An Exact Method for Assortment Optimization under the Nested Logit Model

Laurent Alfandari, Alborz Hassanzadeh, Ivana Ljubić

ESSEC BUSINESS SCHOOL

The pioneering spirit

An Exact Method for Assortment Optimization under the Nested Logit Model

Laurent Alfandari
Alborz Hassanzadeh
Ivana Ljubic

ESSEC RESEARCH CENTER

WORKING PAPER 2001

An Exact Method for Assortment Optimization under the Nested Logit Model

Laurent Alfandari

ESSEC Business School of Paris, Cergy-Pontoise, France, alfandari@essec.edu

Alborz Hassanzadeh

ESSEC Business School of Paris, Cergy-Pontoise, France, al.zadeh@essec.edu

Ivana Ljubić[1]

ESSEC Business School of Paris, Cergy-Pontoise, France, ivana.ljubic@essec.edu

Abstract

We study the problem of finding an optimal assortment of products maximizing the expected revenue, in which customer preferences are modeled using a Nested Logit choice model. This problem is known to be polynomially solvable in a specific case and NP-hard otherwise, with only approximation algorithms existing in the literature. We provide an exact general method that embeds a tailored Branch-and-Bound algorithm into a fractional programming framework. Contrary to the existing literature, in which assumptions are imposed on either the structure of nests or the combination and characteristics of products, no assumptions on the input data are imposed. Although our approach is not polynomial in the input size, it can solve the most general problem setting for large-size instances. We show that the fractional programming scheme's parameterized subproblem, a highly non-linear binary optimization problem, is decomposable by nests, which is the primary advantage of the approach. To solve the subproblem for each nest, we propose a two-stage approach. In the first stage, we fix a large set of variables based on the single-nest subproblem's newly-derived structural properties. This can significantly reduce the problem size. In the second stage, we design a tailored Branch-and-Bound algorithm with problem-specific upper bounds. Numerical results show that the approach is able to solve assortment instances with five nests and with up to 5,000 products per nest. The most challenging instances for our approach are those with a mix of nests' dissimilarity parameters, where some of them are smaller than one and others are greater than one.

Keywords: combinatorial optimization, revenue management, assortment optimization, fractional programming, nested logit.

[1]Corresponding author

1. Introduction

Assortment optimization is the problem of choosing a portfolio of products in a competitive environment to offer to customers to maximize expected revenue. This class of problems has important applications in retail, online advertising, or revenue management, see, e.g. [3, 19, 17]. The research of Talluri and Van Ryzin [31] was the first of its kind to demonstrate the importance of incorporating the choice behavior of customers when deciding which products to offer in an assortment. Their work contributes to the growing literature on discrete choice models. Such models have long been used to describe and understand how customers choose a product from an offered set of products that vary in different attributes such as price and quality. A popular way to model customer choices is to follow the principle of utility maximization. Based on this concept, a customer attributes a random utility to each product and selects the product with the largest utility. The Multinomial logit (MNL) model ([23, 26]) is one of the most popular choice models based on the utility maximization theory. The MNL choice model is based on the Independence of Irrelevant Alternatives property, which states that when individuals face two alternatives, the odds of choosing one over the other is not affected by the introduction of a third alternative. This property has often been contested, which motivated the creation of the general nested attraction model of which Williams [35] first introduced the well-known Nested Logit (NL), model.

In the NL model, customers first select a nest and then choose a product within that nest (each product appears in exactly one nest). Since the multinomial logit model suffers from the independence of irrelevant alternatives property, the nested logit model was developed to capture the degree of dissimilarity of products within a nest. This model has various applications. For example, consider a tourist who is using online search engines to book a hotel room. In this case, the nests are the available hotels to choose from, and a product is a room within each hotel. Naturally, the customer first selects a hotel and then selects a room from that hotel according to her preferences over the available options. For empirical work in this area, see, e.g., [20, 32, 33, 34, 36].

1.1. Problem definition

We first provide a formal definition of the *Assortment Optimization Problem under the Nested-Logit choice model* (AOPNL). Let $M = \{1, \dots, m\}$ be the set of all nests, and $N = \{1, \dots, n\}$ the index set of all products in a nest and in every nest, up to $|N|$ products can be offered. We assume that each product can appear in exactly one nest. We call $\mathcal{C}_i$ the collection of all feasible assortments for nest $i \in M$, and an assortment in nest i is denoted by $S_i \in \mathcal{C}_i$. We also denote by $r_{ik} \geqslant 0$ the revenue obtained by selling one unit of product $k \in S_i$ offered in nest i. Finally, customers' preference for product k in nest i is $v_{ik} \geqslant 0$.

Assuming a customer chooses nest i and assortment $S_i \in \mathcal{C}_i$ is offered for nest i, based on MNL choice probabilities, the probability for that customer to choose product k is:

$$P_{ik}(S_i) = \begin{cases} \dfrac{v_{ik}}{v_{i0} + \sum_{j \in S_i} v_{ij}} & k \in S_i \\ 0 & k \notin S_i \end{cases} \qquad i \in M, k \in N, \tag{1}$$

where $v_{i0} \geqslant 0$ is the attractiveness of leaving nest i without any purchase. Furthermore, the probability that a particular nest i is chosen given that assortment $(S_1, \ldots, S_m) \in \mathcal{C}_1 \times \ldots \times \mathcal{C}_m$ is offered is:

$$Q_i(S_1, \ldots, S_m) = \frac{(v_{i0} + \sum_{k \in S_i} v_{ik})^{\gamma_i}}{V_0 + \sum_{l \in M} (v_{l0} + \sum_{k \in S_l} v_{lk})^{\gamma_l}} \qquad i \in M,$$

where $V_0 \geqslant 0$ is the attractiveness of the option of leaving the system without picking any nest (in the first stage), and $\gamma_i \geqslant 0$ is the *dissimilarity* parameter of nest i. The dissimilarity parameter of a nest characterizes the degree of dissimilarity between the products within that nest. It can also be interpreted as the influence of a nest over others [11, 15]. If assortment S_i is offered in nest i and if this nest is selected, the expected revenue is:

$$R_i(S_i) = \sum_{k \in S_i} r_{ik} P_{ik}(S_i),$$

where $R_i(\emptyset) = 0$. Therefore, if we offer assortment $(S_1, \ldots, S_m)$ where $S_i \in \mathcal{C}_i$, for $i \in M$, the expected revenue we obtain over all nests is:

$$\Pi(S_1, \ldots, S_m) \quad = \quad \sum_{i \in M} Q_i(S_1, \ldots, S_m) \sum_{k \in S_i} r_{ik} P_{ik}(S_i).$$

Finally, the assortment optimization problem where customer choice follows a nested logit model is to find an assortment so that the expected revenue is maximized, i.e., we have:

$$Z \quad = \quad \max_{\substack{(S_1, \ldots, S_m) \\ S_i \in \mathcal{C}_i, i \in M}} \Pi(S_1, \ldots, S_m). \tag{2}$$

Davis et al. [9] proved that unless $\gamma_i \leqslant 1$ and $v_{i0} = 0$ for all $i \in M$, problem (2) is NP-hard. In this article, we provide a fast non-trivial exact solution framework for solving the AOPNL in a general setting, even when $\gamma_i > 1$ or $v_{i0} > 0$, for some $i \in M$.

1.2. Motivation and main contribution

The literature of assortment optimization under the nested logit model is quite rich concerning analytical properties of the problem under various assumptions. For example, one primary assumption is that the dissimilarity parameter of *all* nests is less than or equal to one (which means that all products within the same nest compete against each other). This assumption might be easily violated since some products might act synergistically. This situation has been thoroughly explained in the research of Davis et al. [9] and Segev [29]. These are, to our knowledge, the only papers that study scenarios in which dissimilarity parameters can be higher than one, and for that, the authors proposed approximation algorithms that provide assortments with a worst-case performance guarantee (see Section 1.3 for further details).

This paper provides an exact method based on fractional-programming to generate optimal assortments under the nested logit model. Our algorithm is not polynomial in the input size but is it sufficiently fast to solve to optimality problems with up to 5,000 products and 5 nests. Contrary to the existing literature, our method can be used to solve a wide range of

problem variants as it imposes no assumption on the input data. The key point of the method is that the parameterized subproblem at each iteration of the main fractional programming algorithm, which is a highly non-linear binary optimization problem, can be decomposed by nests. This enables us to solve a collection of subproblems that are relatively easier to handle. We prove in Section 3.1 that the subproblem is still NP-hard; however, it can be tackled with an efficient, tailored branch-and-bound algorithm.

Each subproblem for each nest is solved in two phases. First, a pre-processing enables to reduce the size of the subproblem by identifying revenue thresholds beyond which products are guaranteed to be offered or not. Then a Branch-and-Bound (B&B) algorithm solves the reduced problem, using both pre-processing and tailored upper bounds at each node. The pre-processing enables a large set of variables to be predetermined at each iteration via structural properties of the single-nest subproblem's newly-driven objective function. To the best of our knowledge, this research is the first to provide a generic non-trivial exact method for the assortment optimization problem under the nested logit model with proven optimality and practical efficiency (by non-trivial, we mean not based on a full enumeration of candidate assortments), which is flexible enough to adapt to any mix of dissimilarity parameters and any mix of utilities of no-purchase options. Moreover, our method is effective for instances of realistic size involving thousands of products, for which optimal assortments can be computed within short computing time.

The paper is organized as follows. In the remainder of this section, we provide a detailed literature overview. The general framework of our parametric search and the decomposition of the parameterized subproblem by nests are explained in Section 2. In Section 3, the parameterized subproblem is analyzed with the identification of the products that are sure to be offered or not at optimality, based on revenues, and the branch-and-bound algorithm for solving the reduced problem is provided. Section 4 is dedicated to numerical experiments to assess the performance of the method on many instances of various sizes, and we conclude our research in Section 5.

1.3. Related literature

The incorporation of choice models in assortment optimization problems (AOP) has attracted much attention in the recent decade. One of the most popular discrete choice models is the multinomial logit model. Since Talluri and Van Ryzin [31] demonstrated that under the multinomial logit model, offering revenue-ordered assortments maximizes the expected revenue, many researchers studied this class of problem under various settings. For example, Flores et al. [14] demonstrated that offering revenue-ordered assortments provides the optimal solution if the consumers' choice model follows a sequential multinomial logit, which is a generalized version of the classic MNL. In the context of robust optimization, Rusmevichientong and Topaloglu [28] demonstrated that even if the utilities of products are unknown yet belong to either a polyhedron or a rectangular uncertainty set, the cost of robustness is small, and the revenue-ordered assortments still provide maximum expected revenue.

The multinomial logit model was developed based on two main assumptions; 1) customers follow the utility maximization principle, and 2) the utilities of products are independent of each other. In real-world conditions, however, the second assumption might be violated. To

remedy this potential shortcoming of MNL, researchers developed other utility-maximizing models such as the Nested Logit model by Williams [35]. Detailed justifications of this model are available in [2, 6, 25]. We focus our literature review on the research that primarily uses variants of the nested logit model to incorporate the customer choice behavior in the assortment optimization problem. The available literature is rich and shows how active this area of research is.

Focusing on a popular form of this nested-logit model with only two levels – that is simply referred to as nested logit (NL) in the literature of AOP – and dealing with revenue maximization instead of pricing, Li and Rusmevichientong [21] developed a greedy algorithm to solve the AOP in polynomial time when the nest dissimilarity parameter is less than or equal to one for all nests and customers have a no-purchase option. In another class of AOP under the nested logit model, Gallego and Topaloglu [15] showed that if one imposes capacity and cardinality constraints on each nest separately, it is possible to get an optimal assortment by reducing the AOP to a knapsack problem. When having nest-specific space or capacity constraints, however, they show how to come up with a combination of products as a candidate assortment for each nest to have a worst-case approximation guarantee of 2.

When capacity constraints are not defined on each nest separately but over all nests, Feldman and Topaloglu [13] show that if each product consumes one unit of capacity, the problem can be solved to optimality using an algorithm that sequentially solves a linear and dynamic program. This approach will not return an optimal solution if we have a general capacity constraint across all nests, and for that case, they propose an algorithm that outputs a 4-approximate solution. In a different context, Davis et al. [10] studied a pricing problem where customer choices follow a nested logit model, and there is a relationship between product quality and price, which they call quality consistency constraint. For this problem, they developed a polynomial-time algorithm with proven optimality. In all the above articles where the focus is on the assortment planning problem under a nested logit model, the dissimilarity parameter is considered to be less than or equal to one.

Focusing only on a nest-specific capacity constraint, Désir et al. [11] developed a fully polynomial-time algorithm to provide an approximation scheme for the assortment optimization problem under various choice models, including nested logit. Segev [29] developed an approximation scheme for the capacitated AOP under the nested logit model in its most general form. The author proved that the approximation runs in a fully polynomial time, and for any $\epsilon > 0$, the obtained solution can be approximated within a factor of $1 - \epsilon$ of the optimal solution. Mai and Lodi [24] provided a general heuristic approach that can handle various AOPs under MNL, MMNL, and NL. They developed an iterative approach based on approximating the non-linear objective function by a linear one, combined with a greedy local search, that manages to find very good solutions for constrained problems.

