + \pi_i \geq r_i \quad , \forall i \in \mathscr{I} \tag{2.37}$$

Lastly, we must constrain the dual variables :

— By definition, if the i^{th} primal constraint is an equality, then the i^{th} dual variable has no restriction of sign.

— The bound on the primal variables are applied in the same way to the dual ones, otherwise

thus giving us :

$$\begin{cases} \pi \geq 0 \\ \pi_0 \in \mathbb{R} \end{cases} \tag{2.38}$$

Finally we get a complete Dual linear integer programming from the Primal relaxation we have found :

$$\text{(AP - LD)} \quad \min_{\pi \geq 0, \pi_0 \in \mathbb{R}} \quad \pi_0 \tag{2.39}$$

$$\text{s.t.} \quad \pi_0 - \sum_{i=1}^{n} \pi_i e^{\mu_i} \geq r_0 \tag{2.40}$$

$$\pi_0 + \pi_i \geq r_i \qquad , \forall i \in \mathscr{I} \tag{2.41}$$

2.3.2 Relationships between Primal and Dual

In the following section, we explain the relationships between the dual variables and the primal constraints, and primal variables and dual constraints.
We can formulate a linear problem as :

$$\begin{array}{lrcl} min & c^T x & & \\ st & Ax & \geq & b \\ & x & \geq & 0 \end{array} \tag{2.42}$$

where :

— x is the vector of decision variables.

— c is the vector of objective function coefficients.

— A is the matrix of constraint coefficients.

— b is the vector of constant terms of constraints.

For this formulation, the associated dual problem with y the dual variables would be :

$$\begin{array}{lrcl} max & b^T y & & \\ st & A^T y & \leq & c \\ & y & \geq & 0 \end{array} \tag{2.43}$$

where :

— If the primal is a minimization, the dual will be a maximization.

— Inequality constraints in the primal $Ax \geq b$ turn into inequality constraints in the dual $A^T y \leq c$.

— The variables of the primal x become constraints in the dual, and the constraints of the primal become the variables of the dual y.

— The coefficients of the primal constraints become the coefficients of the objective function of the dual.

Following these rules we have for the (AP-L) :

— Primal variable y_0 corresponds to the dual constraint $\pi_0 - \sum_{i=1}^{n} \pi_i e^{\mu_i} \geq r_0$.

— Primal variable y_i corresponds to the dual constraint $\pi_0 + \pi_i \geq r_i$.

— Dual variable π_0 corresponds to the primal constraint $y_0 + \sum_{i=1}^{n} y_i = 1$.

— Dual variable π_i corresponds to the primal constraint $y_i \leq y_0 e^{\mu_i}$.

3

Ideal Formulation and a Greedy Algorithm

This section will focus on showing that the Assortment Planning Problem is solvable in polynomial time and in finding an algorithm for it.

3.1 Polynomial Time Solvable

Hereafter, is the reasoning to demonstrate that the AP Problem is solvable in polynomial time through the Primal Linear Relaxation previously found (**2.28,2.29,2.30**). We will then consider the following solutions, for any $k \in \mathscr{I}$:

$$y_0^k := \frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}} \tag{3.1}$$

$$y_i^k := \begin{cases} \frac{e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases} \quad , \forall i \in \mathscr{I} \tag{3.2}$$

3.1.1 Solution Feasibility for the Primal Linear Relaxation

First, we will prove that the considered solutions are feasible for the (AP-L) model.
(y^k, y_0^k) is feasible if it satisfies the constraints (**2.29**) and (**2.30**) of the (AP-L), we have :

$$y_0^k + \sum_{i=1}^{n} y_i^k = 1 \tag{3.3}$$

$$y_i^k \leq y_0^k e^{\mu_i} \quad , \forall i \in \mathscr{I} \tag{3.4}$$

by replacing by **3.1** and **3.2** we obtain :

$$\frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}} + \sum_{i=1}^{k} \frac{e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} + \sum_{i=k+1}^{n} 0 = 1$$

$$\Leftrightarrow \frac{1+\sum_{i=1}^{k} e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} = 1 \tag{3.5}$$

and,

$$\frac{e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} \leq \frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}} e^{\mu_i} \quad , \forall i \in [1,\ k] \tag{3.6}$$

$$0 \leq \frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}} e^{\mu_i} \quad , \forall i \in [k+1,\ n] \tag{3.7}$$

The constraints **3.5**, **3.6** and **3.7** are satisfied.

The objective value of the solution (y^k, y_0^k) is written :

$$r_0 y_0 + \sum_{i=1}^{k} r_i y_i \quad \text{or} \quad \frac{r_0}{1+\sum_{j=1}^{k} e^{\mu_j}} + \sum_{i=1}^{k} \frac{r_i e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} = \frac{r_0 + \sum_{i=1}^{k} r_i e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} \tag{3.8}$$

The solutions consider the first k elements, meaning it takes into account the products having a net revenue higher or equal to r_k as the products are ordered in decreasingly in term of net revenue.

3.1.2 Introduction of the decision variable

The Assortment Planning Problem is about selecting the products to be put in the set of alternatives **S** proposed to the customer. Furthermore, the considered solutions involve only considering the k first products from $\mathscr{I}$ which is the same as considering the potential inclusion or exclusion of those products. We can then consider the same change of variable as in the linearization of the Integer Program (**Section 2.2.1**) :

$$y_0 := \frac{1}{1+\sum_{j=1}^{k} x_j e^{\mu_j}} \tag{3.9}$$

$$y_i := \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{k} x_j e^{\mu_j}} \quad \forall i \in \mathscr{I} \tag{3.10}$$

We can show that x^k is an integer :

For $i \leq k$:

$$y_i = \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}}$$

Since $y_0^k = \frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}} \Leftrightarrow 1+\sum_{j=1}^{k} e^{\mu_j} = \frac{1}{y_0^k}$, we write :

$$y_i^k = \frac{e^{\mu_i}}{\frac{1}{y_0^k}} = e^{\mu_i} y_0^k$$

then for y_i,

$$\frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} = e^{\mu_i} y_0^k$$

$$\Leftrightarrow \frac{x_i e^{\mu_i}}{\frac{1}{y_0^k}} = e^{\mu_i} y_0^k \quad \text{(for } i \leq k\text{)}$$

$$\Leftrightarrow x_i e^{\mu_i} y_0^k = e^{\mu_i} y_0^k$$

$$\Leftrightarrow x_i = 1$$

For $i > k$:

$$y_i^k = 0$$

$$\Leftrightarrow \frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}} = 0$$

