Models and Algorithms for the Product Pricing with Single-Minded Customers Requesting Bundles

Víctor Bucarey, Sourour Elloumi, Martine Labbé, Fränk Plein

Models and Algorithms for the Product Pricing with Single-Minded Customers Requesting Bundles

Víctor Bucarey[1,2], Sourour Elloumi[3,4], Martine Labbé[1,2], and Fränk Plein[1,2]

[1]Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium.
[2]Inria Lille-Nord Europe, Villeneuve d'Ascq, France.
{vbucarey, mlabbe, fplein}@ulb.ac.be
[3]UMA-Ensta ParisTech, Paris, France.
[4]CEDRIC-Cnam, Paris, France.
sourour.elloumi@ensta-paristech.fr

March 4, 2019

Abstract

We analyze a product pricing problem with single-minded customers, each interested in buying a bundle of products. The objective is to maximize the total revenue and we assume that supply is unlimited for all products. We contribute to a missing piece of literature by giving some mathematical formulations for this single-minded bundle pricing problem. We first present a mixed-integer nonlinear program with bilinear terms in the objective function and the constraints. By applying classical linearization techniques, we obtain two different mixed-integer linear programs. We then study the polyhedral structure of the linear formulations and obtain valid inequalities based on an RLT-like framework. We develop a Benders decomposition to project strong cuts from the tightest model onto the lighter models. We conclude this work with extensive numerical experiments to assess the quality of the mixed-integer linear formulations, as well as the performance of the cutting plane algorithms and the impact of the preprocessing on computation times.
Keywords: Pricing Problems, Integer Programming Formulations, Benders Decomposition

1 Introduction.

In this article we analyze a problem in product pricing under a model of single-minded customer behavior, determined by a bundle and a budget. A bundle is a subset of products, that a given customer wants to purchase, and he is willing to pay at most his budget. The single-mindedness of the client translates into him purchasing his bundle if and only if the corresponding total price does not exceeds his budget. The objective is to maximize the total revenue, assuming unlimited supply. This problem is known under Single-Minded Bundle Pricing Problem (SMBPP).

The SMBPP is a particular problem from the field of product pricing. We consider a *reservation price* customer behavior given by the budget, as opposed to a multinomial-logit approach that is often found in the revenue management literature [1, 9, 15]. However, there is a link between both models exhibited in [13]. Furthermore, [18] gives an overview of different client behavior models.

Under the model of reservation price, clients behavior is determined by the optimization of a certain utility function. In [17], maximum and minimum utility functions, as well as a maximum rank utility function, are introduced. In the SMBPP, clients can be seen to maximize their utility defined by the difference of their budget minus their bundle price.

When supplies are limited, the notion of *envy-free assignment* is discussed in [8]. Since we assume unlimited supply, every assignment is automatically envy-free. In the PhD thesis of [12] several other aspects of product pricing problems are discussed.

The SMBPP is the problem of pricing individual products statically when customers have a budget and request a single subset of products. A similar problem is considered in [6], where prices are fixed for

subsets of products, instead of individual products. They show that under the assumptions of unlimited supply, this problem can be solved in polynomial time. On the other hand, [7] show that the SMBPP is NP-hard, even when bundles have size 2. The authors devise a PTAS from their analysis for some special cases. It is also shown that the SMBPP is APX-hard in [8] and a 4-approximation algorithm is given in [2]. Further research was done on the hardness of approximation in [10] improving the best-known approximation bounds. To the best of our knowledge, most of the literature on the SMBPP is concerned with complexity and approximation results, see also for instance [4, 12]. This justifies our choice of contributing to a missing piece of literature by formulating the problem as a mixed-integer linear program (MILP).

Other problems that have attracted great interest in the literature and that share a similar combinatorial structure with the SMBPP are combinatorial auctions. Consider [11] and the references therein. Bidders request a bundle of products and their bids are the maximum price they are willing to pay. The goal of the auctioneer is however not to price his goods, but rather to determine an envy-free assignment of a limited supply to bidders in order to maximize his revenue.

The contributions of this work include the introduction of mathematical models for the SMBPP. In particular, we propose mixed-integer nonlinear and linear formulations. In order to get stronger formulations, valid inequalities and a polyhedral study are presented. Also, a Benders reformulation is presented in order to increase the scalability of our formulations.

The structure of the paper is as follows. The problem statement and notation is introduced in Section 2. In Section 3, we introduce basic notations and give new mixed-integer nonlinear programming (MINLP) and MILP formulations for the SMBPP. In Section 4, we discuss polyhedral properties of the MILP formulations. Furthermore, based on an RLT-like approach, we obtain valid inequalities that lead to a new tighter MILP formulation. Section 5 presents a Benders reformulation of the tightest MILP formulation. We then discuss stabilization methods in order to accelerate the cutting plane generation procedure. In Section 6, we present preprocessing methods to decrease the size of the problem. Section 7 summarizes the main computational results obtained. Finally, we present our conclusions and discuss future work in Section 8.

2 Problem statement.

Let $N = \{1, \dots, n\}$ be a set of products with unlimited supply and let $M = \{1, \dots, m\}$ be a set of clients.

We consider a reservation price model with single-minded customers, i.e., each client $j \in M$ is entirely represented by a bundle $S^j \subseteq N$ and a budget $b^j > 0$. The problem information can thus be encoded in a 0-1-matrix $S = [S_i^j]_{i \in N, j \in M}$ where

$$S_i^j := \begin{cases} 1, & \text{if product } i \in S^j, \\ 0, & \text{otherwise,} \end{cases}$$

and a vector of positive budgets b. A single-minded client purchases his bundle whenever its total price is inferior to his budget. The total price of a bundle is given by the sum of the prices for the products that constitute it.

The *single-minded bundle pricing problem (SMBPP)* consists of determining prices $p_i \geq 0$ for each product $i \in N$, in order to maximize the total revenue obtained by selling bundles to clients. The total bundle price is denoted by

$$p(S^j) := \sum_{i \in S^j} p_i.$$

A client purchases his bundle if and only if the total price for the bundle is less than his budget, so when $p(S^j) \leq b^j$. The SBMPP which maximizes the total revenue is

$$\max_{p \geq 0} \sum_{j \in M} \text{Revenue}_j(p)$$

where $\text{Revenue}_j(p)$, is the revenue obtained from client $j \in M$, which takes value $p(S^j)$ if $p(S^j) \leq b^j$ and 0 otherwise.

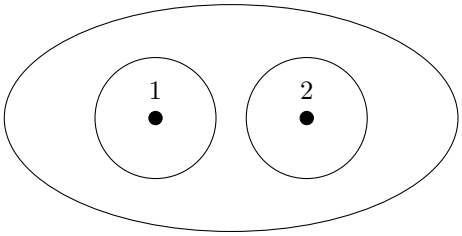

Figure 1: Illustration of bundles.