In their thorough research, Davis et al. [9] studied the same version of the AOP with the nested logit model, which is the focus of our manuscript. They divide their work based on whether nest dissimilarity parameters are below one or not and whether the no-purchase option exists for each nest or not. They proved that if in all nests, products compete with each other, and if there is no option of leaving the nest without a purchase, the optimal assortment has sorted-by-revenue products for each nest. They prove that if the dissimilarity parameter

is less than 1 for all nests and customers can leave the nest without purchasing anything, their heuristic has a worst-case performance guarantee of 2 and if the dissimilarity parameter is free and customers cannot leave the nest without a purchase, offering sorted-by-revenue assortments in each nest we can get a worst-case performance guarantee of $\max\{\rho, 2\kappa\}$. Here κ bounds the ratio between the largest and the smallest preference parameters in a nest, i.e., $\kappa = \max_{i\in M}\{\frac{\max_{k\in N} v_{ik}}{\min_{k\in N} v_{ik}}\}$, and $\rho = \max_{i\in M}\{\frac{\max_{k\in N} r_{ik}}{\min_{k\in N} r_{ik}}\}$. Finally, for the most general case with free dissimilarity parameters and customers having the no-purchase option, the approach of Davis et al. [9] provides a performance guarantee of 2κ. They also give a pseudo-polynomial-time algorithm and an approximation algorithm with a performance guarantee of $\delta^{2\gamma_{max}+1}$ for any given $\delta > 1$. If δ is close to one, the performance of the obtained assortment is better, but with a higher computational time. Regarding the complexity of the problem, the authors also show that if customers have the no-purchase option or some nest dissimilarity parameters are greater than one, the AOPNL is NP-hard. This complexity motivated our research.

A generic nested logit model with more than two levels is studied in the research of Li et al. [22], for a pricing optimization problem. More recently, Chen and Jiang [7] incorporated product pricing in the context of assortment optimization also under the d-level nested logit model. They studied cardinality and capacity constraints and provided an $\epsilon-$approximate solution that can be found in fully polynomial-time under capacity constraint. They also developed an algorithm for the case of cardinality and nest-specific constraints that can obtain the optimal solution in strongly polynomial time.

While the above articles consider static settings when demand parameters are known a priori, Chen et al. [8] studied a dynamic assortment planning problem in which customer choice behavior follows a nested logit model and the decision-maker learns the demand as the assortment changes dynamically. As is common in the literature of dynamic assortment planning, the authors focused on a regret minimization problem and proposed efficient policies.

In Table 1, we classify the literature on the AOP under the nested logit model based on the value of the dissimilarity parameter of each nest and whether the proposed method provides an optimal solution or an approximation.

Table 1: An overview of research on assortment planning under nested logit

	$\gamma \in [0,\ 1]$			γ free
Solution	Unrestricted	Cardinality cons.	Capacity cons.	Unrestricted
Exact	[9, 21]	[13, 15]		our paper
Approx.			[11, 13, 15, 29]	[9]

2. Fractional programming approach

Recall from the previous section, the assortment optimization problem when customers' choice model follows a nested logit can be written as:

$$
\begin{aligned}
Z = \max_{\substack{(S_1,\ldots,S_m)\\ S_i\in\mathcal{C}_i, i\in M}} \quad & \Pi(S_1,\ldots,S_m) \quad \text{where} \\
\Pi(S_1,\ldots,S_m) &= \sum_{i\in M} Q_i(S_1,\ldots,S_m)\sum_{k=1}^{n} P_{ik}(S_i) r_{ik} \\
&= \sum_{i\in M} \frac{(v_{i0}+\sum\limits_{k\in S_i} v_{ik})^{\gamma_i}}{V_0+\sum\limits_{l\in M}(v_{l0}+\sum\limits_{k\in S_l} v_{lk})^{\gamma_l}} \times \frac{\sum\limits_{k\in S_i} r_{ik}v_{ik}}{v_{i0}+\sum\limits_{k\in S_i} v_{ik}} \\
&= \frac{\sum\limits_{i\in M}(v_{i0}+\sum\limits_{k\in S_i} v_{ik})^{\gamma_i-1}\sum\limits_{k\in S_i} r_{ik}v_{ik}}{V_0+\sum\limits_{i\in M}(v_{i0}+\sum\limits_{k\in S_i} v_{ik})^{\gamma_i}}. \qquad (3)
\end{aligned}
$$

2.1. Problem reformulation

Let x_{S_i} be a binary variable which is set to one if and only if we offer assortment $S_i \in \mathcal{C}_i$. Then AOPNL can be rewritten as the following non-linear binary program with an exponential number of binary variables:

$$
\begin{aligned}
\max \quad & \frac{\sum\limits_{i\in M}\sum\limits_{S_i\in\mathcal{C}_i}\left((v_{i0}+\sum\limits_{k\in S_i} v_{ik})^{\gamma_i-1}\sum\limits_{k\in S_i} r_{ik}v_{ik}\right)x_{S_i}}{V_0+\sum\limits_{i\in M}\sum\limits_{S_i\in\mathcal{C}_i}\left((v_{i0}+\sum\limits_{k\in S_i} v_{ik})^{\gamma_i}\right)x_{S_i}} = \frac{N(x)}{D(x)} \qquad (4)\\
\text{s.t:} \quad & \sum_{S_i\in\mathcal{C}_i} x_{S_i} = 1 && i\in M \\
& x_{S_i}\in\{0,1\}, && S_i\in\mathcal{C}_i, i\in M.
\end{aligned}
$$

Observe that if $S_i \in \mathcal{C}_i$, for all $i \in M$ are explicitly generated, then the expressions before variables x_{S_i} in the nominator $N(x)$ and denominator $D(x)$ of (4) are constant coefficients, and so the problem boils down to a binary fractional program in variables x_{S_i}. Integer fractional programming is a powerful modeling tool which has been successfully applied to a wide range of applications including landscape ecology and forest fragmentation [4], biodiversity conservation [5], inventory routing [1], portfolio optimization and wind-farm optmization [30], or network optimization [18].

Following Dinkelbach [12], Megiddo [27] proposed a method to solve fractional programs with 0-1 variables $x \in X$, where X is a set of linear constraints. The idea is to iteratively solve a so-called *parameterized* problem $F_{par}(\lambda) = N(x) - \lambda D(x)$ until $N(x) - \lambda D(x) = 0$ (in practice, $|N(x) - \lambda D(x)| \leqslant \epsilon$ where ϵ is a very small positive real number). The final value of λ that sets $N(x) - \lambda D(x)$ to be zero is the optimal ratio (λ^*), and the vector of variables x^* corresponding to λ^* is the vector optimizing the ratio, i.e.,

$$
\max_{x\in X}\left\{\frac{N(x)}{D(x)}\right\} = \frac{N(x^*)}{D(x^*)} = \lambda^*.
$$

For the AOPNL, the corresponding iterative *parameterized problem* $F_{par}(\lambda)$ is:

$$F_{par}(\lambda) = \max_{\substack{(S_1,...,S_m) \\ S_i \in \mathcal{C}_i, i \in M}} \left\{ \sum_{i \in M} \left((v_{i0} + \sum_{k \in S_i} v_{ik})^{\gamma_i - 1} \sum_{k \in S_i} r_{ik} v_{ik} \right) - \lambda \left(V_0 + \sum_{i \in M} (v_{i0} + \sum_{k \in S_i} v_{ik})^{\gamma_i} \right) \right\} \quad (5)$$

The following theorem follows from the findings in Megiddo [27].

Theorem 1. *Let Z^* be the optimal value of problem* (3)*. If one can find a λ^* for which $F_{par}(\lambda^*) = 0$, then the collection of assortments $(S_1(\lambda^*), ..., S_m(\lambda^*))$ defined as the solution of problem* (5) *solved with $\lambda = \lambda^*$, is the optimal solution to problem* (4) *(equivalent to problem* (3)*) and $Z^* = \lambda^*$.*

We now show that the above parameterized problem is decomposable by nest.

2.2. Decomposability of the parameterized subproblem

Proposition 1. *The parameterized problem* (5) *is separable by nest and can be written as:*

$$F_{par}(\lambda) = -\lambda V_0 + \sum_{i \in M} \max \left\{ (v_{i0} + \sum_{k \in N} v_{ik} x_{ik})^{\gamma_i - 1} (\sum_{k \in N} v_{ik}(r_{ik} - \lambda) x_{ik} - \lambda v_{i0}) \right\} \quad (6)$$
$$s.t: \quad x_{ik} \in \{0, 1\}, k \in N, i \in M.$$

where $x_{ik} = 1$ if and only if product $k \in N$ is offered in nest $i \in M$.

Proof. Since in problem (5) we have simple assignment constraints of choosing one assortment per nest, we observe that the objective of the parameterized program can be rewritten as:

$$-\lambda V_0 + \sum_{i \in M} \max_{S_i \in \mathcal{C}_i} \left\{ (v_{i0} + \sum_{k \in S_i} v_{ik})^{\gamma_i - 1} \left(\sum_{k \in S_i} v_{ik}(r_{ik} - \lambda) - \lambda v_{i0} \right) \right\}.$$

Hence, the program is *decomposable by nest*, and the maximization subproblem for nest i is given as:

$$\max_{S_i \in \mathcal{C}_i} \left\{ (v_{i0} + \sum_{k \in S_i} v_{ik})^{\gamma_i - 1} \left(\sum_{k \in S_i} v_{ik}(r_{ik} - \lambda) - \lambda v_{i0} \right) \right\}. \quad (7)$$

We can see the above problem as a binary non-linear program with decision variables x_{ik} indicating whether product $k \in N$ is offered in nest $i \in M$, and so the whole problem can be reformulated like in (6). □

This result enables us to speed up the resolution of the subproblem by separately solving a smaller problem for each nest $i \in M$. To find the optimal value of λ, we use the framework proposed by Megiddo [27] that is illustrated in Algorithm 1. The algorithm is a dichotomic

search on λ and it is based on the ability to solve the parametric problem in an exact way. It is also based on the fact that function $F_{par}(\lambda)$ in (6) is piece-wise linear in λ. We refer the reader to [12, 27] for further details on the algorithm and its proven convergence to optimality.

2.3. Algorithmic details

To initialize our parametric search, we first need to determine a lower bound λ_l and an upper bound λ_u for λ^*. The value of λ_l corresponds to a primal bound of the AOPNL that can be obtained by applying a heuristic approach. We choose to set the initial value of λ_l by running the heuristic proposed by [9] in which λ_l corresponds to the maximum expected revenue obtained by offering sorted-by-revenue assortments, which is known to be the optimal value of the following linear program (see [9]):

$$\begin{aligned} \lambda_l \quad = \quad & \min \quad \lambda & (8)\\ & \text{s.t:} \quad V_0\lambda \geqslant \sum_{i\in M} y_i & (9)\\ & \qquad\quad y_i \quad \geqslant V(N_{ik})^{\gamma_i}(R(N_{ik}) - \lambda), \quad i \in M, k \in N. & (10) \end{aligned}$$

where $N_{ik} = \{1, \ldots, k\}$, for $k \in N$, is the subset of k highest-revenue products for nest $i \in M$ (sorted-by-revenue assortments), $V(N_{ik}) = v_{i0} + \sum\limits_{k\in N_{ik}} v_{ik}$ and $R(N_{ik}) = \sum_{k\in N_{ik}} r_{ik}v_{ik}/V(N_{ik})$.

To determine an initial upper bound λ_u on λ, observe that in the expression of $\Pi(S_1, \ldots, S_m)$ at the second line of (3), probabilities Q_i and P_{ik} sum over at most one (less than one if V_0 and v_{i0} are positive). Therefore, the expected revenue (Z) is bounded above by a linear combination of revenues r_{ik} with weights summing over one, which is at most the highest revenue. Therefore, we set

$$\lambda_u = \max\{r_{ik} : k \in N, i \in M\}.$$

We notice that in the work of Davis et al. [9], the authors propose a much tighter upper bound for the AOPNL, which is based on solving a continuous relaxation of the problem reformulation. The latter boils down to a convex semi-infinite program which is solved using a cutting plane approach, based on outer approximation. Calculation of this bound is computationally too expensive, which is why we stick to a simple bound that can be precomputed in the initialization phase.

Moreover, in the following proposition, we provide an additional stopping criterion that speeds up the dichotomic search (see, e.g., [18]).

Proposition 2. *At any iteration of the dichotomic search, let $S_l(\lambda_l)$ and $S_u(\lambda_u)$ be the optimal assortments found for $F_{par}(\lambda_l)$ and $F_{par}(\lambda_u)$, respectively. If $S_l(\lambda_l) = S_u(\lambda_u)$, then $S^* = S_l(\lambda_l) = S_u(\lambda_u)$ is the optimal assortment for the original problem.*

We prove this proposition in Appendix A.1. In the pretests, the above criterion was shown to be an effective measure to reduce the processing time of the developed parametric search. We summarize the steps of the developed parametric search in Algorithm 1. We start by

initializing the values for λ_u and λ_l as described above. The function "Solve $F_{par}(\lambda)$", called in Step 8 returns the optimal assortment S^* over all subproblems with respect to parametric objective function value given in (5), and the optimal solution value F^*. Upon calling this function, $|M|$ subproblems are solved independently, according to Proposition 1. As we prove in Proposition 3, the single-nest subproblem is NP-hard. For that reason, we provide a detailed analysis of the subproblem in Section 3 and develop an exact branch-and-bound method to solve it. If the time limit is reached, our framework returns the best incumbent solution S_{best} as well as a final upper bound λ_u, which can be used to measure the quality of S_{best}. Our empirical results (see Section 4.2) indicate that, the final upper bound λ_u exhibits very small gaps with respect to the best-found solution (for those instances for which the optimal solution could not be found).

Algorithm 1: Parametric search for solving the AOPNL

Data: Instance of the AOPNL problem, time limit TL
Result: Optimal or best found (if TL is reached) solution S_{best}. The upper bound λ_u.