Since $e^{\mu_i} \neq 0$ for all i, for $\frac{x_i e^{\mu_i}}{1+\sum_{j=1}^{n} x_j e^{\mu_j}}$ to equal 0, we need the numerator to be equal to 0. This force x_i to be 0. We can conclude :

$$x_i^k := \begin{cases} 1 & \text{if } i \leq k \\ 0 & \text{if } i > k \end{cases}$$

The variable x^k is an integer and indicates that we consider the first k products to be taken from $\mathscr{I}$ to **S**. As stated in the **Section 2.1.1**, the integer variables x, are binary decision variables representing the fact we include or exclude the product x_i in the set of alternatives **S**.

3.1.3 Quality of the Considered Solutions

Now we want to show that (y^k, y_0^k) is a strictly better solution than (y^{k-1}, y_0^{k-1}) if and only if :

$$r_k > \frac{r_0 + \sum_{j=1}^{k-1} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.11}$$

Suppose (y^k, y_0^k) is a strictly better solution than (y^{k-1}, y_0^{k-1}) then :

$$\frac{r_0 + \sum_{i=1}^{k} r_i e^{\mu_i}}{1 + \sum_{j=1}^{k} e^{\mu_j}} > \frac{r_0 + \sum_{j=1}^{k-1} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.12}$$

$$\Leftrightarrow r_0 + \sum_{i=1}^{k} r_i e^{\mu_i} > \left(1 + \sum_{j=1}^{k} e^{\mu_j}\right) \frac{r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i}}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.13}$$

$$\Leftrightarrow r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i} + r_k e^{\mu_k} > \left(1 + \sum_{j=1}^{k-1} e^{\mu_j} + e^{\mu_k}\right) \frac{r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i}}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.14}$$

$$\Leftrightarrow r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i} + r_k e^{\mu_k} > r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i} + e^{\mu_k} \frac{r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i}}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.15}$$

$$\Leftrightarrow r_k e^{\mu_k} > e^{\mu_k} \frac{r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i}}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.16}$$

$$\Leftrightarrow r_k > \frac{r_0 + \sum_{i=1}^{k-1} r_i e^{\mu_i}}{1 + \sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.17}$$

3.1.4 Additional Proof on the Value of k

Here we want to show that there is no $k \in \mathscr{I}$ such that :

$$r_k \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} \quad \text{and} \quad r_{k+1} > \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}}$$

Suppose that both inequalities hold, we have :

$$r_k \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} \quad \text{and} \quad r_{k+1} > \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} \tag{3.18}$$

we can make appear the terms of the first inequality in the second inequality :

$$r_{k+1} > \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} = \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j + e^{\mu_k} r_k}{1+\sum_{j=1}^{k-1} e^{\mu_j} + e^{\mu_k}} \tag{3.19}$$

By using the definition of the first inequality we then have :

$$r_{k+1} > \frac{\frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}(1+\sum_{j=1}^{k-1} e^{\mu_j}) + e^{\mu_k} r_k}{1+\sum_{j=1}^{k-1} e^{\mu_j} + e^{\mu_k}} \tag{3.20}$$

since $r_k \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}$, if we replace r_k :

$$\frac{\frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}(1+\sum_{j=1}^{k-1} e^{\mu_j}) + \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} e^{\mu_k}}{1+\sum_{j=1}^{k-1} e^{\mu_j} + e^{\mu_k}} \tag{3.21}$$

$$\Leftrightarrow \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.22}$$

We obtain that for $r_k = \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}$, $\frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} = \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}}$,
but we know from the first condition that $r_k \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}$, then :

$$\frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.23}$$

The second inequality indicates $r_{k+1} > \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}}$, we just showed that $\frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}$, then :

$$r_{k+1} > \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}} \tag{3.24}$$

However, we first supposed that $r_k \leq \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}$, we find a contradiction. Thus, it is not possible for both inequalities to hold at the same time.

The function $k \rightarrow r^T y^k$ take the form of a step function where the addition of each product represents an increment in the value of the objective until a maximum is reached for the optimal k.

3.1.5 Optimal k^*

In this section, we will prove that there is only one optimal k^* such that :

$$r_1 > r_2 > \cdots > r_{k^*} > \frac{r_0+\sum_{j=1}^{k^*-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k^*-1} e^{\mu_j}} \quad \text{and} \quad \frac{r_0+\sum_{j=1}^{k^*} e^{\mu_j} r_j}{1+\sum_{j=1}^{k^*} e^{\mu_j}} \geq r_{k^*+1} > r_{k^*+2} > \cdots > r_n$$

We know $r_1 > r_2 > \cdots > r_n$ which implies $r_{i+1} < r_i,\ \forall i$.

We define k^* as the largest k such that the constraint $r_k > \frac{r_0+\sum_{j=1}^{k-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k-1} e^{\mu_j}}$ is satisfied (meaning the constraint is not satisfied for r_{k+1}).

With this definition, we can write the first condition on k^* :

$$r_{k^*} > \frac{r_0 + \sum_{j=1}^{k^*-1} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*-1} e^{\mu_j}} \tag{3.25}$$

The second condition on $k+1$:

$$\frac{r_0 + \sum_{j=1}^{k^*} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*} e^{\mu_j}} \geq r_{k^*+1} \tag{3.26}$$

If we have k^*, the largest value such that these constraints are satisfied. This ensure the uniqueness of $k*$ because we have seen in the previous point that the conditions ($r_{k^*} > \frac{r_0+\sum_{j=1}^{k^*-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k^*-1} e^{\mu_j}}$) and ($\frac{r_0+\sum_{j=1}^{k^*} e^{\mu_j} r_j}{1+\sum_{j=1}^{k^*} e^{\mu_j}} \geq r_{k^*+1}$) cannot hold for another k simultaneously.