Example 1. Take for instance the following input data

$$S = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad\qquad b = \begin{pmatrix} 2 & 3 & 4 \end{pmatrix}$$

We then need to solve the SMBPP with two products $N = \{1, 2\}$ and three clients $M = \{1, 2, 3\}$. Furthermore, the bundles can be represented as in Figure 1. A first client desires to purchase both products and is willing to pay at most 2 price units in total. A second client seeks to purchase product 1 for at most 3 price units, whereas the third client requests product 2 at a budget of 4. For instance, the single-mindedness of the first client implies that he decides to purchase his bundle, when its total price $p(S^1) = p_1 + p_2$ is at most 2. If however, the price of his bundle exceeds his budget, he does not purchase it.

The optimal solution is obtained by setting $p_1 = 3$ and $p_2 = 4$. Then, client 1 does not purchase, and clients 2 and 3 purchase, resulting in a total profit of 7.

3 Formulations.

In this section, we first derive a mixed-integer nonlinear program (MINLP) formulation for the SMBPP involving products of price variables and buying decision variables. Next, we show how price variables can be bounded and how to linearize the nonlinear terms. We then give two differ- ent mixed-integer linear program (MILP) formulations obtained by applying linearization in two different ways.

3.1 MINLP formulation.

We start by giving a MINLP formulation that is directly derived from the problem description. We introduce binary variables $x_j \in \{0, 1\}$ for every client's purchase decision, i.e. $x_j = 1$ if and only if customer $j \in M$ purchases his bundle S^j.

$$\max_{p_i, x_j} \quad \sum_{j \in M} p(S^j) x_j \tag{1a}$$

$$\text{s.t.} \quad x_j(p(S^j) - b^j) \leq 0, \qquad j \in M, \tag{1b}$$

$$(1 - x_j)(p(S^j) - b^j) \geq 0, \qquad j \in M, \tag{1c}$$

$$p_i \geq 0, \qquad i \in N, \tag{1d}$$

$$x_j \in \{0, 1\}, \quad j \in M. \tag{1e}$$

Constraints (1b) ensure that if the total bundle price is greater than the budget, $p(S^j) > b^j$, then client $j \in M$ cannot purchase, $x_j = 0$. Conversely, Constraints (1c) ensure that $x_j = 1$ if the price is lower than the budget. Finally, we maximize the total profit given by $\sum_{j \in M} p(S^j) x_j$.

We would like to conclude this subsection by some remarks:

First, by optimality, Constraints (1c) are always satisfied, because as soon as $p(S^j) \leq b^j$, it is profitable to set $x_j = 1$. Our goal is to solve the SMBPP to optimality, we will thus often use the following formulation

$$\text{(NLM)} \quad \max_{p_i, x_j} \quad \sum_{j \in M} p(S^j) x_j \tag{2a}$$
$$\text{s.t.} \quad \text{(1b)}, \text{(1d)}, \text{(1e)}.$$

In a feasible solution of this formulation, a client is not forced to purchase his bundle when $p\left(S^j\right) \leq b^j$. Nonetheless, an optimal solution of the relaxed formulation has the same value as an optimal solution of the complete formulation.

Second, again by optimality, variables x_j always take a binary value even when (NLM) is solved over $[0, 1]$.

Third, the objective function and constraints of (NLM) are bilinear non-convex. Finally, the problem is always bounded since the maximum revenue cannot be larger than $\sum_{j \in M} b^j$. Furthermore, the revenue corresponding to an optimal solution is at least $\max_{j \in M} b^j$.

3.2 Implicit upper bound on prices.

To linearize terms that are product of continuous variables p_i and binary variables x_j, we first need an upper bound on the price variables.

Let us denote by $U(p_i)$ the upper bound on variable p_i for $i \in N$. Even though, prices p_i are not bounded a priori, it is not hard to observe that we can set

$$U(p_i) := \max_{j \in M} \{b^j : i \in S^j\}.$$

If we fix $p_i > U(p_i)$, any client $j \in M$ with bundle $S^j \ni i$ cannot purchase. Hence, product i is never bought. This implies that $p_i \leq U(p_i)$ in any optimal solution. We give an example to convince the reader that these bounds are tight in general.

Example 2. Consider the instance of 1 product and 2 clients given by the following input:

$$S = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad\qquad b = \begin{pmatrix} 1 & 10 \end{pmatrix}$$

Both clients want to purchase the same product but with different budgets. It is straightforward to see that an optimal solution yields a revenue of 10 and is obtained by setting $p = 10 = U(p)$, so that $x_1 = 0$ and $x_2 = 1$.

For a subset of products $S \subseteq N$, denote by $U(S)$ the upper bound on the total price $\text{p(S)}{:=}\sum_{i \in S} p_i$ of the subset S. As a natural extension, we define $U(S) := \sum_{i \in S} U(p_i)$.

3.3 Aggregated and disaggregated MILP formulations.

In a first attempt, we linearize the objective function and the constraints of (NLM) by replacing the products of total bundle price and buying decision using the classical techniques of [14]. To that end, we introduce new variables $r_j := p(S^j) x_j$ for all $j \in M$ and obtain a first MILP formulation direct from (NLM),

$$(\text{LM}_1) \quad \max_{p_i, x_j, r_j} \quad \sum_{j \in M} r_j \tag{3a}$$
$$\text{s.t.} \quad \text{(1d)}, \text{(1e)},$$
$$r_j \leq b^j x_j, \qquad j \in M, \tag{3b}$$
$$r_j \leq p(S^j), \qquad j \in M, \tag{3c}$$
$$r_j \geq p(S^j) - U(S^j)(1 - x_j), \quad j \in M, \tag{3d}$$
$$r_j \geq 0, \qquad j \in M. \tag{3e}$$

Constraints (3b) are the linear version of Constraints (1b). We can omit the additional McCormick inequalities $r_j \leq U(S^j) x_j$, which are dominated by (3b).

After this linearization step, the new variables $r_j := p\left(S^j\right) x_j$ can be interpreted as $\text{Revenue}_j(p)$, for client $j \in M$. Furthermore, r_j represents an aggregate of products $p_i x_j$ for all $i \in S^j$, hence the name of aggregated model.