1 Initialize (λ_l, λ_u);
2 $S_l \leftarrow$ Optimal assortment from solving (8)–(10);
3 $S_u \leftarrow \emptyset$;
4 $S_{best} = S_l$;
5 $F^* = \inf$;
6 **while** $|F^*| > 0$ *and* $S_l \neq S_u$ *and TL is not reached* **do**
7 $\quad \lambda \leftarrow (\lambda_l + \lambda_u)/2$;
8 $\quad (F^*, S^*) \leftarrow$ Solve $F_{par}(\lambda)$;
9 $\quad$ **if** $\Pi(S^*) > \Pi(S_{best})$ **then**
10 $\quad\quad S_{best} \leftarrow S^*$;
11 $\quad$ **if** $F_{par}(\lambda) < 0$ **then**
12 $\quad\quad \lambda_u \leftarrow \lambda$, $S_u \leftarrow S^*$;
13 $\quad$ **else**
14 $\quad\quad \lambda_l \leftarrow \lambda$, $S_l \leftarrow S^*$;
15 return (S_{best}, λ_u);

3. Subproblem analysis and the branch-and-bound

In the previous sections, we showed how implementing a fractional programming framework enables us to deal with one nest at a time. In other words, instead of considering the original assortment optimization problem (3) with all nests, we can focus on providing the optimal assortment for each nest with a different objective function that is derived from the parametric function (6). We dedicate this section to study and analyze the properties of the resulting subproblem. We observe that removing the constant term λV_0 from formulation (6) does not change the optimal solution, and throughout this section, we drop index i and refer to the following problem as the subproblem of the parametric search algorithm, that we denote by (NLAPP) for the *Nested-Logit Assortment Parameterized Problem*:

$$\text{(NLAPP)} \quad \max_{x \in \{0,1\}^{|N|}} \left\{ (v_0 + \sum_{k \in N} v_k x_k)^{\gamma - 1} (\sum_{k \in N} v_k (r_k - \lambda) x_k - \lambda v_0) \right\}. \tag{11}$$

where x_k is a binary decision variable on whether to offer product k in the given nest or not. We first show NP-hardness of that problem, then we describe the revenue thresholds for fixing variables and the branch-and-bound algorithm.

3.1. Subproblem complexity

In this subsection, we prove that the subproblem is NP-hard by reduction from the Subset-Sum problem.

Proposition 3. *The subproblem (NLAPP) is NP-hard for any $\gamma > 1$.*

Proof. We reduce the Subset-Sum decision problem, to our nested-logit parameterized subproblem (NLAPP) with arbitrary values of $\gamma > 1$ and $\lambda \geq 1$. The Subset-sum problem (SSP) which is a known NP-complete problem (see [16], Chapter 3) is described as follows. Given n integer numbers $a_1, \ldots, a_n$ and an integer number $B < \sum_{k=1}^{n} a_k$, is there a subset $S \subset \{1, \ldots, n\}$ such that $\sum_{k \in S} a_k = B$?

We reduce a general instance of (SSP) to a specific (NLAPP) as follows. Set v_0 at some arbitrary value less than B. Set $N = \{1, \ldots, n, n+1\}$, $v_k = a_k$ and $r_k = \lambda - 1$ for $k = 1, \ldots, n$. Moreover, we set:

$$\begin{aligned} v_{n+1} &= B - v_0, \quad \text{and} \\ r_{n+1} &= \lambda + \frac{B\left(\frac{\gamma+1}{\gamma-1}\right) + \lambda v_0}{v_{n+1}}. \end{aligned}$$

Define constant

$$C_1 \;=\; v_{n+1}(r_{n+1} - \lambda) - \lambda v_0.$$

After substituting v_{n+1} and r_{n+1} by their expression above in C_1, we obtain

$$C_1 \;=\; B\left(\frac{\gamma+1}{\gamma-1}\right).$$

Based on Proposition 7, we know that product $n+1$ is guaranteed to be offered in any optimal assortment since $r_{n+1} > \lambda$ and from the formulation above, $C_1 > 0$. Therefore, $x^*_{n+1} = 1$, and after denoting $V_1 = v_0 + v_{n+1} = B$, the subproblem reduces to the following problem (Q):

$$(Q) \max_{x \in \{0,1\}^n} f(x) = \left(V_1 + \sum_{k=1}^{n} v_k x_k\right)^{\gamma-1} \times \left(C_1 + \sum_{k=1}^{n} v_k (r_k - \lambda) x_k\right). \tag{12}$$

We show that we can find a solution of (Q) with value at least $\frac{(2B)^\gamma}{\gamma-1}$ if and only if the answer to the subset-sum decision problem is yes, i.e., there exists a subset $S \subset \{1, \ldots, n\}$ such that $\sum_{k \in S} a_k = B$. Observe that with the above setting, our (NLAPP) instance can be rewritten as:

$$\begin{aligned} \max_{x \in \{0,1\}^n} f(x) \;&=\; \left(V_1 + \sum_{k=1}^{n} a_k x_k\right)^{\gamma-1} \times \left(C_1 - \sum_{k=1}^{n} a_k x_k\right) \\ &=\; (V_1 + g(x))^{\gamma-1} \times (C_1 - g(x)) \qquad \text{with } g(x) = \sum_{k=1}^{n} a_k x_k. \end{aligned}$$

Now define function $h(X) = (V_1 + X)^{\gamma-1} \times (C_1 - X)$ over $[0, C_1]$, such that $h(g(x)) = f(x)$. Observe that without loss of generality we can focus on the interval $[0, C_1]$, since for $X > C_1$ we have $h(X) < 0$. Function h is differentiable and we have

$$\begin{aligned} h'(X) &= (C_1 - X)(\gamma - 1)(V_1 + X)^{\gamma-2} - (V_1 + X)^{\gamma-1} \\ &= (V_1 + X)^{\gamma-1} \left(\frac{(C_1 - X)(\gamma - 1)}{V_1 + X} - 1 \right) \\ &= 0 \text{ if and only if } X = C_1 \frac{\gamma - 1}{\gamma} - \frac{V_1}{\gamma} = B. \end{aligned}$$

Since $h'(X) \geq 0$ if and only if $X \leq B$, function h is increasing then decreasing over $[0, C_1]$ (observe $h(C_1) = 0$) and reaches it maximum for $X = B$, with

$$h(B) = (2B)^{\gamma-1} \times \left(B \left(\frac{\gamma + 1}{\gamma - 1} \right) - B \right) = 2^{\gamma-1} B^{\gamma} \left(\frac{2}{\gamma - 1} \right) = \frac{(2B)^{\gamma}}{\gamma - 1} > 0;$$

We conclude that $x \in \{0, 1\}^n$ reaches value $\frac{(2B)^{\gamma}}{\gamma-1}$ for the objective function of (Q) if and only if $g(x) = \sum_{k=1}^{n} a_k x_k = B$ which is a solution for the Subset-Sum decision problem. Since the latter is NP-complete and the reduction described above is a polynomial reduction, this completes the NP-completeness proof for the decision version of (NLAPP). □

We now describe the Branch-and-Bound (B&B) algorithm. We classify our analysis based on the value of the dissimilarity parameter of a nest and consequently, the properties of the subproblem for a nest i change according to the value of γ_i.

3.2. A tailored Branch-and-Bound algorithm

We develop a tailored B&B algorithm to determine the optimal assortment for each nest. We have a set of variables that have been fixed to 1 throughout the B&B tree until node t. We denote this set by K_1^{t-1}. We also have a set of variables that have been fixed to 0. We use K_0^{t-1} to denote this set. Finally, we have a set of undecided variables denoted by $\bar{K}^{t-1}$. We define

$$C_1^{t-1} = \sum_{j \in K_1^{t-1}} v_j (r_j - \lambda) - \lambda v_0,$$

as the sum of the weighted revenue of products offered thus far in a particular nest. We also define

$$V_1^{t-1} = \sum_{j \in K_1^{t-1}} v_j + v_0,$$

as the sum of preferences for the respective products. Using these notations, at each node t in the B&B tree, the objective function to be maximized is:

$$\max_{x\in\{0,1\}^{|\bar{K}^{t-1}|}} f^t(x) = \left(V_1^{t-1} + \sum_{k\in\bar{K}^{t-1}} v_k x_k\right)^{\gamma-1} \times \left(C_1^{t-1} + \sum_{k\in\bar{K}^{t-1}} v_k(r_k-\lambda)x_k\right). \tag{13}$$

We naturally branch on binary variable $x_k \in \bar{K}^{t-1}$ fixing $x_k = 1$ and $x_k = 0$ in the two child nodes of a node t of the tree. In the branching ordering, we choose to branch on a variable with highest revenue, as fixing such variables to one has a strong potential impact on the value of objective function. We enumerate the branching tree in a depth-first-search fashion, following branches that fix variables to one. The main issue is how to estimate an upper bound on the objective value of a node, say t, that is both *i*) tight enough to enable efficient pruning, and *ii*) tractable enough to be quickly computed, despite the non-linearity of the objective. Observe that the structure of the subproblem varies with the value of the nest dissimilarity parameter γ, considering two cases: $\gamma \leqslant 1$ and $\gamma > 1$.

It turns out that the subproblem has interesting characteristics that allow to perform a *preprocessing* procedure and eliminate some of the variables without branching, which in turn speeds up the computation time. We update sets $\bar{K}^{t-1}$, K_0^{t-1} and K_1^{t-1} and constants C_1^{t-1} and V_1^{t-1} by removing or adding variables that were fixed either by branching or the preprocessing. In the next sections, we provide detailed descriptions of this preprocessing procedure and upper bound calculations, for each case $\gamma \leqslant 1$ and $\gamma > 1$.

3.3. Competitive products ($\gamma \leqslant 1$)

We identify three possible scenarios at the root node. For each scenario, we derive some measures to perform a preprocessing and potentially fix a relatively large number of variables before entering the B&B tree. We first observe that if $\gamma \leqslant 1$, we can reformulate the objective function (11) at the root node 0 as:

$$\max_{x\in\{0,1\}^{|N|}} f^0(x) = \left\{ \frac{\sum_{k\in N} v_k(r_k-\lambda)x_k - \lambda v_0}{(v_0 + \sum_{k\in N} v_k x_k)^{1-\gamma}} \right\}, \tag{14}$$

and we proceed to the following proposition on preprocessing at the root node. We provide pseudo-codes of all preprocessing procedures in Appendix B.1 through Appendix B.4.

Proposition 4. *At the root node ($t = 0$), set $\mathcal{H} = \sum_{k\in N: r_k \geqslant \lambda} v_k(r_k - \lambda) - \lambda v_0$, and let x^* denote an optimal solution of the subproblem. We have three possible scenarios based on the value of $\mathcal{H}$:*

i) if $\mathcal{H} = 0$ (`Case 0`), the problem is solved by preprocessing without branching; defining $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$ and $K_0^0 = \{k \in N : r_k < \lambda\}$, the optimal solution of the subproblem is $x_k^* = 1$ for $k \in K_1^0$ and $x_k^* = 0$ for $k \in K_0^0$.

ii) if $\mathcal{H} > 0$ (`Case 1`), let $K_0^0 = \{k \in N : r_k < \lambda\}$, then $x_k^* = 0$ for $k \in K_0^0$. Moreover, define $K' = N \setminus K_0^0$ and $r_{max} = \max_{j\in N}\{r_j\}$ and let $K_1^0 = \{k \in N \setminus K_0^0 : r_k \geqslant \lambda + (1-\gamma)(r_{max} - \lambda - (v_0 r_{max})/(v_0 + \sum_{j\in K'} v_j))\}$, then $x_k^* = 1$ for $k \in K_1^0$.

iii) if $\mathcal{H} < 0$, `(Case 2)` *let* $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$, *then* $x_k^* = 1$ *for* $k \in K_1^0$. *Moreover, we define* $K' = N \setminus K_1^0$, $r_{min} = \min_{j \in K'}\{r_j\}$, *and* $\Delta = C_1^0 - V_1^0(r_{min} - \lambda)$, *then we have two cases:*

- *If* $\Delta \geqslant 0$, $x_k^* = 0$ *for* $k \in K_0^0 = \left\{k \in K' : r_k \leqslant \lambda + (1-\gamma)\left(\frac{C_1^0 + (r_{min}-\lambda)\sum_{j\in K'} v_j}{V_1^0 + \sum_{j\in K'} v_j}\right)\right\}$.
- *If* $\Delta < 0$, *then* $x_k^* = 0$ *for* $k \in K_0^0 = \left\{k \in K' : r_k \leqslant \lambda + (1-\gamma)\left(C_1^0/V_1^0\right)\right\}$.

We provide the proof of the above proposition in Appendix A.2. Remark that its findings are crucial for the preprocessing of nodes other than the root node. It is trivial that if $\mathcal{H} = 0$ at the root node, we prune the root node and the optimal assortment is found for the nest. Note that this condition $\mathcal{H} = 0$ was very rarely satisfied in the numerical experiments. Remark that if $\mathcal{H} > 0$, after performing the pre-processing of Proposition 4 we have $r_k \geqslant \lambda$ at any other node of the B&B tree. On the other hand, if $\mathcal{H} < 0$ we have $r_k < \lambda$ for the remaining undecided variables at any node t.

These findings are the result of the structure of the tree and follow from the proof of Proposition 4 and the fact that in *Case 1*, where $\mathcal{H} > 0$, there exists a combination of products that realizes a positive value for the objective function (11). This means that after fixing all the variables with $r_k < \lambda$ to zero at the root node, it is not beneficial to add any of these variables to the optimal assortment of a nest at any other node since doing so would provide an objective value that is less than the one at the root node which is in contradiction with the maximization of the objective function. We can follow a similar logic for *Case 2*. We provide the details of preprocessing for the remaining of the tree in the following proposition (with proof in Appendix A.3).