We conclude there is exactly one k^* such that (**3.25**) and (**3.26**) are satisfied.

3.1.6 Quality of Optimal k^*

Hereafter, we show that we cannot have $r_{k^*} < \frac{r_0+\sum_{j=1}^{k^*} e^{\mu_j} r_j}{1+\sum_{j=1}^{k^*} e^{\mu_j}}$ and deduce that we have :

$$r_1 > \cdots > r_{k^*} \geq \frac{r_0 + \sum_{j=1}^{k^*} r_j e^{\mu_j}}{1 + \sum_{j=1}^{k^*} e^{\mu_j}} \geq r_{k^*+1} > \cdots > r_n \tag{3.27}$$

We will be using a reasoning by contradiction, assume we have :

$$r_{k^*} < \frac{r_0 + \sum_{j=1}^{k^*} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*} e^{\mu_j}} \tag{3.28}$$

then,

$$\Leftrightarrow r_{k^*} \left(1 + \sum_{j=1}^{k^*} e^{\mu_j}\right) < r_0 + \sum_{j=1}^{k^*} e^{\mu_j} r_j \tag{3.29}$$

$$\Leftrightarrow r_{k^*} \left(1 + \sum_{j=1}^{k^*} e^{\mu_j}\right) < r_0 + \sum_{j=1}^{k^*-1} e^{\mu_j} r_j + e^{\mu_{k^*}} r_{k^*} \tag{3.30}$$

$$\Leftrightarrow r_{k^*} \left(1 + \sum_{j=1}^{k^*} e^{\mu_j}\right) - e^{\mu_{k^*}} r_{k^*} < r_0 + \sum_{j=1}^{k^*-1} e^{\mu_j} r_j \tag{3.31}$$

$$\Leftrightarrow r_{k^*} \left(1 + \sum_{j=1}^{k^*-1} e^{\mu_j}\right) < r_0 + \sum_{j=1}^{k^*-1} e^{\mu_j} r_j \tag{3.32}$$

$$\Leftrightarrow r_{k^*} < \frac{r_0 + \sum_{j=1}^{k^*-1} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*-1} e^{\mu_j}} \tag{3.33}$$

We have a contradiction because by definition k^* is the largest k such that $r_{k^*} > \frac{r_0+\sum_{j=1}^{k^*-1} e^{\mu_j} r_j}{1+\sum_{j=1}^{k^*-1} e^{\mu_j}}$. We can conclude that the assumption (**3.28**) is false.

Then from the previous point we know that for $k = 1, \dots, k^*$ we have :

$$r_1 > \cdots > r_{k^*} \geq \frac{r_0 + \sum_{j=1}^{k^*} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*} e^{\mu_j}} \tag{3.34}$$

And that for $k = k^* + 1, \dots, n$ we have :

$$r_{k^*} \geq \frac{r_0 + \sum_{j=1}^{k^*} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*} e^{\mu_j}} \geq r_{k^*+1} > \cdots > r_n \tag{3.35}$$

And since (**3.28**) is false, we can conclude :

$$r_1 > \cdots > r_{k^*} \geq \frac{r_0 + \sum_{j=1}^{k^*} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k^*} e^{\mu_j}} \geq r_{k^*+1} > \cdots > r_n \tag{3.36}$$

3.2 Polynomial Time Algorithm

In the following section, we will propose a polynomial time algorithm solving the AP. To do so we first study the Dual model. In the same manner we worked on the Primal, we consider the following possible solutions, for $k \in \mathscr{I}$:

$$\pi_0^k := \frac{r_0 + \sum_{j=1}^{k} e^{\mu_j} r_j}{1 + \sum_{j=1}^{k} e^{\mu_j}} \tag{3.37}$$

$$\pi_i^k := \begin{cases} r_i - \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases}, \ \forall i \in \mathscr{I} \tag{3.38}$$

3.2.1 Solution Feasibility for the Dual Linear Relaxation

The objective value is defined by :

$$\pi_0^k r_0 + \sum_{i=1}^{k} \pi_i r_i \tag{3.39}$$

By substituting the value of π_i by π_i^k, we obtain :

$$\frac{r_0 + \sum_{j=1}^{k} r_j e^{\mu_j}}{1 + \sum_{j=1}^{k} e^{\mu_j}} r_0 + \sum_{i=1}^{k} \left(r_i - \frac{r_0 + \sum_{j=1}^{k} r_j e^{\mu_j}}{1 + \sum_{j=1}^{k} e^{\mu_j}} \right) r_i \tag{3.40}$$

This solution (π_k, π_0^k) is feasible for AP-LD if and only if it satisfies the constraints of the problem :

$$\pi_0 - \sum_{i=1}^{n} \pi_i e^{\mu_i} \geq r_0 \tag{3.41}$$

$$\pi_0 + \pi_i \geq r_i \quad \forall i \in \mathscr{I} \tag{3.42}$$

Consider the first constraint :

$$\begin{aligned}
&\pi_0 - \textstyle\sum_{i=1}^{n} \pi_i e^{\mu_i} \geq r_0 \\
\Leftrightarrow\ &\tfrac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} - \textstyle\sum_{i=1}^{k} \left(r_i - \tfrac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} \right) e^{\mu_i} \geq r_0 \\
\Leftrightarrow\ &\tfrac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} - \textstyle\sum_{i=1}^{k} r_i e^{\mu_i} - \left(\tfrac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} \right) \sum_{i=1}^{k} e^{\mu_i} \geq r_0 \\
\Leftrightarrow\ &r_0 + \tfrac{\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} - \textstyle\sum_{i=1}^{k} r_i e^{\mu_i} + \tfrac{\sum_{j=1}^{k} e^{\mu_j} r_j \sum_{i=1}^{k} e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} \geq r_0 \\
\Leftrightarrow\ &r_0 + \tfrac{\sum_{j=1}^{k} e^{\mu_j} r_j (1+\sum_{i=1}^{k} e^{\mu_i}) - (1+\sum_{j=1}^{k} e^{\mu_j}) \sum_{i=1}^{k} r_i e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} \geq r_0 \\
\Leftrightarrow\ &r_0 \geq r_0
\end{aligned}$$