In a second attempt, we can be more precise in our linearization of (NLM) and replace the products $p_i x_j$ for any $i \in S^j$ and $j \in M$. Therefore, we introduce variables

$$s_{ij} = p_i x_j \quad j \in M,\, i \in S^j. \tag{4}$$

We obtain a disaggregated formulation as opposed to aggregated formulation (LM_1).

$$\begin{aligned}
(\text{LM}_2) \quad & \max_{p_i, x_j, s_{ij}} \quad \sum_{j \in M} \sum_{i \in S^j} s_{ij} && && \text{(5a)}\\
& \text{s.t.} \quad && (1d), (1e), && \\
& && \sum_{i \in S^j} s_{ij} \leq b^j x_j, && j \in M, && \text{(5b)}\\
& && s_{ij} \leq p_i, && i \in S^j, j \in M, && \text{(5c)}\\
& && s_{ij} \geq p_i - U(p_i)(1 - x_j), && i \in S^j, j \in M, && \text{(5d)}\\
& && s_{ij} \geq 0, && i \in S^j, j \in M. && \text{(5e)}
\end{aligned}$$

We only point out that for $j \in M$ and $i \in S^j$, Constraints (5b) imply that $s_{ij} \leq b^j x_j$ and hence, $s_{ij} \leq U(p_i) x_j$ are redundant by definition of $U(p_i)$.

4 Polyhedral properties and a new formulation.

In this section, we study the polyhedral properties of (LM_1) and (LM_2). We then discuss families of valid inequalities and finally present a stronger formulation for the SBMPP.

4.1 Polyhedral study.

Let us first introduce some notation. Denote by $v(\cdot)$ the optimal value of a given problem. We also denote ($\overline{\text{LM}_i}$) the linear relaxation of model (LM_i), $i = 1, 2$. We first establish how formulations (LM_1) and (LM_2) are related through the following proposition:

Proposition 3. The following inequality holds:

$$v(\overline{\text{LM}_1}) \leq v(\overline{\text{LM}_1}).$$

Proof. Proof:

Let (p, x, s) be a feasible solution to $\overline{\text{LM}_2}$. Then define r_j satisfying

$$r_j = \sum_{i \in S^j} s_{ij} \quad j \in M. \tag{6}$$

It is easy to verify that (p, x, r) is feasible for $\overline{\text{LM}_1}$ with the same objective value. □

□

We give an example to show that we can find $v(\overline{\text{LM}_2}) < v(\overline{\text{LM}_1})$:

Example 4. Consider an instance of 2 products and 2 clients given by

$$S = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad b = \begin{pmatrix} 10 & 20 \end{pmatrix}.$$

The optimal solution of ($\overline{\text{LM}_1}$) is obtained by setting $x = (\frac{1}{2}, 1)$ and $p = (20, 0)$ yielding a revenue of $v(\overline{\text{LM}_1}) = 25$. The optimal value of ($\overline{\text{LM}_2}$) is $v(\overline{\text{LM}_1}) = 20$. The optimal prices are the same, but the fractional optimal solution of ($\overline{\text{LM}_1}$) is cut off by constraints $s_{1,1} \geq p_1 - 20(1 - x_1)$, $s_{1,1} + s_{1,2} \leq 10 x_1$ and the fact that $s_{1,2} \geq 0$.

We are also interested in the structure of the convex hull of mixed-integer feasible solutions. We denote it by $P^I(\text{LM}_1)$ (respectively $P^I(\text{LM}_2)$) the convex hull of mixed integer solutions to LM_1 (respectively LM_2). A detailed polyhedral study is given in [16]. The author has shown the following proposition:

Proposition 5. $P^I(\text{LM}_1)$ and $P^I(\text{LM}_2)$ are full dimension in their relative space. Further, Constraints (5b), (5c), (5d), and (5e) induce facets of $P^I(\text{LM}_2)$.

4.2 Valid inequalities and a new formulation.

The different formulations we have obtained so far are all based on linearizations of our first MINLP model (NLM). The definition of $U(p_i)$ implies that the relaxations become weaker when products appear in more bundles. Furthermore, the structure of (NLM) does not link the binary buying decisions of clients $j, k \in M$ explicitly, but there is an implicit link through the prices p_i for $i \in S^j \cap S^k$. We intend to create an explicit link using RLT-like valid inequalities:

$$(p(S^k) - b^k)(x_k + x_j - 1) \leq 0 \qquad j, k \in M,\ j \neq k \tag{7}$$

$$(p(S^k) - b^k)(x_k - x_j) \leq 0 \qquad j, k \in M,\ j \neq k \tag{8}$$

Proposition 6. Expressions (7) and (8) are valid inequalities.

Proof. Proof: Consider two clients $j, k \in M$, with $j \neq k$. For client k, we multiply the corresponding Constraint (1b) by $-x_j \leq 0$ and Constraint (1c) by $1 - x_j \geq 0$.

$$(p(S^k) - b^k)(x_k - x_k x_j) \leq 0$$
$$-(p(S^k) - b^k)(x_j - x_k x_j) \leq 0$$

Summing up the resulting valid inequalities, we obtain Constraint (8) corresponding to clients j and k. Constraint (8) is obtained similarly, by multiplying Constraint (1b) by $-(1 - x_j) \leq 0$ and Constraint (1c) by $x_j \geq 0$. □ □

To take advantage of this explicit link, let us extend the definition (4) of variables s_{ij} to all $i \in N$, $j \in M$. With these additional variables s_{ik}, $i \notin S^k$, we can then linearize Constraints (7) and (8). This leads to the following formulation:

$$(\text{LM}_3) \quad \max_{p_i, x_j, s_{ij}} \quad \sum_{j \in M} \sum_{i \in S^j} s_{ij} \tag{9a}$$

$$\text{s.t.} \quad (1d), (1e), (5b),$$

$$s_{ij} \leq p_i, \qquad i \in N, j \in M, \tag{9b}$$

$$s_{ij} \leq U(p_i) x_j, \qquad i \in N, j \in M, \tag{9c}$$

$$s_{ij} \geq p_i - U(p_i)(1 - x_j), \qquad i \in N, j \in M, \tag{9d}$$

$$\sum_{i \in S^k} (s_{ik} + s_{ij} - p_i) \leq b^k (x_k + x_j - 1), \qquad j, k \in M, j \neq k, \tag{9e}$$

$$\sum_{i \in S^k} (s_{ik} - s_{ij}) \leq b^k (x_k - x_j), \qquad j, k \in M, j \neq k, \tag{9f}$$

$$s_{ij} \geq 0, \qquad i \in N, j \in M. \tag{9g}$$

This model is clearly an extension of (LM_2). The following proposition states that Constraints (9c) can be omitted, even for $i \notin S^j$.

Proposition 7. Constraints (9c) are redundant in (LM_3).

Proof. Proof: In Section 3, we already observed that $s_{ij} \leq U(p_i) x_j$ is redundant for $i \in S^j$. Further, for every optimal solution of (LM_3) with $s_{ij} > U(p_i) x_j$, for $i \notin S^j$, we can build a new feasible solution with the same objective function value by setting $s_{ij} = U(p_i) x_j$ for those variables.