Proposition 5. *At any node* $t \neq 0$, *depending on the value of* $\mathcal{H}$ *calculated at the root node, we can have one of the two following scenarios:*

i) if $\mathcal{H} > 0$ `(Case 1)`, *set* $r_{max} = \max_{j\in\bar{K}^{t-1}}\{r_j\}$ *and define* $K_1^t = \left\{k \in \bar{K}^{t-1} : r_k \geqslant \lambda + (1-\gamma)\left(\frac{C_1^{t-1} + (r_{max}-\lambda)\sum_{j\in\bar{K}^{t-1}} v_j}{V_1^{t-1} + \sum_{j\in\bar{K}^{t-1}} v_j}\right)\right\}$. *Then* $x_k^* = 1$ *for* $k \in K_1^t$.

ii) if $\mathcal{H} < 0$ `(Case 2)`, *we define* $r_{min} = \min_{j\in\bar{K}^{t-1}}\{r_j\}$ *and* $\Delta = C_1^{t-1} - V_1^{t-1}(r_{min} - \lambda)$, *then we have two cases:*

- *If* $\Delta \geqslant 0$, $x_k^* = 0$ *for* $k \in K_0^t = \left\{k \in \bar{K}^{t-1} : r_k \leqslant \lambda + (1-\gamma)\left(\frac{C_1^{t-1} + (r_{min}-\lambda)\sum_{j\in\bar{K}^{t-1}} v_j}{V_1^{t-1} + \sum_{j\in\bar{K}^{t-1}} v_j}\right)\right\}$.
- *If* $\Delta < 0$, *then* $x_k^* = 0$ *for* $k \in K_0^t = \left\{k \in \bar{K}^{t-1} : r_k \leqslant \lambda + (1-\gamma)\left(C_1^{t-1}/V_1^{t-1}\right)\right\}$.

After fixing variables using the above revenue thresholds, we restrict the set of undecided variables to $\bar{K}^t$ and at each node t after preprocessing, the subproblem will reduce to:

$$\max_{x\in\{0,1\}^{|\bar{K}^t|}} f^t(x) = \left(V_1^t + \sum_{k\in\bar{K}^t} v_k x_k\right)^{\gamma-1} \times \left(C_1^t + \sum_{k\in\bar{K}^t} v_k(r_k - \lambda)x_k\right). \tag{15}$$

We now proceed to the development of an efficient upper bound for function (15) at each node t. These bounds are tight enough to enable efficient pruning, and tractable enough to be quickly computed despite the high non-linearity of the objective (15). For the case where $C_1^t > 0$, we derive a bound in the following proposition (proof in Appendix A.4).

Proposition 6. *[Upper bound of $f^t(x)$ in (15)] If $C_1^t > 0$, then*

$$f^t(x) \leqslant (V_1^t)^{\gamma-1}\ C_1^t\ e^{z_1^*(t)}, \tag{16}$$

where

$$z_1^*(t) = \sum_{k\in\bar{K}} v_k \max\left(0, \frac{2(\gamma-1)}{2V_1^t + \sum\limits_{k\in\bar{K}^t} v_k} + \frac{(r_k-\lambda)}{C_1^t}\right).$$

Preliminary numerical experiments enabled to show that using this bound for pruning branches improves the computation time of the Branch-and-Bound algorithm (by roughly 30%).

3.4. Possibly synergistic products ($\gamma > 1$)

In this section, we analyze the properties of subproblem (11) when the dissimilarity parameter of a nest is higher than one. These properties enable again to fix variables without further calculation. We summarize these properties in the preprocessing phase. The structure of the B&B tree depending heavily on the characteristics of the root node, again we first introduce the preprocessing at the root node in the following proposition (proof in Appendix A.5).

Proposition 7. *Define $\mathcal{H} = \sum_{k\in K: r_k \geqslant \lambda} v_k(r_k - \lambda) - \lambda v_0$. Depending on the value of $\mathcal{H}$, we can have one of the following scenarios:*

i) if $\mathcal{H} = 0$ (`Case 0`), then defining $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$ and $K_0^0 = \{k \in N \setminus K_1^0 : r_k < \lambda\}$, we have $x_k^* = 1$ for $k \in K_1^0$ and $x_k^* = 0$ for $k \in K_0^0$, and the subproblem is solved without branching.

ii) if $\mathcal{H} > 0$ (`Case 1`), we define $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$. Then $x_k^* = 1$ for $k \in K_1^0$. In addition, set $K_0^0 = \{k \in N \setminus K_1^0 : r_k < \lambda - (\gamma-1)(C_1^0/V_1^0)\}$. Then $x_k^* = 0$ for $k \in K_0^0$.

iii) if $\mathcal{H} < 0$ (`Case 2`), define $K_0^0 = \{k \in N : r_k < \lambda\}$. Then $x_k^* = 0$ for $k \in K_0^0$.

As for the remaining of the tree, we benefit from the result of the following proposition (See Appendix A.6 for the proof).

Proposition 8. *If $\mathcal{H} > 0$, at any node $t \neq 0$, $x_k^* = 0$ for $k \in K_0^t$, where $K_0^t = \{k \in \bar{K}^{t-1} : r_k < \lambda - (\gamma-1)(C_1^t/V_1^t)\}$.*

We note then the objective function after preprocessing at node t:

$$\max_{x\in\{0,1\}^{|\bar{K}^t|}} f^t(x) = \left(V_1^t + \sum_{k\in\bar{K}^t} v_k x_k\right)^{\gamma-1} \times \left(C_1^t + \sum_{k\in\bar{K}^t} v_k(r_k-\lambda)x_k\right). \tag{17}$$

We now design an upper bound for pruning nodes in the case where $\gamma > 1$ and $C_1^t > 0$.

Proposition 9. *[Upper bound of $f^t(x)$ in (17)] If $C_1^t > 0$ then*

$$f^t(x) \leqslant (V_1^t)^{\gamma-1}\ C_1^t\ e^{z_2^*(t)},$$

where

$$z_2^*(t) = \sum_{k\in\bar{K}} v_k \max\left(0, \frac{\gamma-1}{V_1^t} + \frac{r_k-\lambda}{C_1^t}\right).$$

See Appendix A.7 for the proof. The above upper bound can be computed in linear time $O(|\bar{K}^t|)$, if revenues r_k are already sorted in a pre-processing phase.

4. Numerical experiments

In this section, we analyze the results of an extensive set of computational experiments designed to evaluate the performance of our tailored branch-and-bound algorithm embedded in the fractional programming framework (denoted by `FP+BB`) compared to the state-of-the-art heuristic proposed by Davis et al. [9], denoted by `Heuristic`. The solution obtained by this heuristic is the optimal solution of the Linear Program (8)-(10) with revenue-ordered candidate assortments in the LP. All the experiments have been carried out on a virtual machine with an Intel(R) Core i7-3770 CPU, a 3.4 GHz processor with 8GB of RAM using a 64-bit Windows operating system. The code was written and run using `Julia v0.6.4`.

4.1. Experimental setup

We divide our computational experiments into three main groups based on the number of products available. For the first group, we solve instances of the AOPNL with $m = 5$ nests and $n = 10$ products per nest. We compute the maximum expected revenue by offering sorted-by-revenue assortments in each nest (`Heuristic` method of Davis et al. [9]) and using two variants of our fractional programming approach:

- `FP+BB:` in which the parameterized subproblem is solved by our tailored branch-and-bound algorithm described in Section 3, and
- `FP+F.Enum:` in which a full enumeration of all possible assortments is used to solve the subproblems. Clearly, this approach was possible only very small problem instances, which is why we restrict this experiment to $n = 10$.

In our pretests, as possible alternatives for solving the non-linear subproblems, we also tested Couenne and Bonmin – two open source solvers for Mixed Integer Nonlinear Programming (MINLP) problems; however, due to numerical issues and the structure of the objective function, these solvers could not find the optimal solution for many instances, so we excluded the use of these solvers from our experiments.

As for the second group, we solve instances with $m = 5$ nests and $n \in \{25,\ 50,\ 100,\ 250\}$ products per nest. In this case, we had to exclude full enumeration of all assortments for the subproblem, which was intractable. Finally, for the last group with super large instances, we solve instances with $m = 5$ and $n \in \{500,\ 1000,\ 5000\}$. Remark that although we have the number n of potential products per nest, we do not impose any assumption on the equality of the number of products finally offered per nest, in the optimal assortment.

To generate the value of preferences v_{ik} and revenues r_{ik}, we follow a similar procedure as in Gallego and Topaloglu [15]. Recall that index i identifies a nest and index k identifies a product. We randomly generate values U_{ik} using a uniform distribution in the interval $[0,\ 1]$. We then randomly generate values X_{ik} and Y_{ik} with a uniform distribution in $[50,\ 300]$ and use these values to generate revenues and preferences as follow: $r_{ik} = 10 \times U_{ik}^2 \times X_{ik}$, $v_{ik} = 10 \times (1 - U_{ik}) \times Y_{ik}$. Using U_{ik}, products with larger revenues are more likely to have smaller preference weights, which means that more expensive products are usually less attractive. However, X_{ik} and Y_{ik} add some noise so that not always products that are more expensive have low attractiveness. The square power in U_{ik}^2 skews the distribution of revenues in order to have a large number of products with small revenues and a small number of products with large revenues ([15, 13]).

To generate the value of dissimilarity parameters $\gamma_i, \forall i \in M$, we use a uniform distribution in $[\gamma^L, \gamma^U]$ with the same intervals $[0.5,\ 1.5]$, $[1,\ 2]$, $[1.5,\ 2.5]$ and $[2,\ 3]$ as in the research of Davis et al. [9]. We set $V_0 = 30$ and for a thorough analysis, for each value of n, we consider three cases where $v_{i0} = 0, \forall i \in M$, $v_{i0} = 30, \forall i \in M$ and finally, each nest $i \in M$ has either $v_{i0} = 0$ or $v_{i0} = 30$ with equal probability $1/2$.

For our `FP+BB` method, we record the overall number of iterations. We also set a time limit of 3,600 seconds for each instance. If `FP+BB` does not converge to optimality within the time limit, we report the value of the best found assortment ($\Pi(S_{best})$ from Algorithm 1). In the latter case, we also calculate the last value of λ_u obtained during the parametric search, as a valid upper bound on the expected revenue of an optimal assortment.

4.2. Computational results

We provide the results of our computational experiments in Tables 2–7. Recall that we solve problems with a number of products ranging from $n = 10$ to $n = 5000$. We report the results of the experiments corresponding to every value of n in a separate table. In all tables, the first column determines the combination of parameters v_{i0} and γ_i. We also consider various intervals for the generation of dissimilarity parameter which we show by $[\gamma^L,\ \gamma^U]$. To better observe the impact of v_{i0} on the performance of `FP+BB`, we consider separate groups of instances. In all tables we solve instances of the AOPNL by offering the best sorted-by-revenue assortments in each nest by solving the linear problem expressed through formulation (8)–(10) and using `FP+BB`. We report the CPU time in seconds of each method (Time) along with the

best expected revenue (Obj.) found by each method. For each combination of parameters (each row), that we call "scenario" in what follows, we solve 20 instances. For each scenario, the average percentage of improvement in the expected revenue using `FP+BB` is determined as:

$$\text{Impr.}(\%) = \frac{1}{20}\sum_{p=1}^{2} 0100 \times \left(\frac{\text{BEST}^p - \text{HEUR}^p}{\text{HEUR}^p}\right),$$

where $p = 1, \ldots, 20$ denotes each instance of the considered scenario, HEUR^p is the expected revenue obtained by offering sorted-by-revenue assortments for instance p and finally, BEST^p is the optimal expected revenue found by `FP+BB` (or $\Pi(S_{best})$, if it reaches the time limit). This gap can also be interpreted as the average improvement in revenue obtained by `FP+BB`. We use Impr.(%) to demonstrate this value in the tables.

For the `FP+BB` method, we also report for each scenario: the number of instances (out of 20) for which the optimal solution was found (# Opt); the overall number of instances (out of 20) for which `FP+BB` has improved the expected revenue found by `Heuristic`, and the average number of iterations of the `FP+BB` approach. Finally, for the `FP+F.Enum.` method we report the average CPU time in seconds (Time).

In Table 2, we report the results for instances with $n = 10$. As these instances are small, we are able to determine the optimal solution for the subproblem (11) and consequently, for the original AOPNL (3), by generating all 2^n possible assortments for each nest. As a result, in Table 2, we report the CPU time of the `FP+F.Enum.` method used to generate all possible assortments when solving the subproblem, instead of using the specialized branch-and-bound.

Results in this table show that our `FP+BB` method performs well in terms of finding the optimal solution with a processing time almost equal to the heuristic of [9]. We can also observe that the heuristic exhibits significant gaps with respect to the optimal solution. As expected, regardless of the value of v_{i0}, the overall trend is that the performance of `FP+BB` improves as the values in $[\gamma^L,\ \gamma^U]$ increase. We can also observe that in general and regardless of the value of v_{i0}, if all $\gamma_i > 1$, the performance of the heuristic method degrades compared to the case where some $\gamma_i \leqslant 1$ and the `FP+BB` provides a much higher revenue. For example, if we shift from $[\gamma^L,\ \gamma^U] = [0.5,\ 1.5]$ to $[\gamma^L,\ \gamma^U] = [1,\ 2]$, using the fractional programming approach we will get an average revenue improvement of 23% if $v_{i0} = 0, \forall i \in M$. We also expected that using full enumeration is computationally much costlier than following our proposed B&B – the `FP+F.Enum.` is an order of magnitude slower than `FP+BB` for the smallest instances with $n = 10$, but already for $n = 25$ the full enumeration approach was intractable.