The first constraint is satisfied, now consider the second constraint :

For $i \leq k$:

$$\begin{aligned}
&\pi_0^k + \pi_i^k \geq r_i \\
\Leftrightarrow\ &r_0 + \tfrac{\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} + r_i - \tfrac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}} \geq r_i \\
\Leftrightarrow\ &r_0 + r_i - \tfrac{r_0}{1+\sum_{j=1}^{k} e^{\mu_j}} \geq r_i \\
\Leftrightarrow\ &r_0 (1 - \tfrac{1}{1+\sum_{j=1}^{k} e^{\mu_j}}) + r_i \geq r_i
\end{aligned}$$

is satisfied because $\frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}} \leq 1$ and $(1 - \frac{1}{1+\sum_{j=1}^{k} e^{\mu_j}}) \geq 0$.

For $i > k$:

$$\begin{aligned}
\pi_0^k + \pi_i^k &\geq r_i \\
\pi_0^k &\geq r_i
\end{aligned}$$

The second constraint is satisfied if and only if $\pi_0^k \geq r_i$.

We conclude that if $\pi_0^k \geq r_i$ for $i \leq k$, then there exists a k such that the solution (π^k, π_0^k) is feasible.

3.2.2 Optimality of k

We want now to prove that the set of alternatives composed of the k first products $S = \{1, \ldots, k\}$ is optimal for AP, given that $k \in \mathscr{I}$, if (π^k, π_0^k) is feasible for AP-LD. We will also determine if AP-L is an ideal formulation for AP.

Assume (π^k, π_0^k) is a feasible solution for AP-LD, i.e., it satisfies :

$$\pi_0^k - \sum_{i=1}^{n} \pi_i^k e^{\mu_i} \geq r_0 \tag{3.43}$$

$$\pi_0^k + \pi_i^k \geq r_i, \quad \forall i \in \mathscr{I} \tag{3.44}$$

Let us show that $S = \{1, \ldots, k\}$ is an optimal solution for the primal AP problem. We can use the relationship between primal an dual variable and obtain the associated primal variable :

$$y_0^k = \frac{1}{1 + \sum_{j=1}^{k} e^{\mu_j}} \tag{3.45}$$

$$y_i^k = \begin{cases} \frac{e^{\mu_i}}{1+\sum_{j=1}^{k} e^{\mu_j}} & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{3.46}$$

Two feasible solution for a primal and a dual problem are optimal if and only if they satisfied the complementary slackness conditions, we need to show :

$$y_i(\pi_0 - \pi_i - r_i e^{\mu_i}) = 0, \quad \forall i \in \mathscr{I} \tag{3.47}$$

$$y_0(\pi_0 - r_0) = 0 \tag{3.48}$$

For $i \leq k$

$$y_i^k = \frac{e^{\mu_i}}{1 + \sum_{j=1}^{k} e^{\mu_j}} \tag{3.49}$$

$$(\pi_0^k - \pi_i^k - r_i e^{\mu_i}) = 0 \tag{3.50}$$

Since $y_i^k > 0$ for $i \leq k$, we have :

$$\pi_0^k - \pi_i^k = r_i e^{\mu_i} \tag{3.51}$$

For $i > k$, $y_i^k = 0$. It is trivial that $y_i^k(\pi_0 - \pi_i - r_i e^{\mu_i}) = 0$

For y_0 :

$$y_0^k = \frac{1}{1 + \sum_{j=1}^{k} e^{\mu_j}} \tag{3.52}$$

$$(\pi_0^k - r_0) = 0 \tag{3.53}$$

Since $y_0^k > 0$, we have that :

$$\pi_0^k = r_0 \tag{3.54}$$

The complementary slackness conditions are satisfied for $S = \{1, \ldots, k\}$, it is an optimal solution for the primal AP. We conclude that the feasibility and complementary slackness conditions implies that solving the linear relaxation directly provides the same optimal integer solution as for the primal problem. Thus, AP-L is an ideal formulation for AP.

3.2.3 Polynomial Time Algorithm

Hereafter, we propose a polynomial time algorithm that solves AP, while estimating its worst-time complexity.

Algorithm 1

Initialisation :
$\mathscr{I} = \{1, 2, \ldots, n\}$
$S_{\text{opt}} = \emptyset$
$\text{value}_{\text{opt}} = 0$
Algorithm :
for $k \in \mathscr{I}$ **do**
  $\pi_0^k \Leftarrow r_0 + \frac{\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}}$ ▷ Compute π value
  $\pi_i^k \Leftarrow r_i - \frac{r_0+\sum_{j=1}^{k} e^{\mu_j} r_j}{1+\sum_{j=1}^{k} e^{\mu_j}}$ for $i \leq k$
  if $\pi_0^k \geq r_i - \pi_i^k e^{\mu_i}$ **and** $\pi_0^k - \sum_{i=1}^{k} \pi_i^k e^{\mu_i} \geq r_0$ **then** ▷ Check feasibility constraints
    $\text{value} \Leftarrow r_0 + \sum_{i=1}^{k} e^{\mu_i} r_i$
    if $\text{value}_{\text{opt}} < \text{value}$ **then** ▷ Update best solution
      $\text{value}_{\text{opt}} \Leftarrow \text{value}$
      $\text{S}_{\text{opt}} \Leftarrow \{1, 2, \ldots, k\}$
    end if
  end if
end for
return S_{opt}

The complexity of this algorithm can be evaluated :

— The initialisation is constant : $O(1)$

— The for loop is execute n times :

  — The computation of π_0^k and π_i^k are in $O(k)$

  — The verification of the constraints is in $O(k)$

  — The computation of the optimal value is in $O(k)$

  The for loop is in $O(n^2)$

The algorithm has a time complexity of $O(n^2)$ which is polynomial.

4 A More Practical Model

In this section, we will consider a more practical case where the retailer can only offer up to p products to its customers. We will then check if the previously found algorithm still works with the new model we found and finally we will propose a Lagrangean Relaxation to solve the new model.