For a client j_0 and a product i_0 such that $i_0 \notin S^{j_0}$, if $s_{i_0 j_0} > U(p_{i_0})x_{j_0}$, we build a new solution with $\tilde{s}_{i_0 j_0} = U(p_{i_0})x_{j_0}$. We show that this new solution is feasible. First, given that this new solution decreases the LHS of (9b) and (9e), they are satisfied. Given that $\tilde{s}_{i_0 j_0} = U(p_{i_0}) \geq p_{i_0}$ when $x_{j_0} = 1$ and $\tilde{s}_{i_0 j_0} = 0 \geq p_{i_0} - U(p_{i_0})$ when $x_{j_0} = 0$, Constraints (9d) are also satisfied. For a fixed client $k \in M$, if $i_0 \notin S^k$, Constraint (9f) is trivially satisfied. Otherwise, $\tilde{s}_{i_0 j_0}$ satisfies

$$\tilde{s}_{i_0 j_0} \geq b^k x_{j_0} \geq b^k x_{j_0} - \left(\sum_{i \in S^k} s_{ik} - b^k x_k \right) - \sum_{i \in S^k \setminus \{i_0\}} s_{i j_0} \,,$$

where the first inequality comes from the fact that $b^k \leq U(p_{i_0})$ when $i_0 \in S^k$ and the second inequality is derived by (5b) and $s_{ik} \geq 0$. Then, Constraint (9f) is also satisfied. Finally, since $i_0 \notin S^{j_0}$, the objective function does not change and the result follows. □ □

The new variables s_{ij} for $i \notin S^j$ allow to create a relation to a client k for whom $i \in S^k$ by means of Constraints (9e) and (9f). We have therefore managed to model an explicit link between clients and the intersection of their bundle. However, (LM_3) contains $\mathcal{O}(NM)$ variables and $\mathcal{O}(M^2)$ constraints, as opposed to the $\mathcal{O}(N+M)$ variables and $\mathcal{O}(M)$ constraints of (LM_1).

5 Benders decomposition.

In this section, we first present a Benders reformulation for (LM_3). We then give a cutting plane algorithm and discuss stabilization methods to accelerate its performance.

5.1 Reformulating (LM_3).

As a first step, we rewrite (LM_3) by introducing variables $r_j := \sum_{i \in S^j} s_{ij}$ for all $j \in M$. The objective function of (LM_3) can then be rewritten as $\max \sum_{j \in M} r_j$. For fixed x, p and r, it remains to solve a feasibility problem given by Constraints (6), (9b), (9d), (9e) and (9f) in the variables s_{ij}, which can be decomposed by clients $j \in M$. We name this problem (SP_j).

The dual of this sub-problem for client j, which determines the most violated cut, is given by

$$\begin{aligned}
(\text{DSP}_j) \quad & \min_{\alpha} \alpha_j^0 r_j + \sum_{i \in N} \alpha_{ij}^1 p_i - \sum_{i \in N} \alpha_{ij}^2 [p_i - U(p_i)(1 - x_j)] \\
& - \sum_{k \in M:\, k \neq j} \alpha_{jk}^3 \left[r_k - b^k (x_k - x_j) \right] \\
& + \sum_{k \in M:\, k \neq j} \alpha_{jk}^4 \left[b^k (x_k + x_j - 1) + \sum_{i \in S^k} p_i - r_k \right] && (10) \\
\text{s.t.} \quad & \alpha_j^0 + \alpha_{ij}^1 - \alpha_{ij}^2 - \sum_{\substack{k \in M:\, k \neq j \\ i \in S^k}} \alpha_{kj}^3 + \sum_{\substack{k \in M:\, k \neq j \\ i \in S^k}} \alpha_{kj}^4 \geq 0 \qquad i \in S^j && (11) \\
& \alpha_{ij}^1 - \alpha_{ij}^2 - \sum_{\substack{k \in M:\, k \neq j \\ i \in S^k}} \alpha_{kj}^3 + \sum_{\substack{k \in M:\, k \neq j \\ i \in S^k}} \alpha_{kj}^4 \geq 0 \qquad i \notin S^j && (12) \\
& \alpha_j^0 \in \mathbb{R},\ \alpha_j^1, \alpha_j^2, \alpha_j^3, \alpha^4 \geq 0, && (13)
\end{aligned}$$

where $\alpha_j^0, \alpha_j^1, \alpha_j^2, \alpha_j^3$ and α_j^4 are the dual variables associated with Constraints (6), (9b), (9d), (9e) and (9f), respectively.

The master problem of the Benders reformulation is

$$
\begin{aligned}
\text{(MP)} \quad & \max \sum_{j \in M} r_j \\
& r_j \le b^j x_j && j \in M \\
& \alpha_j^0 r_j + \sum_{i \in N} \alpha_{ij}^1 p_i - \sum_{i \in N} \alpha_{ij}^2 [p_i - U(p_i)(1 - x_j)] && (14) \\
& - \sum_{k \in M:\, k \neq j} \alpha_{kj}^3 [r_k - b^k (x_k - x_j)] \\
& + \sum_{k \in M:\, k \neq j} \alpha_{kj}^4 \left[b^k (x_k + x_j - 1) + \sum_{i \in S^k} p_i - r_k \right] \ge 0 \qquad \alpha_j \text{ extreme ray of } Q^D && (15) \\
& p_i \ge 0,\ r_j \ge 0, x_j \in \{0,1\} && i \in N,\ j \in M
\end{aligned}
$$

Constraints (15) can be generated using the classical Benders approach. We denote (RMP) to the relaxed master problem which includes only some of Constraints (15). In a regular iteration, the algorithm solves the master problem with some feasibility cuts retrieving optimal values x, p, and r. The optimal value v(RMP) of the problem at every iteration is an upper bound for the original problem. The optimal solution is introduced in each (DSP$_j$). Each of these problems are upper bounded by 0 ($\alpha = 0$ is always a feasible solution). In consequence, for each problem (DSP$_j$) there is either an optimal solution with value 0 or it admits an extreme ray which makes the problem unbounded.

As a consequence, there are no optimality cuts. The procedure will stop if at any iteration, all the problems (DSP$_j$) have optimal value 0. In that case, the last values of x, p and r are optimal. This procedure is summarized in Algorithm 1.

Algorithm 1 Benders decomposition algorithm for (LM$_3$)

Require: N, M, S, b
  $UB := +\infty$
  Set convergence = False
  while not convergence **do**
    Solve (RMP) retrieving x, p and r.
    Update upper bound with v(RMP).
    Set convergence = True
    for $j \in M$ **do**
      Solve (DSP$_j$)
      if (DSP$_j$) is unbounded **then**
        Add feasibility cut to (RMP)
        Set convergence = False
      end if
    end for
  end while
  return x, p, v(RMP)

We use this general scheme to solve the (LM$_3$) to optimality or solving just its LP relaxation, at the root node of the Branch-and-Bound algorithm. Furthermore, we use this algorithm to improve weaker formulations. In particular, given that (LM$_1$) is the fastest formulation, with weak LP bound—see computational results in Section 7.2—we can use the cuts generated by applying the Benders decomposition to the strongest formulation (LM$_3$) and strengthen the LP bound of (LM$_1$).