In Tables 3–6, we give our main results obtained by solving instances with $n \in \{25,\ 50,\ 100,\ 250\}$. The main observation is that as long as $v_{i0} = 0, \forall i \in M$, we are able to obtain the optimal solution using `FP+BB` in less than 0.3 second and with a relatively significant improvement in the expected revenue. Same as before, as the value of $[\gamma^L,\ \gamma^U]$ increases, the expected revenue improvement realized by using `FP+BB` increases and the performance of the sorted-by-revenue assortments degrades. This is the general pattern in all the tables regardless of the value of v_{i0} and the number of products. This shows the performance of our exact algorithm `FP+BB`

for solving diverse cases of the AOPNL and it can be explained by the possible synergy between products which is also in line with the findings of Davis et al. [9]. As the value of γ increases, the degree to which products might act as synergies increases and as a result, the sorted-by-revenue assortment structure might not be optimal. For a detailed explanation, see Sections 5 and 7 in the work of Davis et al. [9].

For example, if we shift the interval $[\gamma^L,\ \gamma^U]$ from $[1.5,\ 2.5]$ to $[2,\ 3]$, based on the results in the eighth column of Table 3, we get an average improvement in the expected revenue by over 6% if $v_{i0} = 0, \forall i \in M$ or if $v_{i0} > 0, \forall i \in M$. We observe a similar pattern of revenue improvement for higher ranges of γ as shown in Tables 4–6; however, this improvement becomes less significant as the number of products increases. Another observation is that when $v_{i0} > 0$ even for some nests, the parametric search can sometimes reach the time limit. However, the resulting `FP+BB` solution still provides a sizable improvement in the expected revenue as seen in the fourth and eighth columns of Tables 3–6. We also see that even in the case of having some $v_{i0} > 0$, the algorithm can solve problems in less than 4 seconds.

As can be expected, with an increase in the number of products, the time required for the parametric search to converge to the optimal solution also increases. Such an increase is more drastic if all or even some of the v_{i0} have non-zero values and if $\gamma_i \leqslant 1$ for some nests. As a result, in Table 6, we observe that unless $\gamma_i > 1$ or $v_{i0} = 0$, $\forall i \in M$, the search process of `FP+BB` reaches the time limit; however, even in that case, using `FP+BB` provides an improvement (albeit a rather modest one) in the expected revenue over the `Heuristic` solution. Recall that for each scenario (combination of parameters), we solve 20 instances, and for the instances where `FP+BB` does not converge to the optimal solution before the time limit is reached, we record the valid upper bound λ_u. In that case, we report in Table 6 the duality gap, denoted by Gap(%), which is calculated as:

$$\text{Gap}(\%) \quad = \quad \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} 100 \times \left(\frac{\lambda_u^p - \text{BEST}^p}{\lambda_u^p} \right),$$

where $\mathcal{P} \subseteq \{1, \ldots, 20\}$ is the subset of instances of the scenario where `FP+BB` search was stopped at the time limit, and λ_u^p is the final upper bound on λ when the time limit is reached.

While some of the important improvements depend on the interval $[\gamma^L,\ \gamma^U]$, we observe that in general, as the number n of products per nest increases, the overall improvement achieved by `FP+BB` decreases. Finally, after an extensive experimentation, we found out that if all $v_{i0} = 0$, the processing time for reaching optimality does not significantly grow with a large increase in the number of products. For that reason, we devised several instances with super large sizes and report our main findings in Table 7. Although we obtain a modest improvement in the revenue for the smallest range of γ (which is the closest scenario from the polynomial case where the heuristic is optimal) as we can see in Table 7, even with $n = 500$ and $n = 1{,}000$, `FP+BB` is able to find the optimal assortment in less than 2 seconds. As for instances with $n = 5{,}000$ products, we observe that the CPU time increases before the search converges to the best solution, however, the optimal solution is still found in around 70 seconds on average.

We conclude this section by summarizing the results of the numerical analysis in two plots shown in Figures 1 and 2 for small and large instances and super large instances. In both plots, the horizontal axis shows the interval of the dissimilarity parameter of nests. We observe that with an increase in the value of the dissimilarity parameter, the overall improvement of `FP+BB` over the value found by `Heuristic` consistently increases, regardless of the value of v_{i0} or the number of products within each nest. Finding the optimal assortment by using our proposed `FP+BB` can improve the expected revenue in average by up to 63% (obtained for $[\gamma^L, \gamma^U] = [1.5, 2.5]$ and $v_{i0} = 0, \forall i \in M$ in Table 2) compared to offering sorted-by-revenue assortments (`Heuristic`).

Table 2: Results on instances with $n = 10$

$v_{i0} = 0, \forall i \in M$	Heuristic		FP + BB						FP + F.Enum.
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.	Time
[0.5, 1.5]	0.30	1319.412	0.35	1455.784	20	5	**17.09**	4.3	0.65
[1.0, 2.0]	0.01	1499.060	0.02	2056.390	20	20	**39.38**	4.5	0.35
[1.5, 2.5]	0.02	1464.430	0.02	2278.883	20	20	**63.66**	6.9	0.53
[2.0, 3.0]	0.02	1470.562	0.02	2215.229	20	20	**60.01**	9.8	0.75
$v_{i0} > 0, \forall i \in M$	Heuristic		FP + BB						FP + F.Enum.
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.	Time
[0.5, 1.5]	0.01	1363.523	0.02	1420.608	20	19	**4.75**	10.9	0.85
[1.0, 2.0]	0.01	1372.336	0.02	1552.196	20	20	**15.26**	5.2	0.39
[1.5, 2.5]	0.01	1290.883	0.02	1542.003	20	20	**25.08**	6.7	0.50
[2.0, 3.0]	0.01	1484.537	0.02	1764.358	20	20	**22.74**	5.5	0.41
$v_{i0} \geqslant 0, \forall i \in M$	Heuristic		FP + BB						FP + F.Enum.
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.	Time
[0.5, 1.5]	0.01	1420.262	0.02	1542.811	20	12	**10.06**	6.8	0.51
[1.0, 2.0]	0.01	1602.591	0.02	1891.330	20	20	**24.22**	4.1	0.31
[1.5, 2.5]	0.01	1406.567	0.03	1801.331	20	20	**31.24**	5.8	0.45
[2.0, 3.0]	0.01	1466.853	0.02	1947.478	20	20	**40.16**	4.9	0.38

Table 3: Results on instances with $n = 25$

$v_{i0} = 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.25	2011.361	0.29	2038.585	20	3	**1.30**	3.9
[1.0, 2.0]	0.02	1874.942	0.02	2278.074	20	20	**24.55**	5.1
[1.5, 2.5]	0.02	1941.861	0.02	2432.010	20	20	**27.50**	5.2
[2.0, 3.0]	0.02	1881.559	0.02	2493.864	20	20	**33.64**	8.1
$v_{i0} > 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.02	1631.454	0.05	1673.522	20	17	**2.63**	15.2
[1.0, 2.0]	0.02	1699.821	0.03	1810.193	20	20	**8.01**	6.2
[1.5, 2.5]	0.02	1743.259	0.03	1845.962	20	20	**5.99**	6.0
[2.0, 3.0]	0.02	1769.376	0.03	1960.964	20	20	**12.03**	5.4
$v_{i0} \geqslant 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.02	1772.294	0.04	1814.347	20	11	**3.04**	14.4
[1.0, 2.0]	0.02	1820.927	0.02	1965.316	20	20	**8.28**	5.2
[1.5, 2.5]	0.02	1826.060	0.02	2078.122	20	20	**15.27**	4.9
[2.0, 3.0]	0.02	1827.334	0.02	2106.624	20	20	**16.90**	4.8

Table 4: Results on instances with $n = 50$

$v_{i0} = 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.02	2102.011	0.03	2154.903	20	4	**2.74**	6.8
[1.0, 2.0]	0.03	2271.554	0.03	2439.062	20	20	**7.90**	4.2
[1.5, 2.5]	0.02	2318.757	0.02	2572.765	20	20	**11.35**	5.0
[2.0, 3.0]	0.02	2195.411	0.02	2579.245	20	20	**18.30**	6.1
$v_{i0} > 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.02	1859.730	0.24	1870.572	20	15	**0.56**	22.6
[1.0, 2.0]	0.02	1932.296	0.04	1957.362	20	19	**1.34**	8.6
[1.5, 2.5]	0.02	1968.766	0.03	2014.973	20	20	**2.51**	6.2
[2.0, 3.0]	0.02	1945.271	0.04	2022.854	20	20	**4.10**	6.5
$v_{i0} \geqslant 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.02	2055.472	0.26	2067.200	20	12	**0.61**	20.3
[1.0, 2.0]	0.02	2037.008	0.04	2137.010	20	20	**5.03**	7.5
[1.5, 2.5]	0.02	2113.793	0.03	2204.823	20	20	**4.32**	5.5
[2.0, 3.0]	0.02	2144.774	0.04	2279.204	20	20	**6.80**	5.7

Table 5: Results on instances with $n = 100$

$v_{i0} = 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.03	2347.244	0.04	2363.281	20	9	**0.72**	8.0
[1.0, 2.0]	0.03	2449.480	0.04	2542.060	20	18	**3.98**	8.1
[1.5, 2.5]	0.03	2393.254	0.07	2623.670	20	20	**9.92**	5.5
[2.0, 3.0]	0.03	2459.432	0.04	2669.409	20	20	**8.66**	5.8
$v_{i0} > 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.03	2040.691	2.40	2042.736	20	9	**0.10**	35.0
[1.0, 2.0]	0.03	2074.101	0.11	2087.427	20	18	**0.66**	11.6
[1.5, 2.5]	0.03	2095.977	0.07	2122.436	20	20	**1.26**	7.0
[2.0, 3.0]	0.03	2099.439	0.07	2141.169	20	20	**2.09**	7.0
$v_{i0} \geqslant 0, \forall i \in M$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.03	2020.733	3.16	2026.195	20	11	**0.26**	25.8
[1.0, 2.0]	0.03	2084.302	0.10	2095.638	20	19	**0.54**	12.9
[1.5, 2.5]	0.03	2105.198	0.06	2127.319	20	20	**1.05**	6.8
[2.0, 3.0]	0.03	2114.391	0.08	2144.870	20	20	**1.49**	7.7

Table 6: Results on instances with $n = 250$

$v_{i0} = 0, \forall i \in M$	Heuristic		FP + BB						
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	Gap(%)	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.05	2467.701	0.22	2479.887	20	0.00	6	**0.53**	22.9
[1.0, 2.0]	0.05	2618.941	0.08	2656.178	20	0.00	19	**1.44**	7.6
[1.5, 2.5]	0.05	2629.566	0.08	2721.555	20	0.00	20	**3.58**	5.2
[2.0, 3.0]	0.05	2630.360	0.08	2749.843	20	0.00	20	**4.60**	5.5
$v_{i0} > 0, \forall i \in M$	Heuristic		FP + BB						
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	Gap(%)	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.06	2203.200	1855.32	2203.200	11	7.72	5	0.00	22.9
[1.0, 2.0]	0.05	2240.957	0.60	2246.826	20	0.00	17	**0.27**	14.8
[1.5, 2.5]	0.05	2251.769	0.31	2268.824	20	0.00	20	**0.76**	8.4
[2.0, 3.0]	0.05	2301.263	0.34	2315.573	20	0.00	20	**0.62**	7.1
$v_{i0} \geqslant 0, \forall i \in M$	Heuristic		FP + BB						
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	Gap(%)	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.06	2245.679	1463.56	2245.960	13	2.58	5	**0.01**	23.1
[1.0, 2.0]	0.06	2254.461	0.58	2262.545	20	0.00	16	**0.35**	15.0
[1.5, 2.5]	0.05	2359.736	0.83	2385.752	20	0.00	20	**1.13**	6.7
[2.0, 3.0]	0.05	2360.125	0.75	2403.292	20	0.00	20	**1.86**	6.2

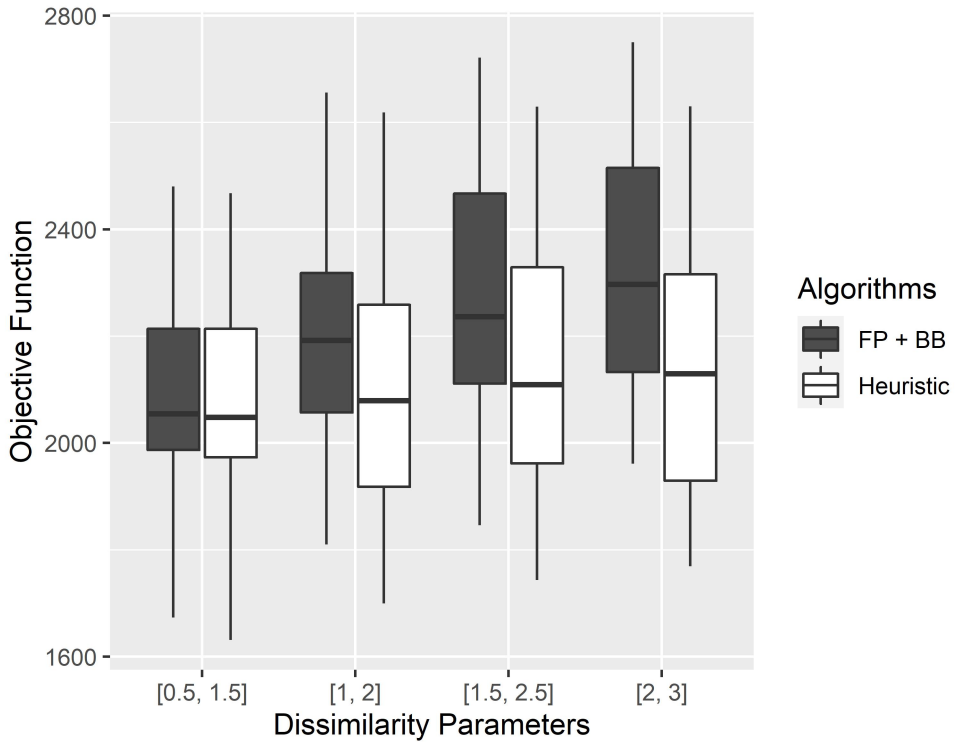

Figure 1: Expected revenues found by optimal (or best found) assortments (`FP+BB`) versus `Heuristic`, for a set of small and large instances ($n \in \{25,\ 50,\ 100,\ 250\}$)