4.1 Mixed Integer Linear Program Model

We then need to first adapt our Linear Integer Program (AP-L) by adding a new variable and constraint in order to produce the new model, which will be a Mixed Integer Linear Program formulation :

$$\text{(AP - L)} \quad \begin{cases} \max_{y, y_0 \geq 0} & r_0 y_0 + \sum_{i=1}^{n} r_i y_i \\ \text{s.t.} & y_0 + \sum_{i=1}^{n} y_i = 1 \\ & y_i \leq y_0 e^{\mu_i} \quad , \forall i \in \mathscr{I} \end{cases} \tag{4.1}$$

4.1.1 Capacity Constraint

Adding a maximal amount of product the retailer can offer to its customers is equivalent to adding a capacity constraint to the problem where p is the upper bound, for this we will introduce a new binary variable representing whether or not the retailer offers the product i to its customers :

$$Z = \{z_1, ..., z_n\} \quad \text{with} \quad z_i = \begin{cases} 1 & \text{product i is offered} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathscr{I} \tag{4.2}$$

Thus by adding this new variable, we can add the following constraint :

$$\sum_{i=1}^{n} z_i \leq p \tag{4.3}$$

This constraint represent the fact that the sum of all products being offered cannot be larger than the maximum amount of products the retailer can offer. We can also make a new link between the decision vector z and the purchase probability vector y, if the product i is not proposed meaning that $z_i = 0$, therefore the

probability of being purchased y_i is 0. But if the product i is proposed meaning $z_i = 1$, then the probability of it being chosen is at most 1, thus we can introduce another constraint :

$$y \leq z \Leftrightarrow y_i \leq z_i \quad , \forall i \in \mathscr{I} \tag{4.4}$$

this constraint is complete and valid as the values for each y_i is lower bounded by zero, see **4.1**. Plus the advantage of this constraint is that it ensures that y_i contributes to the objective function when z_i and prevent the model from allocating non-zero y_i to products not chosen in the assortment.

4.1.2 APC-MILP

Summing up we finally get the new model :

$$\text{(APC-MILP)} \quad \max_{y, y_0 \geq 0, z \in \{0,1\}} \quad r_0 y_0 + \sum_{i=1}^{n} r_i y_i \tag{4.5}$$

$$\text{s.t.} \quad y_0 + \sum_{i=1}^{n} y_i = 1 \tag{4.6}$$

$$y_i \leq y_0 e^{\mu_i} \quad , \forall i \in \mathscr{I} \tag{4.7}$$

$$y \leq z \tag{4.8}$$

$$\sum_{i=1}^{n} z_i \leq p \tag{4.9}$$

In term of validity, the objective function stays the same as the objective is to maximize the expected revenue from the proposed products, only a binary variable is added (i.e. z_i). This new variable allows to add two new constraints : one on the purchase probability (**4.8**) ensuring that the objective function only consider the offered products and one on the capacity (**4.9**) ensuring we cannot propose more that the retailer can propose.

4.2 Study of the Maximum Amount of Product

In this section, we study the solvability of the new problem with the previous algorithm, we also propose a new algorithm for the actualized problem while comparing different values for the new parameter.

4.2.1 APC-MILP implementation

To determine the applicability of the previous algorithm to the APC-MILP problem, we need to analyse in detail the differences and similarities between the constraints and steps of the algorithm and those of the APC-MILP formulation.

Binary variables and their limits :

We begin by analysing the binary variables and their limits :

* for the APC-MILP formulation, we know that $y \leq z$ and $\sum_{i=1}^{n} z_i \leq p$
* The previous algorithm does not directly manipulate binary variables z. It focuses only on selecting an optimal subset of the products S_{opt}.

Analysis :

— The constraint $y \leq z$ in **4.9** means that the variables y_i can only be positive if the corresponding binary variables z_i are equal to 1, which is not explicitly taken into account in the previous algorithm.

— In addition, the constraint $\sum_{i=1}^{n} z_i \leq p$ imposes an upper limit on the number of products selected, which could be approximated by the maximum number of products k in the previous algorithm, but this requires explicit verification of the binary constraints.

Normalisation :

* The APC-MILP formulation, we have $y_0 + \sum_{i=1}^{n} y_i = 1$
* The previous algorithm calculates intermediate values π and values based on the product parameters , but it does not explicitly guarantee that the sum of y_0 and y_i is equal to 1.

Analysis :

— The conservation of the proportions of the variables y in **4.9** guarantees that the sum of all the product shares and the non-consumption share y_0 is equal to 1, thus representing a valid probability distribution.

— The previous algorithm, although it computes intermediate values to determine the best product selection, does not ensure that these values are normalised to satisfy this constraint.

Use of average utilities :

* The formulation **4.5** has the constraint $y_i \leq y_0 e^{\mu_i} \quad , \forall i \in \mathscr{I}$.
* The previous algorithm computes the π values using the average μ_i utilities and integrates them into the feasibility conditions for each i product.

Analysis :

— The constraint $y_i \leq y_0 e^{\mu_i}$ in APC-MILP imposes a relationship between the product shares y_i and the non-consumption share y_0, modulated by the exponential utilities of the products.

— The previous algorithm integrates the average utilities μ_i into the π calculations, which is consistent with the idea of modulating product shares by their utilities. However, the algorithm does not explicitly formulate this constraint in a linear manner as in APC-MILP.

Conclusion

In conclusion, although the previous algorithm incorporates some of the logic and structures of the APC-MILP formulation, it does not explicitly take into account all the essential constraints, in particular those related to the binary z variables and the normalisation of the y variables. Thus, the previous algorithm is not fully suitable for solving the APC-MILP problem.

4.2.2 Collected results

The main program execute the APC-MILP formulation for $p \in \{1, \frac{n}{5}, \frac{n}{2}, n\}$.

Experiments performed

In this analysis, we take $n = 10$, which means that the retailer can offer up to 10 different products.
We assume that z is the objective value.

— when $p = 1$, $z = 0.6149$

— when $p = \frac{n}{5}$, $z = 0.679$

— when $p = \frac{n}{2}$, $z = 0.704$

— when $p = n$, $z = 0.704$

In conclusion, we can state that :

— For $p = 1$: The expected optimal revenue is the lowest, $z = 0.6149$, which is logical since we only have one product available to customers.