5.2 Stabilization methods.

In general, cutting plane algorithms can perform very poorly, mainly, due to two reasons. First, the upper bound, i.e., the optimal value of the current relaxation, can decrease very slowly. Second, even when the optimal value is reached the algorithm may continue generating cuts to obtain an optimality certificate.

Benders decomposition is not an exception to these problems. In this section, we compare and analyze two methods: the first one, In-Out stabilization, was proposed by [3]; the second one, equivalent to a facet separation, was proposed by [5], we name it (CW). In both methods, the separation point used to generate a new cut lies in the segment linking a point in the relative interior of the feasible domain of (MP) and the optimal point of the current relaxation. The main goal is to decrease the number of cuts generated, by obtaining fewer but tighter cuts.

The interior point, denoted by (x^{in}, p^{in}, r^{in}), can be computed as an interior point of the feasible region of (LM_3) and projected into (MP). In algorithm In-Out, the point used in each cut generation LP is a convex combination of the interior point (x^{in}, p^{in}, r^{in}) and the optimal solution of the relaxed master problem, denoted by $(x^{out}, p^{out}, r^{out})$. Let $\Delta x = x^{out} - x^{in}$ and define Δp and Δr, analogously. For fixed $\bar{\lambda} \in (0, 1]$, the separation point used in each sub-problem is then given by $(x^{sep}, p^{sep}, r^{sep}) = (x^{in}, p^{in}, r^{in}) + \bar{\lambda}(\Delta x, \Delta p, \Delta r)$.

There are two possible outcomes. Either the separation point is feasible, in that case the interior point is updated, or it is infeasible and a new cut is generated. The method is outlined in Algorithm 2.

Algorithm 2 In-Out stabilization for Benders decomposition

Require: N, M, S, b and $\bar{\lambda} \in (0, 1]$.
  $UB := +\infty$
  Compute an interior point (x^{in}, p^{in}, r^{in}).
  Set convergence = False
  while not convergence **do**
    Solve (RMP) retrieving $x^{out}, p^{out}, r^{out}$.
    Update upper bound with $v(RMP)$.
    Compute $(x^{sep}, p^{sep}, r^{sep}) = (x^{in}, p^{in}, r^{in}) + \bar{\lambda}[(x^{out}, p^{out}, r^{out}) - (x^{in}, p^{in}, r^{in})]$
    for $j \in M$ **do**
      Solve (DSP_j) using $(x^{sep}, p^{sep}, r^{sep})$ as a separation point.
      if (DSP_j) is unbounded. **then**
        Add feasibility cut to (RMP)
      end if
    end for
    if There is no cuts to add **then**
      Update the interior point $(x^{in}, p^{in}, r^{in}) = (x^{sep}, p^{sep}, r^{sep})$
      Optional: Check feasibility of $(x^{out}, p^{out}, r^{out})$
    end if
    if $\sum_{j \in M} r_j^{out} - \sum_{j \in M} r_j^{in} < \epsilon \sum_{j \in M} r_j^{in}$ **then**
      convergence = True
    end if
  end while
  return (x^{in}, p^{in}, r^{in}) and v(RMP)

In the methodology proposed by [5], the algorithm determines the largest value of λ such that $(x^{sep}, p^{sep}, r^{sep}) = (x^{in}, p^{in}, r^{in}) + \lambda(\Delta x, \Delta p, \Delta r)$ is feasible for (SP_j). With probability 1, this cut defines a facet of the convex hull of feasible solutions for LM_3. The sub-problem for a client $j \in M$ is stated as follows:

$$
\begin{aligned}
(\text{SP-CW}_j)\ \max_{\lambda,\tilde{r}}\ & \lambda \\
& \sum_{i\in S^j} s_{ij} = r_j^{in} + \lambda \Delta r_j \\
& s_{ij} \le p_i^{in} + \lambda \Delta p_i && i \in N \\
& s_{ij} \ge p_i^{in} - U(p_i)(1 - x_j^{in}) + \lambda(\Delta p_i - U(p_i)\Delta x_j) && i \in N \\
& \sum_{i\in S^k} s_{ij} \ge r_k^{in} - b^k(x_k^{in} - x_j^{in}) + \lambda[\Delta r_k - b^k(\Delta x_k - \Delta x_j)] && k \in M,\ k \ne j \\
& \sum_{i\in S^k} s_{ij} \le b^k(x_j^{in} + x_k^{in} - 1) + \sum_{i\in S^k} p_i^{in} - r_k^{in} + \lambda[b^k(\Delta x_k + \Delta x_j) \\
& \qquad + \sum_{i\in S^k} \Delta p_i - \Delta r_k] && k \in M,\ k \ne j \\
& 0 \le \lambda \le 1 \\
& s_{ij} \ge 0 && i \in N
\end{aligned}
$$

Its dual is:

$$
\begin{aligned}
(\text{DSP-CW}_j)\ \min_{\alpha,\beta}\ & \alpha^0 r_j^{in} + \sum_{i\in N} \alpha_i^1 p_i^{in} - \sum_{i\in N} \alpha_i^2 [p_i^{in} - U(p_i)(1 - x_j^{in})] \\
& - \sum_{k\in M:\, k\ne j} a_k^3 \left[r_k^{in} - b^k(x_k^{in} - x_j^{in})\right] \\
& + \sum_{k\in M:\, k\ne j} a_k^4 \left[b^k(x_k^{in} + x_j^{in} - 1) + \sum_{i\in S^k} p_i^{in} - r_k^{in}\right] + \beta \\
\text{s.t. } & \alpha^0 + \alpha_i^1 - \alpha_i^2 - \sum_{\substack{k\in M:\, k\ne j \\ i\in S^k}} \alpha_k^3 + \sum_{\substack{k\in M:\, k\ne j \\ i\in S^k}} \alpha_k^4 \ge 0 && i \in S^j \\
& \alpha_i^1 - \alpha_i^2 - \sum_{\substack{k\in M:\, k\ne j \\ i\in S^k}} \alpha_k^3 + \sum_{\substack{k\in M:\, k\ne j \\ i\in S^k}} \alpha_k^4 \ge 0 && i \notin S^j \\
& - \alpha^0 \Delta r_j - \sum_{i\in N} \alpha_i^1 \Delta p_i + \sum_{i\in N} \alpha_i^2 [\Delta p_i + U(p_i)\Delta x_j] + \\
& \sum_{k\ne j} \alpha_k^3 [\Delta r_k - b^k(\Delta x_k - \Delta x_j)] \\
& - \sum_{k\ne j} \alpha_k^4 [b^k(\Delta x_k + b^k \Delta x_j + \sum_{i\in S^k} \Delta p_i - \Delta r_k] + \beta \ge 1 \\
& \alpha^0 \in \mathbb{R}, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \beta \ge 0,
\end{aligned}
$$

Given that (SP-CW$_j$) is always feasible ($\lambda = 0$ is feasible) and its optimal value bounded by 1, both Q_j and (DSP-CW$_j$) has always finite optimal solutions. If $\lambda = 1$, then $(x^{out}, p^{out}, r^{out})$ is feasible. Cuts are added whenever the optimal value of (DSP-CW$_j$) is strictly smaller than 1. The new cut has the same form as in (15).