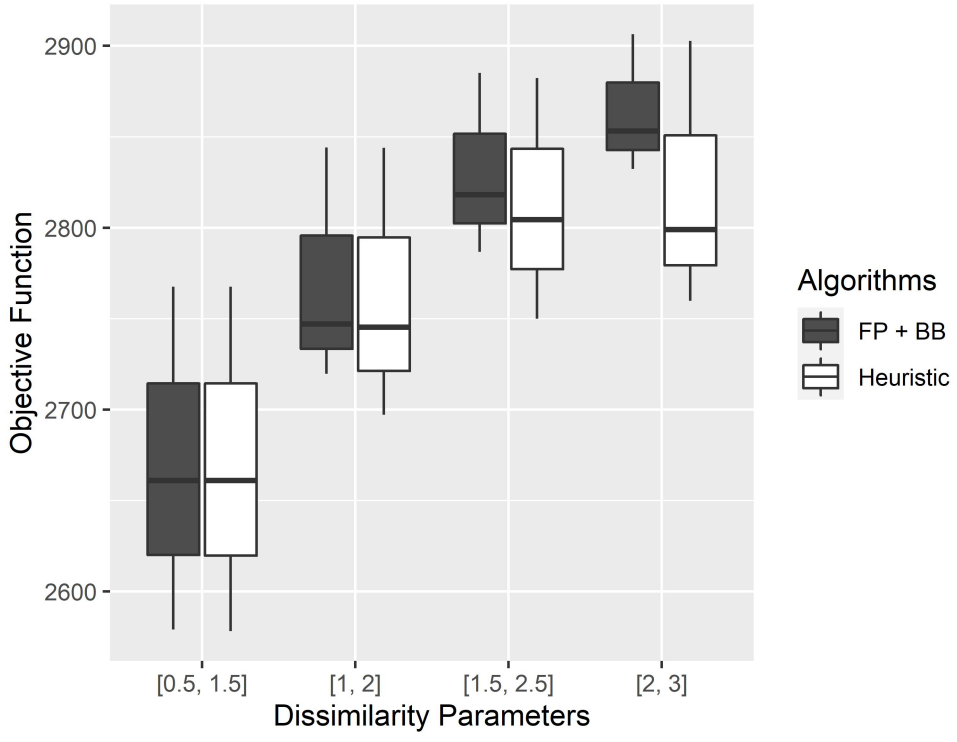

Figure 2: Expected revenues found by optimal assortments (`FP+BB`) versus `Heuristic`, for a set of super large instances ($n \in \{500,\ 1000,\ 5000\}$)

Table 7: Results on super large instances with $v_{i0} = 0, \forall i \in M$

$n = 500$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.10	2578.208	0.49	2579.121	20	5	**0.03**	26.3
[1.0, 2.0]	0.11	2697.146	0.22	2719.671	20	20	**0.84**	9.2
[1.5, 2.5]	0.10	2749.865	0.18	2786.700	20	19	**1.36**	6.3
[2.0, 3.0]	0.11	2759.730	0.16	2832.230	20	20	**2.67**	5.2
$n = 1{,}000$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	0.21	2661.075	1.49	2661.075	20	2	0.00	35.0
[1.0, 2.0]	0.21	2745.328	1.07	2747.184	20	10	**0.07**	28.2
[1.5, 2.5]	0.22	2804.493	0.81	2818.214	20	17	**0.49**	13.7
[2.0, 3.0]	0.22	2799.017	0.35	2853.180	20	20	**1.96**	5.6
$n = 5{,}000$	Heuristic		FP + BB					
$[\gamma^L, \gamma^U]$	Time	Obj.	Time	Obj.	# Opt.	# Imp.	Impr.(%)	# Iter.
[0.5, 1.5]	2.14	2767.521	14.17	2767.523	20	7	0.00	36.7
[1.0, 2.0]	2.12	2843.984	70.97	2844.074	20	9	0.00	30.6
[1.5, 2.5]	2.04	2882.258	10.53	2885.153	20	15	**0.10**	15.5
[2.0, 3.0]	1.98	2902.711	14.23	2906.345	20	20	**0.13**	10.2

5. Conclusion

In this paper, we studied the assortment problem when customer choice behavior is modeled using a nested logit model. We proposed an exact method combining fractional programming and a tailored branch-and bound algorithm for the non-linear parametrized subproblem associated with each nest, which we proved to be NP-hard by a non-trivial reduction from the Subset-Sum problem. Our method enables to find an optimal assortment regardless of the value of the dissimilarity parameter or the attractiveness of the no-purchase option within a nest, i.e. any mix of the values of these parameters can be handled by the algorithm. Computational results reveal that finding the optimal assortment can have a significant impact on the expected revenue compared to the benchmark heuristic in the literature (Davis et al. [9]), which is based on sorted-by-revenue assortments (this heuristic being optimal in one particular case). The highest revenue improvement (63% on average) of the optimal assortment vs the heuristic solution was obtained for problems with 10 products per nest and dissimilarity parameters between 1.5 and 2.5. As the number of products increases, the improvement made was modest though (at most 3% for our large instances). However, for some instances without the no-purchase option, our exact method could solve to optimality problems with up to 5,000 products in less than one minute and half, and was generally fast for all scenarios tested. Our method also enables to assess the empirical performance of the sorted-by-revenue heuristic which can be a good compromise for a huge number of products and dissimilarity parameters higher than but close to one.

An exciting direction for future research could be to modify our method to solve problems with capacity or cardinality constraints. An AOPNL with both capacity and cardinality constraint for each particular nest in which all dissimilarity parameters are at most one has been studied in the research of Gallego and Topaloglu [15] and Désir et al. [11]. Developing an efficient exact method for a more general case in which some dissimilarity parameters are higher than one remains open.

References

[1] C. Archetti, G. Desaulniers, and M. G. Speranza. Minimizing the logistic ratio in the inventory routing problem. *EURO Journal on Transportation and Logistics*, 6(4):289–306, 2017.

[2] M. E. Ben-Akiva and S. R. Lerman. *Discrete choice analysis: theory and application to travel demand*, volume 9. MIT press, 1985.

[3] D. Bertsimas and P. Vayanos. Data-driven learning in dynamic pricing using adaptive optimization. *Preprint*, 2015.

[4] A. Billionnet. Optimal selection of forest patches using integer and fractional programming. *Operational Research An International Journal*, 10:1–26, 2010.

[5] A. Billionnet. Mathematical optimization ideas for biodiversity conservation. *European Journal of Operational Research*, 231(3):514 –534, 2013.

[6] A. Börsch-Supan. On the compatibility of nested logit models with utility maximization. *Journal of Econometrics*, 43(3):373–388, 1990.

[7] R. Chen and H. Jiang. Capacitated assortment and price optimization under the multilevel nested logit model. *Operations Research Letters*, 47(1):30–35, 2019.

[8] X. Chen, Y. Wang, and Y. Zhou. Dynamic assortment selection under the nested logit models. *arXiv preprint arXiv:1806.10410*, 2018.

[9] J. M. Davis, G. Gallego, and H. Topaloglu. Assortment optimization under variants of the nested logit model. *Operations Research*, 62(2):250–273, 2014.

[10] J. M. Davis, H. Topaloglu, and D. P. Williamson. Pricing problems under the nested logit model with a quality consistency constraint. *INFORMS Journal on Computing*, 29 (1):54–76, 2016.

[11] A. Désir, V. Goyal, and J. Zhang. Near-optimal algorithms for capacity constrained assortment optimization. *Submitted to Operations Research, Available at SSRN 2543309*, 2014.

[12] W. Dinkelbach. On nonlinear fractional programming. *Management science*, 13(7): 492–498, 1967.

[13] J. B. Feldman and H. Topaloglu. Capacity constraints across nests in assortment optimization under the nested logit model. *Operations Research*, 63(4):812–822, 2015.

[14] A. Flores, G. Berbeglia, and P. Van Hentenryck. Assortment optimization under the sequential multinomial logit model. *European Journal of Operational Research*, 273(3): 1052–1064, 2019.

[15] G. Gallego and H. Topaloglu. Constrained assortment optimization for the nested logit model. *Management Science*, 60(10):2583–2601, 2014.

[16] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[17] V. Goyal, R. Levi, and D. Segev. Near-optimal algorithms for the assortment planning problem under dynamic substitution and stochastic demand. *Operations Research*, 64(1): 219–235, 2016.

[18] G. W. Klau, I. Ljubić, P. Mutzel, U. Pferschy, and R. Weiskircher. The fractional prize-collecting Steiner tree problem on trees. In G. D. Battista and U. Zwick, editors, *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, volume 2832 of *Lecture Notes in Computer Science*, pages 691–702. Springer, 2003.

[19] A. G. Kök, M. L. Fisher, and R. Vaidyanathan. Assortment planning: Review of literature and industry practice. In *Retail supply chain management*, pages 99–153. Springer, 2008.

[20] B. Lee. Calling patterns and usage of residential toll service under self selecting tariffs. *Journal of Regulatory economics*, 16(1):45–82, 1999.

[21] G. Li and P. Rusmevichientong. A greedy algorithm for the two-level nested logit model. *Operations Research Letters*, 42(5):319–324, 2014.

[22] G. Li, P. Rusmevichientong, and H. Topaloglu. The d-level nested logit model: Assortment and price optimization problems. *Operations Research*, 63(2):325–342, 2015.

[23] R. D. Luce. *Individual choice behavior: A theoretical analysis*. Wiley, Republicated by Dover Publications, 2012, 1959.

[24] T. Mai and A. Lodi. An algorithm for assortment optimization under parametric discrete choice models. *Available at SSRN 3370776*, 2019.

[25] D. McFadden. Econometric models for probabilistic choice among products. *Journal of Business*, pages S13–S29, 1980.

[26] D. McFadden et al. Conditional logit analysis of qualitative choice behavior. *Frontiers in Econometrics, ed. by P. Zarembka, New York: Academic Press*, pages 105–142, 1973.

[27] N. Megiddo. Combinatorial optimization with rational objective functions. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 1–12. ACM, 1978.

[28] P. Rusmevichientong and H. Topaloglu. Robust assortment optimization in revenue management under the multinomial logit choice model. *Operations Research*, 60(4): 865–882, 2012.

[29] D. Segev. Approximation schemes for capacity-constrained assortment optimization under the nested logit model. *Available at SSRN 3553264*, 2020.

[30] S. Sethuraman and S. Butenko. The maximum ratio clique problem. *Computational Management Science*, 12:197–218, 2015.

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

[32] P. Tiwari and H. Hasegawa. Demand for housing in Tokyo: A discrete choice analysis. *Regional Studies*, 38(1):27–42, 2004.

[33] K. E. Train, D. L. McFadden, and M. Ben-Akiva. The demand for local telephone service: A fully discrete model of residential calling patterns and service choices. *The RAND Journal of Economics*, pages 109–123, 1987.

[34] K. E. Train, M. Ben-Akiva, and T. Atherton. Consumption patterns and self-selecting tariffs. *The Review of Economics and Statistics*, pages 62–73, 1989.

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

[36] J. Yates and D. F. Mackay. Discrete choice modelling of urban housing markets: A critical review and an application. *Urban studies*, 43(3):559–581, 2006.

Appendix A. Proof of Lemmas and Propositions

Appendix A.1. Proof of Proposition 2.