— For $p = \frac{n}{5}$: The optimal expected revenue increases to $z = 0.679$, showing that adding more products improves revenue by offering customers more choice.

— For $p = \frac{n}{2}$: The optimal expected revenue keeps increasing and reaching $z = 0.704$. This indicates that diversifying the assortment to half of the available products almost maximises the expected revenue.

— For $p = n$: The expected optimal revenue remains stable at z =0.704. This suggests that beyond $\frac{n}{2}$ products, adding more products to the assortment no longer increases expected revenue.

To conclude, increasing the size of the assortment of available products (value of p) also increases the expected revenue up to a certain threshold. Once p value equals half the total number of products ($p = \frac{n}{2}$), the optimal expected revenue stabilises. This suggests that offering all products to maximise revenue is not necessary. This analysis shows that diversifying the assortment is beneficial, but that there is a saturation point beyond which adding more products no longer brings any additional benefits.

4.3 Lagrangian Relaxation

In this section we propose a Lagrangian relaxation on an adapted version of the previous Integer Programming problem (AP-IP) by adding the capacity constraint (i.e. adding the maximum amount of available products p). The problem then becomes :

$$\text{(APC-IP)} \quad \max_{x\in\{0,1\}^n} \frac{r_0 + \sum_{i=1}^n x_i r_i e^{\mu_i}}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \tag{4.10}$$

$$\text{s.t.} \quad \sum_{i=1}^n x_i \leq p \tag{4.11}$$

4.3.1 Pricing Problem

Before doing the Lagrangian Relaxation, we can notice that the constraint **4.11** is equivalent to :

$$\frac{\sum_{i=1}^n x_i}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \leq \frac{p}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \tag{4.12}$$

We can then proceed to the Lagrangean Relaxation, the constraint **4.11** complicate significantly the new problem, we will then add it to the objective function with a non-negative penalty term, $\lambda \geq 0$.
First we rewrite the constraint to make further computations easier :

$$[\lambda] \quad \frac{\sum_{i=1}^n x_i}{1 + \sum_{i=1}^n x_i e^{\mu_i}} - \frac{p}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \leq 0 \tag{4.13}$$

The relaxation aims to find an upper bound to the objective value as this is a maximization problem, by penalizing it therefore the penalization member is to be subtracted from the objective function :

$$\begin{aligned} &\frac{r_0 + \sum_{i=1}^n x_i r_i e^{\mu_i}}{1 + \sum_{i=1}^n x_i e^{\mu_i}} - \lambda\left(\frac{\sum_{i=1}^n x_i}{1 + \sum_{i=1}^n x_i e^{\mu_i}} - \frac{p}{1 + \sum_{i=1}^n x_i e^{\mu_i}}\right) \\ \Leftrightarrow &\frac{r0}{1 + \sum_{i=1}^n x_i e^{\mu_i}} + \frac{\sum_{i=1}^n x_i r_i e^{\mu_i}}{1 + \sum_{i=1}^n x_i e^{\mu_i}} - \frac{\lambda(\sum_{i=1}^n x_i)}{1 + \sum_{i=1}^n x_i e^{\mu_i}} + \frac{\lambda p}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \end{aligned} \tag{4.14}$$

As both fractions in the center of the expression have the same denominator and are both iterating on the same values (i.e. $\sum_{i=1}^n x_i$) we can assemble them. The first and last terms both have the same denominator they can be put on the same denominator.

$$\Leftrightarrow \frac{r_0 + \lambda p}{1 + \sum_{i=1}^n x_i e^{\mu_i}} + \frac{\sum_{i=1}^n (r_i e^{\mu_i} - \lambda) x_i}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \tag{4.15}$$

By using the same change variable as presented in the **Section 2.2.1** (i.e. **2.7**, **2.8**) and by using a mathematical trick we can further simplify the expression :

$$\Leftrightarrow \frac{r_0 + \lambda p}{1 + \sum_{i=1}^n x_i e^{\mu_i}} + \frac{\sum_{i=1}^n (r_i e^{\mu_i} - \lambda) x_i}{1 + \sum_{i=1}^n x_i e^{\mu_i}} \times \frac{x_i e^{\mu_i}}{x_i e^{\mu_i}} \tag{4.16}$$

$$\Leftrightarrow (r_0 + \lambda p) y_0 + \sum_{i=1}^{n} \frac{r_i e^{\mu_i} - \lambda}{e^{\mu_i}} y_i \tag{4.17}$$

Due to the change of variables we must set new constraints. By using the same development as in the section **2.2.2**, we can recast the pricing problem with a penalization $\lambda \geq 0$ in the following way :

$$\text{(AP-L)}(\lambda) \quad \omega(\lambda) := \max_{y, y_0 \geq 0} \quad r_0(\lambda) y_0 + \sum_{i=1}^{n} r_i(\lambda) y_i = (r_0 + \lambda p) y_0 + \sum_{i=1}^{n} \frac{r_i e^{\mu_i} - \lambda}{e^{\mu_i}} y_i \tag{4.18}$$

$$\text{s.t.} \quad y_0 + \sum_{i=1}^{n} y_i = 1 \tag{4.19}$$

$$y_i \leq y_0 e^{\mu_i}, \quad \forall i \in \mathscr{I} \tag{4.20}$$

$$\text{Where :} \quad \begin{cases} r_0(\lambda) = r_0 + \lambda p \\ r_i(\lambda) = \frac{r_i e^{\mu_i} - \lambda}{e^{\mu_i}} \end{cases} \tag{4.21}$$

4.3.2 Best Dual Bound

To solve the univariate problem of minimising $\omega(\lambda)$ for $\lambda \geq 0$ and find the best dual bound, we need to design a binary search. This approach is motivated by the convex and piecewise linear nature of the function $\lambda \to \omega(\lambda)$. Thus, we will develop the following binary search algorithm to determine the optimal value of λ^* :