5.3 Preliminary comparison of In-Out and CW.

We tested the behavior in solving the LP relaxation of (LM$_3$) on a small instance with $n = 20$ and $m = 40$. We compare the traditional Benders decomposition, the In-Out stabilization with parameter $\bar{\lambda} \in \{0.25, 0.5, 0.75\}$ and the CW procedure. We fix a limit of 100 iterations, that is, the relaxed master problem is solved at most 100 times. To obtain a point in the relative interior, we use the barrier algorithm

implemented in CPLEX 12.8 without crossover applied to LM_3 and project the point into (MP) by setting $r_j = \sum_{i \in S^j} s_{ij}$ for every client $j \in M$.

Given that this problem only has feasibility cuts, only the upper bound is updated. Figure 2 shows the behavior of the upper bound through the iterations. CW is the procedure that takes fewer iterations to obtain the optimal solution. The main drawback of the In-Out stabilization is that it takes a lot of time to certify the optimality. To improve on this aspect, we add a step to check if the outer point is feasible (step 3 of Algorithm 2).

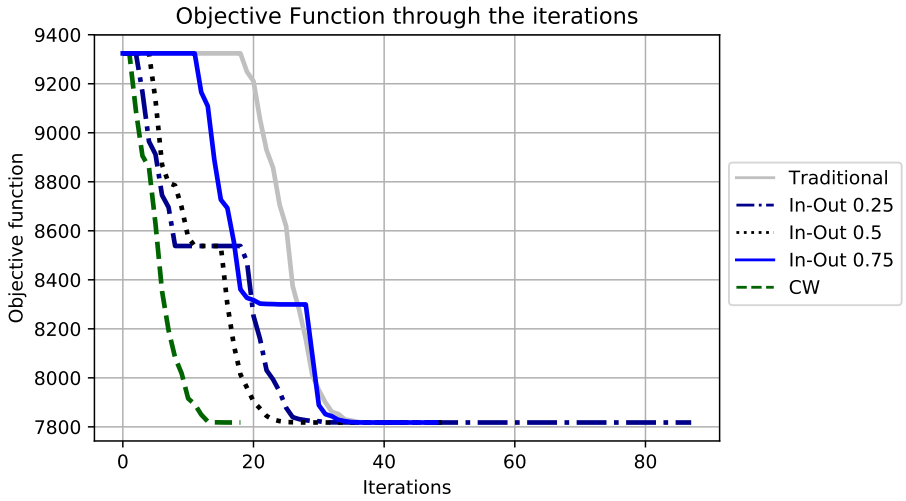

Figure 2: Upper bound through the iterations.

Table 5.3 shows the number of cuts added, the iteration number and the solution time. Clearly, the CW procedure outperforms all other methods.

	Traditional	In Out 0.25	In Out 0.5	In Out 0.75	CW
Cuts added	1691	2049	1176	969	526
Iterations	53	90	54	51	30
Solution Time [s]	136.00	225.20	109.83	103.58	54.27

Table 1: Some statistics of the example.

A more complete comparison of In-Out and CW is presented in Section 7.3.

6 Preprocessing.

In this section, we describe some methods that exploit the combinatorial structure of the SMBPP in order to decrease the problem size.

6.1 Connected components and merging.

We build an undirected graph G where the set of nodes are the clients and there exists an edge between clients j and k if their bundles have products in common, i.e., $S^j \cap S^k \neq \emptyset$. This graph can be built in $\mathcal{O}(NM^2)$ operations. If the graph has more than one connected component, we can decompose the problem. It is not hard to observe that prices and buying decisions corresponding to products and clients in different connected components are independent. Connected components can be computed in linear time in a straightforward way using either breadth-first search or depth-first search.

This decomposition also induces a partition in the products as is shown in the following example:

$$S = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad b = \begin{pmatrix} 10 & 50 & 20 & 15 \end{pmatrix}.$$

The induced graph is shown in Figure 3. The SMBPP for this instance can be solved by splitting the problem into two sub-problems: The first considering clients 1 and 4 and product 1, and the second considering clients 2 and 3, and products 2 and 3.

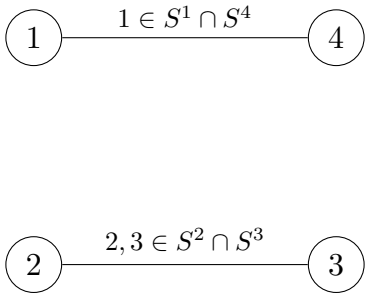

Figure 3: Induced graph in the example.

Also, the set of products can be reduced in the following way. Suppose that a subset of products $\tilde{N} \subseteq N$ appear only together in client bundles. Then those products can be considered as a single product, with $p_{\tilde{N}} := \sum_{i \in \tilde{N}} p_i$. For example, in the following instance, matrix S indicates that products 2 and 3 appear together in bundles of clients 1 and 2, then they can be treated as one single product. The resulting matrix S' has one row less.

$$S = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad S' = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

6.2 Fixing rich clients buying decisions.

Suppose there is one client with a large budget. Then it is reasonable to think that he will be able to buy his bundle in any optimal solution. The following proposition states one condition on the budgets to know a priori if a client with high budget is going to buy his bundle.

Proposition 8. If client $j \in M$ satisfies $\sum_{k \in M \setminus \{j\}: S^k \cap S^j \neq \emptyset} b^k \leq b^j$, then $x_j = 1$ in any optimal solution.

Proof. Proof: Suppose there exists an optimal solution x with $x_j = 0$. If for all k such that $S^k \cap S^j \neq \emptyset$ $x_k = 0$ then by setting $x_j = 1$ and $p_i = b^j$ a higher value solution is reached and x is not optimal. Let K_1 be the set of clients k such that $S^k \cap S^j \neq \emptyset$ and $x_k = 1$. That means that $b^k \geq \sum_{i \in S^k} p_i$ for all $k \in K_1$. Analogously, define K_0 as the set of clients k such that $S^k \cap S^j \neq \emptyset$ and $x_k = 0$. Note that $K_0 \neq \emptyset$ under the assumption $x_j = 0$. Then,

$$b^j > \sum_{k \in K_1} b^k \geq \sum_{k \in K_1} \sum_{i \in S^k} p_i \geq \sum_{i \in \cup_{k \in K_1} S^k} p_i \geq \sum_{i \in \cup_{k \in K_1} S^k \cap S^j} p_i$$

Then by setting $x_j = 1$, $p_i = 0$ for products $i \notin \cup_{k \in K_1} S^k \cap S^j$ and keeping the other prices the solution obtained is feasible and has a higher revenue, the result follows. □ □

We call clients satisfying this condition relatively rich clients. As a consequence, when considering Constraints (1c) for relatively rich clients, we observe that $x_j = 1$ is an implicit equality.