Since the parameterized problem has a piece-wise linear structure, each piece of line corresponding to an i has a unique slope. On the other hand, all assortments corresponding to that piece of line are equal. As a result, if $S_l(\lambda_l) = S_u(\lambda_u)$, it means that both $S_l(\lambda_l)$ and $S_u(\lambda_u)$ belong to the same piece of line and since $F_{par}(\lambda_l) > 0$ and $F_{par}(\lambda_u) < 0$, therefore, $S_l(\lambda_l) = S_u(\lambda_u)$, then $S^* = S_l(\lambda_l) = S_u(\lambda_u)$. □

Appendix A.2. Proof of Proposition 4.

i) if $\mathcal{H} = 0$, it means that by adding all the products whose revenue is greater than or equal to λ, the second part on the right-hand-side of the objective function (11) will be zero and therefore, the value of the function (11) will be zero. This means that the lower bound is zero and removing any of these products will result in a negative value of the objective function therefore $x_k = 1$ for $k \in K_1^0$ where $K_1^0 = \{k \in K : r_k \geqslant \lambda\}$. On the other hand, since the lower bound is zero, adding a product with $r_k < \lambda$ will turn the second part of the right-hand-side of the objective function (11) negative, which in turn results in a negative value. Therefore, $x_k^* = 0$ for $k \in K_0^0 = \{k \in K \setminus K_1^0 : r_k < \lambda\}$.

ii) if $\mathcal{H} > 0$, then there exists a subset of products with $r_k \geqslant \lambda$ that if offered in the assortment of nest i, so the value of the objective function (14) will be positive. As a result, offering any product from $K_0^0 = \{k \in K : r_k < \lambda\}$ will reduce the numerator of the formulation (14) and at the same time increase the value of its denominator which in turn decreases the value of the fraction. Therefore, $x_k^* = 0$ for $k \in K_0^0$. Now that we are left with only products with revenue $r_k > \lambda$, we define $K' = N \setminus K_0^0$, $r_{max} = \max_{j \in N}\{r_j\}$ and calculate the partial derivative of function (11). We have:

$$
\begin{aligned}
\frac{\partial f^0(x)}{\partial x_k} &= (\gamma-1)\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-2} v_k\left(\sum_{j\in K'} v_j(r_j-\lambda)x_j-\lambda v_0\right)+\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-1} v_k(r_k-\lambda)\\
&= v_k\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{\sum_{j\in K'} v_j(r_j-\lambda)x_j-\lambda v_0}{v_0+\sum_{j\in K'} v_j x_j}+(r_k-\lambda)\right)\\
&= v_k\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{\sum_{j\in K'} v_j(r_j-\lambda)x_j-\lambda v_0+v_0(r_{max}-\lambda)-v_0(r_{max}-\lambda)}{v_0+\sum_{j\in K'} v_j x_j}\right.\\
&\qquad \left.+(r_k-\lambda)\right)\\
&\geqslant v_k\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{(r_{max}-\lambda)\sum_{j\in K'} v_j x_j+v_0(r_{max}-\lambda)-\lambda v_0-v_0(r_{max}-\lambda)}{v_0+\sum_{j\in K'} v_j x_j}\right.\\
&\qquad \left.+(r_k-\lambda)\right)\\
&\quad \text{since} \qquad \gamma-1<0\\
&= v_k\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left((r_{max}-\lambda)-\frac{v_0 r_{max}}{v_0+\sum_{j\in K'} v_j x_j}\right)+(r_k-\lambda)\right)\\
&\geqslant v_k\left(v_0+\sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left((r_{max}-\lambda)-\frac{v_0 r_{max}}{v_0+\sum_{j\in K'} v_j}\right)+(r_k-\lambda)\right)\\
&\quad \text{since} \qquad r_{max}-\lambda>0\\
&\geqslant 0 \qquad \text{if} \qquad r_k\geqslant \lambda+(1-\gamma)\left(r_{max}-\lambda-\frac{v_0 r_{max}}{v_0+\sum_{j\in K'} v_j}\right)
\end{aligned}
$$

The function is increasing in x_k for $k\in K_1^0=\{k\in N\setminus K_0^0 : r_k\geqslant \lambda+(1-\gamma)(r_{max}-\lambda-(v_0 r_{max})/(v_0+\sum_{j\in K'} v_j))\}$, then for these products k we have $x_k^*=1$.

iii) if $\mathcal{H}<0$, in this case, there exists no positive value for the objective function. In other words, for any vector x, $f^0(x)<0$. In this case, since the value of the fraction on the right-hand-side of (14) is always negative, adding products with $r_k\geqslant\lambda$ will always make the numerator less negative and at the same time, increases the value of the denominator which in turn results in an overall less negative value of the objective function (14). Therefore, define $K_1^0=\{k\in N : r_k\geqslant\lambda\}$ then $x_k^*=1, \forall k\in K_1^0$.

Defining $C_1^0=\sum_{k\in K_1^0} v_k(r_k-\lambda)-\lambda v_0$, $V_1^0=\sum_{k\in K_1^0} v_k+v_0$ and $K'=N\setminus K_1^0$, we can update the objective function (11) with the remaining of the variables as:

$$\max_{x\in\{0,1\}^{|K'|}} f^0(x) = \left(V_1^0 + \sum_{k\in K'} v_k x_k\right)^{\gamma-1} \times \left(C_1^0 + \sum_{k\in K'} v_k(r_k-\lambda)x_k\right)$$

We define $r_{min} = \min_{j\in K'}\{r_j\}$ and $\Delta = C_1^0 - V_1^0(r_{min}-\lambda)$, calculating the partial derivative of the function above with respect of variable x_k, we have:

$$\begin{aligned}
\frac{\partial f^0(x)}{\partial x_k} &= (\gamma-1)\left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-2} v_k \left(C_1^0 + \sum_{j\in K'} v_j(r_j-\lambda)x_j\right) + \left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-1} v_k(r_k-\lambda) \\
&= v_k\left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{C_1^0 + \sum_{j\in K'} v_j(r_j-\lambda)x_j}{V_1^0 + \sum_{j\in K'} v_j x_j} + (r_k-\lambda)\right) \\
&\leqslant v_k\left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{C_1^0 + (r_{min}-\lambda)\sum_{j\in K'} v_j x_j}{V_1^0 + \sum_{j\in K'} v_j x_j} + (r_k-\lambda)\right) \\
&= v_k\left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{(r_{min}-\lambda)\sum_{j\in K'} v_j x_j + V_1^0(r_{min}-\lambda) + C_1^0 - V_1^0(r_{min}-\lambda)}{V_1^0 + \sum_{j\in K'} v_j x_j}\right. \\
&\qquad \left. + (r_k-\lambda)\right) \\
&= v_k\left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left((r_{min}-\lambda) + \frac{C_1^0 - V_1^0(r_{min}-\lambda)}{V_1^0 + \sum_{j\in K'} v_j x_j}\right) + (r_k-\lambda)\right)(\star)
\end{aligned}$$

We now have two cases. If $\Delta \geqslant 0$, we can transform the expression denoted by $(\star)$ as:

$$\begin{aligned}
&\leqslant v_k\left(V_1^0 + \sum_{j\in K'} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left((r_{min}-\lambda) + \frac{C_1^0 - V_1^0(r_{min}-\lambda)}{V_1^0 + \sum_{j\in K'} v_j}\right) + (r_k-\lambda)\right) \\
&\quad \text{since} \qquad \gamma - 1 < 0 \\
&\leqslant 0 \qquad \text{if} \qquad r_k \leqslant \lambda + (1-\gamma)\left(\frac{C_1^0 + (r_{min}-\lambda)\sum_{j\in K'} v_j}{V_1^0 + \sum_{j\in K'} v_j}\right)
\end{aligned}$$

On the other hand, if $\Delta < 0$, we transform $(\star)$ as follows:

$$
\begin{aligned}
&\leqslant \quad v_k \left(V_1^0 + \sum_{j \in K'} v_j x_j \right)^{\gamma - 1} \left((\gamma - 1) \left((r_{min} - \lambda) + \frac{C_1^0 - V_1^0 \, (r_{min} - \lambda)}{V_1^0} \right) + (r_k - \lambda) \right) \\
&\qquad \text{since} \qquad \gamma - 1 < 0 \\
&\leqslant \quad 0 \qquad \text{if} \qquad r_k \leqslant \lambda + (1 - \gamma) \left(\frac{C_1^0}{V_1^0} \right)
\end{aligned}
$$

Therefore, if $\Delta \geqslant 0$, $x_k^* = 0$ for $k \in K_0^0 = \{k \in K' : r_k \leqslant \lambda + (1 - \gamma) \left(\frac{C_1^0 + (r_{min} - \lambda) \sum_{j \in K'} v_j}{V_1^0 + \sum_{j \in K'} v_j} \right)$, and if $\Delta < 0$, then $x_k^* = 0$ for $k \in K_0^0 = \{k \in K' : r_k \leqslant \lambda + (1 - \gamma) \, (C_1^0 / V_1^0)$. □

Appendix A.3. Proof of Proposition 5.

i) We set $r_{max} = \max_{j \in \bar{K}^{t-1}} \{r_j\}$. Calculating the partial derivative of the function (13), we have:

$$
\begin{aligned}
\frac{\partial f^t(x)}{\partial x_k} &= (\gamma-1)\left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-2} v_k\left(C_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j(r_j-\lambda)x_j\right) \\
&\quad + \left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-1} v_k(r_k-\lambda) \\
&= v_k\left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\frac{C_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j(r_j-\lambda)x_j}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j}+(r_k-\lambda)\right) \\
&= v_k\left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left(\frac{C_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j(r_j-\lambda)x_j+V_1^{t-1}(r_{max}-\lambda)}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j}\right.\right. \\
&\quad \left.\left.-\frac{V_1^{t-1}(r_{max}-\lambda)}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j}\right)+(r_k-\lambda)\right) \\
&= v_k\left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left(\frac{C_1^{t-1}+(r_{max}-\lambda)\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j+V_1^{t-1}(r_{max}-\lambda)}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j}\right.\right. \\
&\quad \left.\left.-\frac{V_1^{t-1}(r_{max}-\lambda)}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j}\right)+(r_k-\lambda)\right) \\
&= v_k\left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left((r_{max}-\lambda)+\frac{C_1^{t-1}-V_1^{t-1}(r_{max}-\lambda)}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j x_j}\right)+(r_k-\lambda)\right) \\
&\geqslant v_k\left(V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j x_j\right)^{\gamma-1}\left((\gamma-1)\left((r_{max}-\lambda)+\frac{C_1^{t-1}-V_1^{t-1}(r_{max}-\lambda)}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j}\right)+(r_k-\lambda)\right) \\
&\geqslant 0 \qquad \text{if} \qquad r_k \geqslant \lambda+(1-\gamma)\left(\frac{C_1^{t-1}+(r_{max}-\lambda)\sum\limits_{j\in\bar{K}^{t-1}} v_j}{V_1^{t-1}+\sum\limits_{j\in\bar{K}^{t-1}} v_j}\right)
\end{aligned}
$$

Therefore, for $k\in K_1^t=\{k\in\bar{K}^{t-1}: r_k\geqslant\lambda+(1-\gamma)(\frac{C_1^{t-1}+(r_{max}-\lambda)\sum_{j\in\bar{K}^{t-1}} v_j}{V_1^{t-1}+\sum_{j\in\bar{K}^{t-1}} v_j})$, $x_k^*=1$.

ii) The proof of this section is similar to that section *iii* in the proof of Proposition 4. □

Appendix A.4. Proof of Proposition 6.

In this case, we can rewrite the objective function (15) as:

$$
\begin{aligned}
f^t(x) &= (V_1^t)^{\gamma-1} C_1^t \left(1 + \frac{\sum\limits_{k\in\bar{K}^t} v_k x_k}{V_1^t}\right)^{\gamma-1} \times \left(1 + \frac{\sum\limits_{k\in\bar{K}^t} v_k (r_k - \lambda) x_k}{C_1^t}\right) \\
&= (V_1^t)^{\gamma-1} C_1^t \exp\left(\ln\left(\left(1 + \frac{\sum\limits_{k\in\bar{K}^t} v_k x_k}{V_1^t}\right)^{(\gamma-1)} \times \left(1 + \frac{\sum\limits_{k\in\bar{K}^t} v_k (r_k - \lambda) x_k}{C_1^t}\right)\right)\right) \\
&= (V_1^t)^{\gamma-1} C_1^t \exp\left((\gamma-1)\ln\left(1 + \frac{\sum\limits_{k\in\bar{K}^t} v_k x_k}{V_1^t}\right) + \ln\left(1 + \frac{\sum\limits_{k\in\bar{K}^t} v_k (r_k - \lambda) x_k}{C_1^t}\right)\right) \\
&\leqslant (V_1^t)^{\gamma-1} C_1^t \exp\left((\gamma-1)\frac{2\sum\limits_{k\in\bar{K}^t} v_k x_k}{2V_1^t + \sum\limits_{k\in\bar{K}^t} v_k x_k} + \frac{\sum\limits_{k\in\bar{K}^t} v_k (r_k - \lambda) x_k}{C_1^t}\right) \\
&\quad \text{since } \frac{2\tau}{2+\tau} \leqslant \ln(1+\tau), \ln(1+\tau) \leqslant \tau \text{ for } \tau \geqslant 0, r_k > \lambda, \forall k \in \bar{K}^t \text{ and } \gamma - 1 < 0 \\
&\leqslant (V_1^t)^{\gamma-1} C_1^t \exp\left((\gamma-1)\frac{2\sum\limits_{k\in\bar{K}^t} v_k x_k}{2V_1^t + \sum\limits_{k\in\bar{K}^t} v_k} + \frac{\sum\limits_{k\in\bar{K}^t} v_k (r_k - \lambda) x_k}{C_1^t}\right) \quad \text{since } \sum_{k\in\bar{K}^t} v_k x_k \leqslant \sum_{k\in\bar{K}^t} v_k \\
&= (V_1^t)^{\gamma-1} C_1^t \exp\left(\sum_{k\in\bar{K}^t} v_k \left(\frac{2(\gamma-1)}{2V_1^t + \sum\limits_{k\in\bar{K}^t} v_k} + \frac{(r_k - \lambda)}{C_1^t}\right) x_k\right)
\end{aligned}
$$

We note the coefficients inside the above exponential function as

$$
w_k^t = v_k \left(\frac{2(\gamma-1)}{2V_1^t + \sum\limits_{k\in\bar{K}^t} v_k} + \frac{(r_k - \lambda)}{C_1^t}\right)
$$

Therefore, we get an upper bound on (15) by solving

$$
\max_{x\in\{0,1\}^{|\bar{K}^t|}} \sum_{k\in\bar{K}^t} w_k^t x_k \tag{A.1}
$$

Problem (A.1) is easily solved by offering all products with $w_k^t > 0$. Therefore, the solution of problem (A.1) is:

$$
z_1^*(t) = \sum_{k\in\bar{K}^t} \max(0, w_k^t)
$$

We finally obtain the upper bound on $f_t(x)$:

$$
f_t(x) \leqslant (V_1^t)^{\gamma-1}\ C_1^t\ e^{z_1^*(t)}
$$

□

Appendix A.5. Proof of Proposition 7.