Algorithm 2 Binary Search for λ^*

Initialisation :
$\lambda_{\min} = 0$
$\lambda_{\max} = \max\{0, \frac{r_1 - r_0}{p}\}$
tolerance = 10^{-4}
Algorithm :
while $(\lambda_{\max} - \lambda_{\min}) >$ tolerance **do**
  $\lambda_{\text{middle}} \leftarrow \frac{\lambda_{\min} + \lambda_{\max}}{2}$
  if $\omega(\lambda_{\text{middle}}) < \omega(\lambda_{\min})$ **then**
    $\lambda_{\min} \leftarrow \lambda_{\text{middle}}$
  else
    $\lambda_{\max} \leftarrow \lambda_{\text{middle}}$
  end if
end while
$\lambda^* \leftarrow \frac{\lambda_{\min} + \lambda_{\max}}{2}$
return λ^*

The complexity of this binary search algorithm is logarithmic as a function of the difference between r_1 and r_0, divided by p and ϵ. This complexity can be explained by the behaviour of the binary search, at each iteration the search space is cut in half. We have a complexity of log(n), but here we know that the search space is between 0 and $max\{0, \frac{r_1 - r_0}{p}\}$ so we obtain $O(\log_2(\frac{r_1 - r_0}{p}))$.

For show that λ^* cannot be greater than $\max\{0, \frac{r_1 - r_0}{p}\}$, we consider our function $\omega(\lambda)$, which we seek to minimise. This function is convex and piecewise linear. When λ is positive, $\omega(\lambda)$ decreases up to a point, then

starts to grow. This point of growth corresponds to the optimal value of λ, called λ^*. Now, if $\max\{0, \frac{r_1-r_0}{p}\}$ exceeds $\max\{0, \frac{r_1-r_0}{p}\}$, this means that $\omega(\lambda)$ will become increasing, which is not optimal for our problem in minimisation. Therefore, the optimal value of λ cannot exceed $\frac{r_1-r_0}{p}$.

4.3.3 Heuristic Building

In this section, we will develop an heuristic to examine each value of λ that provides a feasible solution to our practical problem. We will compare these solutions with the best found so far, in order to select the optimal one, in our case the smallest value for λ.

Algorithm 3 Heuristic for searching the best λ

function HEURISTIC(ϵ, r_0, r_1, p)
  Initialisation :
  $\lambda^* \leftarrow 0$
  $z^* \leftarrow \infty$
  $\lambda_{min} \leftarrow 0$
  $\lambda_{max} \leftarrow \max\{0, \frac{r_1-r_0}{p}\}$
  Algorithm :
  while $(\lambda_{max} - \lambda_{min}) > \epsilon$ **do**
    $\lambda_{middle} \leftarrow \frac{\lambda_{min}+\lambda_{max}}{2}$
    $z_{current} \leftarrow$ compute_the_objective_value($\lambda_{middle}, r_0, r_1, p$)
    if $(z_{current} < z^*)$ **then**
      $\lambda^* \leftarrow \lambda_{middle}$
      $z^* \leftarrow z_{current}$
    end if
    if $(z_{current} <$ compute_the_objective_value($\lambda_{min}, r_0, r_1, p$)) **then**
      $\lambda_{max} \leftarrow \lambda_{middle}$
    else
      $\lambda_{min} \leftarrow \lambda_{middle}$
    end if
  end while
  return λ^*, z^*
end function

4.3.4 Lagrangian Algorithm

The algorithm uses binary search to find the optimal $lambda$. Each iteration, the search interval is divided in half. The initial interval for $lambda$ is between 0 and $max\{0, \frac{r_1-r_0}{p}\}$. The number of iterations required to have a precision of ϵ is $O(\log_2(\frac{r_1-r_0}{\frac{p}{\epsilon}}))$ which gives us $O(\log_2(\frac{r_1-r_0}{p\epsilon}))$.

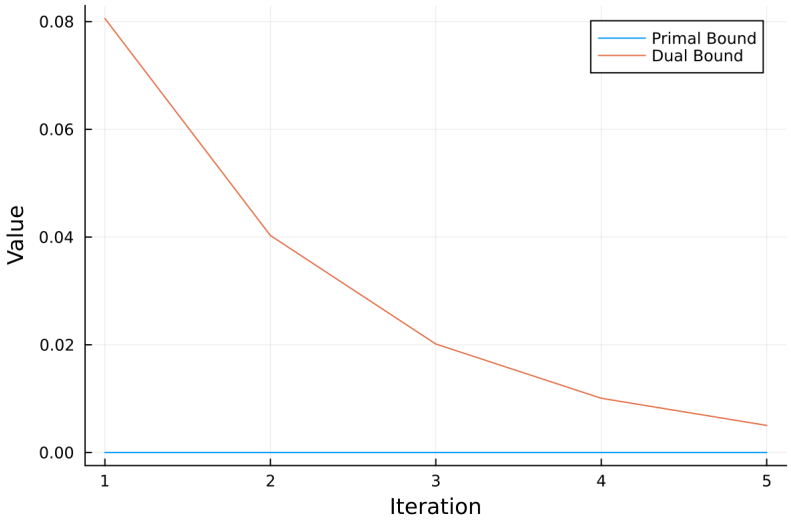

FIGURE 4.1: Illustration of primal and dual bounds on the *small* instance with $p = n/2$

5 Implementation and Evaluation

We analysed the different instances for each size and each model. The tests were performed on 100 runs with $p = \frac{n}{2}$. We calculated and reported the minimum, average and maximum values of the optimal solutions obtained, as well as the corresponding resolution times, in the following tables. These statistics provide a detailed overview of the performance of each model in terms of solution quality and time efficiency.

We define a maximum run time of 300 seconds.

	Average	*Minimum*	*Maximum*
small	$0.006767s$	$0.004554s$	$0.175702s$
medium	$0.123433s$	$0.108694s$	$0.315176s$

TABLE 5.1: Run Time results of the **APC_L** model on *small* and *medium* instances.

	Average	*Minimum*	*Maximum*
small	$0.008265s$	$0.004824s$	$0.260106s$
medium	$0.23053s$	$0.20924s$	$0.453266s$

TABLE 5.2: Run Time results of the **APC_MILP** model on *small* and *medium* instances.