Now for each relatively rich client, we can replace the binary variables representing its purchase decision by constraints that only enforce prices to satisfy their budgets. Let $\tilde{M}$ be the set of relatively rich clients. For those clients, $r_j = \sum_{i \in S^j} p_i$ and $s_{ij} = p_i$, then we can rewrite (LM_1):

$$
\begin{aligned}
(\text{LM}_1 - \tilde{M}) \quad & \max \sum_{j \notin \tilde{M}} r_j + \sum_{j \in \tilde{M}} \sum_{i \in S^j} p_i & \\
s.t. \quad & r_j \leq b^j x_j & j \notin \tilde{M} \\
& \sum_{i \in S^j} p_i \leq b^j & j \in \tilde{M} \\
& r_j \leq \sum_{i \in S^j} p_i & j \notin \tilde{M} \\
& r_j \geq \sum_{i \in S^j} p_i - U(S^j)(1 - x_j) & j \notin \tilde{M} \\
& r_j \geq 0 & \forall j \notin \tilde{M} \\
& p_i \geq 0 & \forall i \in N \\
& x_j \in \{0, 1\} & j \notin \tilde{M}
\end{aligned}
$$

We can also reformulate (LM_3) but for the sake of shortness we describe this reformulation in Appendix A.

7 Computational Results.

In this section, we report the computational results of the models and algorithms proposed in this paper. All experiments have been carried out using CPLEX 12.8 and Python 3.5, in a single thread on a server with a 3.40Ghz Intel i7 processor and 64 GB of memory.

7.1 Instances.

A SMBPP instance consists of the number n of products and m clients. Each client is encoded by a positive real representing his budget and a subset of products representing his bundle. One aspect that seems to play an important role in solution times is the density of the matrix S. This density is defined by

$$d = \frac{\sum_{i \in N} \sum_{j \in M} S_i^j}{nm} \tag{16}$$

.

The size of LM_2, in terms of the number of constraints and variables, depends on the size of this parameter. On the other hand, the size of LM_1 and LM_3 do not depend on d. The sizes of the different formulations are summarized in Table 2.

Formulation	Variables	Constraints
LM_1	$n + 2m$	$3m$
LM_2	$n + m + dnm$	$m + 2dnm$
LM_3	$n + m + nm$	$m + 3nm + 2m(m-1)$

Table 2: Number of variables and constraints for each formulation.

In order to test the methods described in this paper, we generate three types of instances.

- Small instances: We generate instances for $n \in \{10, 25, 50\}$ number of products, $m \in \{10, 25, 50, 100, 150\}$ number of clients, and $d \in \{0.2, 0.5, 0.8\}$ density of matrix S. For every size and density, we generate 10 instances. Each instance was generated using Algorithm 3. We denote this family by Small(n, m, d). These instances are used to test the performance of (LM_1), (LM_2) and (LM_3).

- Big instances: Instances generated for $n \in \{100, 500\}$ products, and $m \in \{150, 200, 300\}$ clients. For every size, we generate 10 instances. We denote this family by Big(n, m). These instances are used to test the performance of the Branch-and-Cut and Cut-and-Branch algorithms. Instances were generated again by Algorithm 3 using $d = 0.5$.

- Rich-poor instances: Instances generated with a fixed number $m_1 \in \{25, 50\}$ of relatively rich clients and $m_2 \in \{25, 50\}$ other clients. The set of products was fixed to be of size $n = 200$. We denote these instances by RP(n, m_1, m_2). For every size, we generate 10 instances. The procedure to generate these instances is stated in Algorithm 4 in Appendix B.

Algorithm 3 Generate instances by density

Require: $m, n \in \mathbb{Z}_+, d \in (0, 1)$

Step 1: Create set of clients $M = \{1, \ldots, m\}$ and products $N = \{1, \ldots, n\}$.

Step 2: Create Budgets. For $j \in \{1, \ldots, m\}$ $b_j = U(1, 1000)$.

Step 3: For each $i \in N$ and $j \in M$ set $S_i^j = 1$ with probability d.

Step 4: For each client with empty bundle, or each product not appearing in any bundle, sample 1 product and client respectively and set $S_i^j = 1$.

return N, M, S, b with S with density approximately d.

7.2 Performance of (LM$_1$), (LM$_2$) and (LM$_3$).

For instances in Small(n, m, d), we compute the optimal solutions, the solution time, the LP gap if optimality is reached, and the number of nodes explored in the Branch-and-Bound algorithm. We set the time limit to 1 hour.

Amongst the 450 instances in Small(n, m, d), (LM$_1$) can solve 85.33% of the instances, while (LM$_2$) and (LM$_3$) solve only 74.67% and 74.44% respectively, within the time limit. In Figure 4, we present the performance profiles for these instances.