i) the proof of this part is similar to that of Proposition 4.

ii) The first part holds because for any S where $k \notin S$, the objective of $S \cup \{k\}$ is larger than that of S if $r_k - \lambda \geqslant 0$. We can then include in the assortment all variables k such that $r_k \geqslant \lambda$. We also calculate $V_1^t = \sum_{k \in K_1^t} v_k + v_0$ and $C_1^t = \sum_{k \in K_1^t} v_k(r_k - \lambda) - \lambda v_0$ and define $K' = N \setminus K_1^0$. The subproblem then can be expressed as:

$$\max_{x \in \{0,1\}^{|K'|}} f^0(x) = \left(V_1^0 + \sum_{k \in K'} v_k x_k\right)^{\gamma - 1} \times \left(C_1^0 + \sum_{k \in K'} v_k (r_k - \lambda) x_k\right) \tag{A.2}$$

We compute the partial derivative of f in above function with respect to variable x_k:

$$\begin{aligned}
\frac{\partial f^0(x)}{\partial x_k} &= (\gamma - 1)\left(V_1^0 + \sum_{j \in K'} v_j x_j\right)^{\gamma - 2} v_k \left(C_1^0 + \sum_{j \in K'} v_j (r_j - \lambda) x_j\right) + \left(V_1^0 + \sum_{j \in K'} v_j x_j\right)^{\gamma - 1} v_k (r_k - \lambda) \\
&= v_k \left(V_1^0 + \sum_{j \in K'} v_j x_j\right)^{\gamma - 1} \left((\gamma - 1) \frac{C_1^0 + \sum_{j \in K'} v_j (r_j - \lambda) x_j}{V_1^0 + \sum_{j \in K'} v_j x_j} + (r_k - \lambda)\right) \\
&\leqslant v_k \left(V_1^0 + \sum_{j \in K'} v_j x_j\right)^{\gamma - 1} \left((\gamma - 1) \frac{C_1^0}{V_1^0} + (r_k - \lambda)\right) \qquad \text{as } \sum_{j \in K'} v_j (r_j - \lambda) < 0 \\
&= v_k \left(V_1^0 + \sum_{j \in K'} v_j x_j\right)^{\gamma - 1} \Big((\gamma - 1)(C_1^0 / V_1^0) + (r_k - \lambda)\Big) \\
&\leqslant 0 \qquad \text{if} \qquad r_k \leqslant \lambda - (\gamma - 1)(C_1^0 / V_1^0)
\end{aligned}$$

Therefore, if we define $K_0^0 = \{k \in N \setminus K_1^0 : r_k < \lambda - (\gamma - 1)(C_1^0 / V_1^0)\}$, then $x_k^* = 0$ for $k \in K_0^0$.

iii) if $\mathcal{H} < 0$, similar to the case when $\gamma \leqslant 1$, there exists no positive value for the objective function. Therefore, since the value of the objective function 11 is always negative, adding products with $r_k < \lambda$ will always result in a lower value. □

Appendix A.6. Proof of Proposition 8.

The proof of this proposition if similar to that of Proposition 7. □

Appendix A.7. Proof of Proposition 9.

We start reformulating $f^t(x)$ as in the third equality of the proof of Proposition 6:

$$
\begin{aligned}
f^t(x) &= (V_1^t)^{\gamma-1} C_1^t \exp\left((\gamma-1)\ln\left(1+\frac{\sum\limits_{k\in\bar{K}^t} v_k x_k}{V_1^t}\right)+\ln\left(1+\frac{\sum\limits_{k\in\bar{K}^t} v_k(r_k-\lambda)x_k}{C_1^t}\right)\right) \\
&\leqslant (V_1^t)^{\gamma-1} C_1^t \exp\left((\gamma-1)\frac{\sum\limits_{k\in\bar{K}^t} v_k x_k}{V_1^t}+\frac{\sum\limits_{k\in\bar{K}^t} v_k(r_k-\lambda)x_k}{C_1^t}\right) \\
&\quad \text{as } \ln(1+\tau)\leqslant\tau \text{ and } r_k\leqslant\lambda \text{ for } k\in\bar{K} \\
&= (V_1^t)^{\gamma-1} C_1^t \exp\left(\sum_{k\in\bar{K}^t} v_k\left(\frac{\gamma-1}{V_1^t}+\frac{r_k-\lambda}{C_1^t}\right)x_k\right)
\end{aligned}
$$

Let us note the coefficients inside the above exponential function as

$$
w_k^t = v_k\left(\frac{\gamma-1}{V_1^t}+\frac{r_k-\lambda}{C_1^t}\right)
$$

At the root node ($t=0$) and for $k\in\bar{K}$, $r_k \geqslant \lambda-(\gamma-1)(C_1^t/V_1^t)$, then all coefficients w_k^0 are non-negative. However, at an arbitrary B&B node t, some k with $r_k-\lambda<0$ might have been added in V_1^t in the previous child nodes. Hence, coefficients w_k^t can be positive or negative. Then the solution to

$$
\max_{x\in\{0,1\}^{|\bar{K}^t|}} \sum_{k\in\bar{K}^t} w_k^t x_k
$$

can be obtained by having

$$
z_2^*(t) = \sum_{k\in\bar{K}^t} \max(0, w_k^t)
$$

Finally, we obtain the upper bound on $f_t(x)$:

$$
f_t(x) \leqslant (V_1^t)^{\gamma-1}\ C_1^t\ e^{z_2^*(t)}
$$

□

Appendix B. Pseudo-code of the preprocessing algorithms

We summarize the findings of this section in Algorithms (2) – (5).

Appendix B.1.

Algorithm 2: Preprocessing for $t = 0$ when $\gamma \leqslant 1$

$N \leftarrow$ set of all products ;
$\mathcal{H} \leftarrow \sum_{k \in N : r_k \geqslant \lambda} v_k(r_k - \lambda) - \lambda v_0$;
if $\mathcal{H} = 0$ **then** // Case 0
  $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$;
  Fix $x_k^* = 1$ for $k \in K_1^0$;
  $K_0^0 = \{k \in N \setminus K_1^0 : r_k < \lambda\}$;
  Fix $x_k^* = 0$ for $k \in K_0^0$;
  Exit (subproblem is solved) ;
else if $\mathcal{H} > 0$ **then**// Case 1
  $K_0^0 = \{k \in N : r_k < \lambda\}$;
  $x_k^* = 0$ for $k \in K_0^0$;
  $r_{max} = \max_{j \in N}\{r_j\}$;
  $K' = N \setminus K_0^0$;
  $K_1^0 = \{k \in K' : r_k \geqslant \lambda + (1-\gamma)(r_{max} - \lambda - (v_0 r_{max})/(v_0 + \sum_{j \in K'} v_j))\}$;
  $x_k^* = 1, \forall k \in K_1^0$;
  Continue with upper bound $(V_1^t)^{\gamma-1} C_1^t e^{z_1^*(t)}$;
else if $\mathcal{H} < 0$ **then** // Case 2
  $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$;
  $x_k^* = 1, \forall k \in K_1^0$;
  $C_1^0 = \sum_{k \in K_1^0} v_k(r_k - \lambda) - \lambda v_0$;
  $V_1^0 = v_0 + \sum_{k \in K_1^0} v_k$;
  $K' = N \setminus K_1^0$;
  $r_{min} = \min_{j \in K'}\{r_j\}$;
  $\Delta = C_1^0 - V_1^0(r_{min} - \lambda)$;
  if $\Delta \geqslant 0$ **then**
    $K_0^0 = \{k \in K' : r_k \leqslant \lambda + (1-\gamma)((C_1^0 + (r_{min} - \lambda)\sum_{j \in K'} v_j)/(V_1^0 + \sum_{j \in K'} v_j))\}$;
  else if $\Delta < 0$ **then**
    $K_0^0 = \{k \in K' : r_k \leqslant \lambda + (1-\gamma)(C_1^0/V_1^0)\}$;
  $x_k^* = 0$ for $k \in K_0^0$;
  Continue with branching ;

Appendix B.2.

Algorithm 3: Preprocessing for node $t \neq 0$ when $\gamma \leqslant 1$

$\bar{K}^{t-1} \leftarrow$ set of undecided variables until node t ;
$K_1^{t-1} \leftarrow$ set of variables fixed to 1 until node t ;
if $\mathcal{H} > 0$ **then** // `Case 1`
  $r_{max} = \max_{j \in \bar{K}^{t-1}} \{r_j\}$;
  $K_1^t = \{k \in \bar{K}^{t-1} : r_k \geqslant \lambda + (1-\gamma)((C_1^{t-1} + (r_{max} - \lambda)\sum_{j \in \bar{K}^{t-1}} v_j)(V_1^{t-1} + \sum_{j \in \bar{K}^{t-1}} v_j))$;
  $x_k^* = 1, \forall k \in K_1^t$;
  Continue with upper bound $(V_1^t)^{\gamma-1} C_1^t e^{z_1^*(t)}$;
else if $\mathcal{H} < 0$ **then** // `Case 2`
  $C_1^{t-1} = \sum_{k \in K_1^{t-1}} v_k(r_k - \lambda) - \lambda v_0$;
  $V_1^{t-1} = v_0 + \sum_{k \in K_1^{t-1}} v_k$;
  $r_{min} = \min_{j \in \bar{K}^{t-1}} \{r_j\}$;
  $\Delta = C_1^{t-1} - V_1^{t-1}(r_{min} - \lambda)$;
  if $\Delta \geqslant 0$ **then**
    $K_0^t = \{k \in \bar{K}^{t-1} : r_k \leqslant \lambda + (1-\gamma)((C_1^{t-1} + (r_{min} - \lambda)\sum_{j \in \bar{K}^{t-1}} v_j)/(V_1^{t-1} + \sum_{j \in \bar{K}^{t-1}} v_j))\}$;
  else if $\Delta < 0$ **then**
    $K_0^t = \{k \in \bar{K}^{t-1} : r_k \leqslant \lambda + (1-\gamma)(C_1^{t-1}/V_1^{t-1})\}$;
  $x_k^* = 0$ for $k \in K_0^t$;
  Continue with branching ;

Appendix B.3.

Algorithm 4: Preprocessing for $t = 0$ when $\gamma > 1$

$N \leftarrow$ set of all products ;
$\mathcal{H} \leftarrow \sum_{k \in K : r_k \geqslant \lambda} v_k(r_k - \lambda) - \lambda v_0$;
if $\mathcal{H} = 0$ **then** // `Case 0`
  $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$;
  $x_k^* = 1, \forall k \in K_1^0$;
  $K_0^0 = \{k \in N \setminus K_1^0 : r_k < \lambda\}$;
  $x_k^* = 0, \forall k \in K_0^0$;
  Exit (subproblem is solved) ;
else if $\mathcal{H} > 0$ **then**// `Case 1`
  $K_1^0 = \{k \in N : r_k \geqslant \lambda\}$;
  $x_k^* = 1, \forall k \in K_1^0$;
  $C_1^0 = \sum_{k \in K_1^0} v_k(r_k - \lambda) - \lambda v_0$;
  $V_1^0 = v_0 + \sum_{k \in K_1^0} v_k$;
  $K_0^0 = \{k \in N \setminus K_1^0 : r_k < \lambda - (\gamma - 1)(C_1^0/V_1^0)\}$;
  $x_k^* = 0, \forall k \in K_0^0$;
  Continue with upper bound $(V_1^t)^{\gamma-1} C_1^t e^{z_2^*(t)}$;
else if $\mathcal{H} < 0$ **then** // `Case 2`
  $K_0^0 = \{k \in N : r_k < \lambda\}$;
  $x_k^* = 0, \forall k \in K_0^0$;
  Continue with branching ;

Appendix B.4.

Algorithm 5: Preprocessing for $t \neq 0$ when $\gamma > 1$

$\bar{K}^{t-1} \leftarrow$ set of undecided variables until node t ;
$K_1^t \leftarrow$ set of variables fixed to 1 until node t ;
$C_1^t \leftarrow \sum_{k \in K_1^t} v_k(r_k - \lambda) - \lambda v_0$;
$V_1^t \leftarrow v_0 + \sum_{k \in K_1^t} v_k$;
if $\mathcal{H} > 0$ **then**
    $K_0^t = \{k \in \bar{K}^{t-1} : r_k < \lambda - (\gamma - 1)(C_1^t / V_1^t)\}$;
    $x_k^* = 0, \forall k \in K_0^t$;
    Continue with upper bound $(V_1^t)^{\gamma-1} C_1^t e^{z_2^*(t)}$;

ESSEC
BUSINESS SCHOOL

ESSEC Business School
3 avenue Bernard-Hirsch
CS 50105 Cergy
95021 Cergy-Pontoise Cedex
France
Tel. +33 (0)1 34 43 30 00
www.essec.edu

ESSEC Executive Education
CNIT BP 230
92053 Paris-La Défense
France
Tel. +33 (0)1 46 92 49 00
www.executive-education.essec.edu

ESSEC Asia-Pacific
5 Nepal Park
Singapore 139408
Tel. +65 6884 9780
www.essec.edu/asia

ESSEC | CPE Registration number 200511927D
Period of registration: 30 June 2017 - 29 June 2023
Committee of Private Education (CPE) is part of SkillsFuture Singapore (SSG)

ESSEC Africa
Plage des Nations - Golf City
Route de Kénitra - Sidi Bouknadel (Rabat-Salé)
Morocco
Tel. +212 (0)5 37 82 40 00
www.essec.edu

CONTACT

RESEARCH CENTER

research@essec.edu