	Average	*Minimum*	*Maximum*
small	0.003173	$1.3e-5$	0.3158
medium	*timeout*	*timeout*	*timeout*

TABLE 5.3: Run Time results of the **greedy** algorithm on *small* and *medium* instances

	Average	*Minimum*	*Maximum*
small	0.35969	0.104521	24.2524
medium	0.8219	0.768497	0.93793

TABLE 5.4: Run Time results of the **lagrangian** algorithm on *small* and *medium* instances

Conclusions :

— The **APC_L** model is faster than the **APC_MILP** model for *small* and *medium* instances, as shown by the lower average computation time for each instance size.

— Variations in computation time (difference between minimum and maximum time) are more pronounced for the **APC_MILP** model, particularly for *small* instances.

— By increasing the size of the instances from *small* to *medium*, the calculation times increase for both models, but this increase is more marked for the **APC_MILP** model.

— The **greedy** algorithm worked fine for the small instances, with an average time of 0.003173 seconds to find the optimal solution. However, for the medium instances, the algorithm exceeded the timeout limit at each execution, meaning that it was unable to complete the execution within the allotted time. This suggests that the greedy algorithm may not be suitable for medium instances due to its complexity or intrinsic characteristics.

— The **Lagrangian** algorithm appears to be more stable and reliable for medium size instances compared to small size instances, where execution times can vary considerably.

In conclusion, the choice of model or algorithm depends on the specific characteristics of the problem instances. For small to medium-sized instances where speed is crucial, the **APC-L** model or the **Lagrangian** algorithm may be preferred. However, for larger or more complex instances, alternative approaches may need to be explored to ensure efficient solution finding within reasonable time constraints.

	Average	*Minimum*	*Maximum*
small	0.63150	0.27159	0.82151
medium	0.9847	0.98243	0.9877

TABLE 5.5: Optimal Values for the **APC_L** model for the *small* and *medium* instances.

	Average	*Minimum*	*Maximum*
small	0.6313	0.27159	0.8186
medium	0.9847	0.9824	0.9877

TABLE 5.6: Optimal Values for the **APC_MILP** model for the *small* and *medium* instances.

	Average	*Minimum*	*Maximum*
small	3.3809	0.0	9.96205
medium	*timeout*	*timeout*	*timeout*

TABLE 5.7: Optimal Values for the **greedy** algorithm for the *small* and *medium* instances.

	Average	*Minimum*	*Maximum*
small	0.86877	0.86877	0.86877
medium	0.9933	0.9933	0.9933

TABLE 5.8: Optimal Values for the **lagrangian** algorithm for the *small* and *medium* instances.

Conclusions :

— The average optimal values obtained with the two models are very similar for the *small* and *medium* instances, indicating that the two models provide solutions of comparable quality.

— The variations in the optimal values for the *small* instances are slightly larger for the **APC_L** model (range from 0.27159 to 0.82151) compared to the **APC_MILP** model (range form 0.27159 to 0.8186).

— For the *medium* instances, the optimal values are almost identical on both models, with very narrow ranges of variation.

— The **greedy** algorithm succeeded in finding optimal solutions for the small instances, with an average time of 3.3809 seconds to calculate the optimal solution. However for medium instances, the algorithm exceeded the timeout limit for each execution, meaning that it was unable to complete the execution within the allotted time. This suggests that the greedy algorithm may not be suitable for medium instances due to its complexity or intrinsic characteristics.

— The **Lagrangean** algorithm appears to be very consistent (i.e. small variability between values) in the optimal solutions it provides, regardless of the instance size or input conditions. This means that the algorithm offers great reliability and consistency in its results thus showing its quality of resolution.

Variations in optimal values are slightly larger for the **APC_L** model in small instances compared to **APC_MILP**, where the differences are minimal. The optimal values are almost identical for the two models in the medium instances, demonstrating their consistency in the quality of the solutions. The gluttonous algorithm (**greedy**) shows promising performance for small instances, but systematically exceeds the time limit for medium instances, limiting its applicability to larger or more complex problems. In contrast, the Lagrangian algorithm (**Lagrangian**) offers remarkable consistency in producing optimal solutions, regardless of instance size or input conditions, making it ideal for applications where stability and consistent performance are essential.

For the model APC-MILP, the following results were obtained using the seed `Random.seed!(123)` :

— **Solution time** : 336.657021 seconds

— **Optimal value** : 0.9989

For the model AP-L, the following results were obtained using the seed `Random.seed!(123)` :

— **Solution time** : 332.049069 seconds

— **Optimal value** : 0.9989

6 Conclusion

This project allowed us to explore the Assortment Planning Problem (AP), focusing on how a retailer can maximize expected revenue by selecting a subset of products. Firstly, we translated the AP into an integer programming problem (AP-IP) and then reformulated it into a linear programming model (AP-IPL). This allowed us to derive a continuous relaxation (AP-L) and its dual (AP-LD). We then demonstrated that the AP can be solved in polynomial time using a greedy algorithm based on the continuous relaxation model. This algorithm efficiently identifies the optimal set of products by iteratively evaluating feasible solutions. We then extended the model to handle situations with a limit on the number of products. This led to the formulation of a Mixed Integer Linear Program (APC-MILP). We compared the results for various values of a number of products. Moreover, we used the Lagrangian Relaxation to approximate solutions for the practical model, transforming the problem into a penalized problem and search for an optimal penalty parameter with a binary search algorithm. Finally, we implemented and evaluated the models and algorithms on instances of different sizes. The choice of the best model or algorithm depends on several factors, including the quality of the solution, computational efficiency and the specific characteristics of the problem instances. Although the Lagrangian algorithm may be the optimal choice for its consistency and reliability in producing optimal solutions, the greedy algorithm may be a viable alternative for small problems where speed of solution is crucial, even if this involves a slight compromise on the quality of the solution.

Articles

[Vin] VINEET GOYAL Retsef Levi, Danny Segev (s. d.). « Near-Optimal Algorithms for the Assortment Planning Problem under Dynamic Substitution and Stochastic Demand ». In : (). URL : https://www.columbia.edu/~vg2277/Assortment_final.pdf.