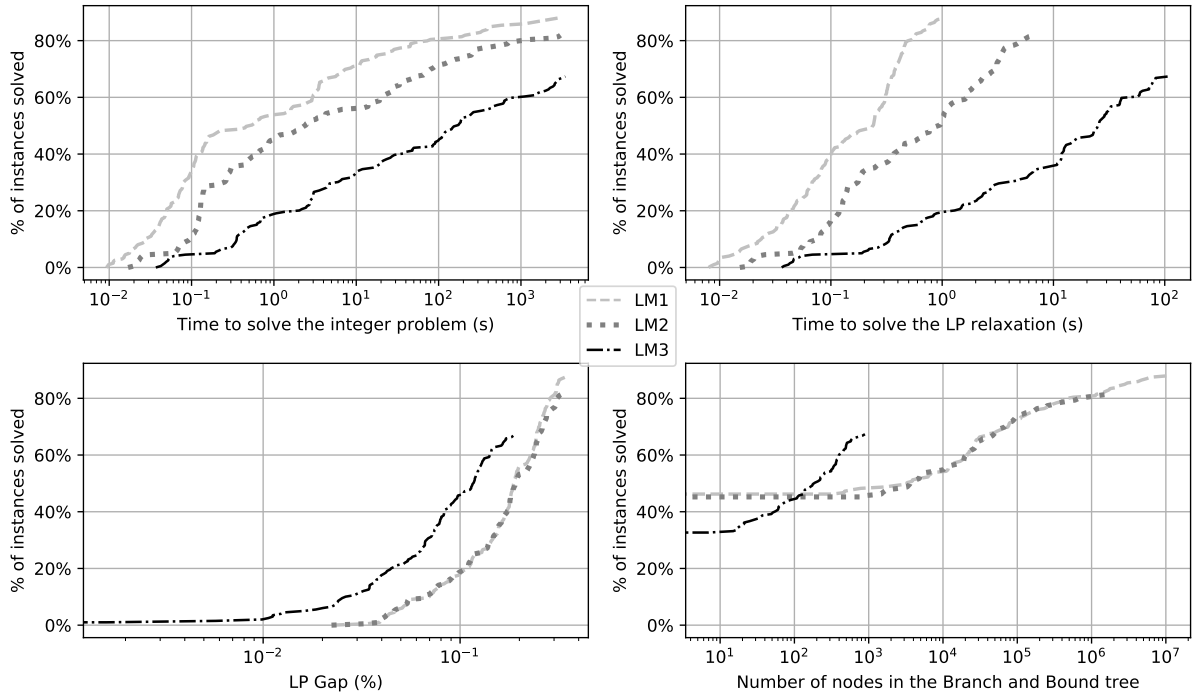

Figure 4: Performance profile for (LM$_1$), (LM$_2$) and (LM$_3$) for instances Small(n, m) within a time limit of one hour.

Model (LM_1) is the most efficient formulation in terms of solution time despite the fact that (LM_1) has the weakest linear relaxation. One interesting result is despite the theoretical result in Proposition 3, the improvement in terms of LP gap of (LM_2) is insignificant. However, the bound of (LM_3) is significantly tighter. The efficiency of (LM_1) came from the small amount of time required to solve its linear relaxation, which counteracts its weak linear relaxation and the large amount of nodes to explore to get an optimal solution.

Table 3 shows the percentage of instances solved by each model. Our results show that the most difficult instances are the ones with medium density $d = 0.5$. They also show that LM_1 and LM_3 perform very well for instances with high density, and LM_2 performs better for instances with the matrix S with low density.

	% solved			
Model / d	0.2	0.5	0.8	Aggregated
LM_1	82.00	80.67	93.33	85.33
LM_2	78.00	70.67	75.33	74.67
LM_3	74.00	70.00	79.33	74.44

Table 3: Instances solved for LM_1, LM_2 and LM_3 within a time limit of 1 hour.

Table 4 shows the same information disaggregated by instance size. LM_1 is the only formulation that can solve instances with 150 clients and 10 products to optimality within the time limit. LM_2 and LM_3 are not able to solve any instance with 150 clients. All instances with $n \leq 50$ and $m \leq 50$ can be solved by the three formulations. We show the LP gap and solution time for those instances in Tables 5 and 6. The three formulations have a bigger LP gap as d increases. However, it does not have a proportional impact on solution times. It seems that the hardest instances are the ones that have medium density $d = 0.5$,

		$d = 0.2$			$d = 0.5$			$d = 0.8$		
n	m	LM_1	LM_2	LM_3	LM_1	LM_2	LM_3	LM_1	LM_2	LM_3
10	10	10	10	10	10	10	10	10	10	10
	25	10	10	10	10	10	10	10	10	10
	50	10	10	10	10	10	10	10	10	10
	100	10	10	10	10	10	10	10	10	10
	150	10	10	10	10	6	5	10	8	10
25	10	10	10	10	10	10	10	10	10	10
	25	10	10	10	10	10	10	10	10	10
	50	10	10	10	10	10	10	10	10	10
	100	10	7	1	10	0	0	10	5	9
	150	0	0	0	0	0	0	10	0	0
50	10	10	10	10	10	10	10	10	10	10
	25	10	10	10	10	10	10	10	10	10
	50	10	10	10	10	10	10	10	10	10
	100	3	0	0	1	0	0	10	0	0
	150	0	0	0	0	0	0	0	0	0

Table 4: Number of instances over 10 solved with LM_1, LM_2 and LM_3 within a time limit of 1 hour dissagregated.

7.3 Benders Decomposition Performance.

We run a second set of experiments to analyze the behavior of the stabilization methods for the Benders decomposition. For each instance in Small(n, m, d), we solve the linear relaxation of (LM_3). Figure 5 shows performance profiles in terms of solution time, number of cuts added and number of iterations required to get the optimal solution. The results show that the procedure (CW) is the most efficient for all three indicators. We also observe that the performance of the In-Out stabilization is very sensitive

		$d = 0.2$			$d = 0.5$			$d = 0.8$		
n	m	LM_1	LM_2	LM_3	LM_1	LM_2	LM_3	LM_1	LM_2	LM_3
10	10	8.595	4.154	0.453	15.172	11.116	1.054	27.523	26.137	2.508
	25	18.959	16.127	2.727	35.401	33.095	8.289	38.784	37.068	3.012
	50	31.16	29.397	6.02	45.636	43.65	13.229	55.161	53.301	5.664
25	10	1.059	0.384	0.054	5.436	5.089	2.717	15.696	15.137	4.283
	25	13.969	10.573	4.057	18.084	17.144	9.246	29.891	29.197	9.291
	50	27.201	23.579	13.186	34.495	33.206	21.051	43.912	43.192	17.125
50	10	0.06	0.05	0.02	0.853	0.791	0.506	6.509	6.305	2.879
	25	3.63	3.144	1.872	9.723	9.375	6.765	23.898	23.575	14.362
	50	16.164	15.12	11.711	24.388	23.935	18.845	36.587	36.23	23.578

Table 5: LP gap (%) for small instances.

		$d = 0.2$			$d = 0.5$			$d = 0.8$		
n	m	LM_1	LM_2	LM_3	LM_1	LM_2	LM_3	LM_1	LM_2	LM_3
10	10	0.016	0.016	0.037	0.017	0.025	0.057	0.019	0.045	0.063
	25	0.028	0.036	0.221	0.046	0.114	0.464	0.05	0.122	0.366
	50	0.108	0.149	2.306	0.269	0.761	6.576	0.237	0.826	3.749
25	10	0.01	0.015	0.063	0.018	0.045	0.144	0.021	0.092	0.138
	25	0.052	0.078	0.945	0.08	0.234	2.097	0.079	0.437	1.408
	50	0.503	1.392	34.345	1.692	11.188	79.31	0.681	11.988	41.958
50	10	0.01	0.022	0.116	0.015	0.055	0.215	0.017	0.118	0.24
	25	0.044	0.122	3.235	0.093	0.615	6.797	0.108	1.47	6.972
	50	1.848	8.43	328.971	3.21	52.053	452.955	1.272	62.464	205.513

Table 6: Solution time for small instances in seconds.

to the parameter λ selected. Furthermore, there is no clear dominance in efficiency between the In-out stabilization and the traditional Benders implementation.

In order to improve the performance and scalability of (LM_1), we can generate valid cuts obtained by the Benders Decomposition of (LM_3). The above results suggest that we can improve the gap of (LM_1) by using very few cuts generated by (CW).

7.4 Branch and cut, beating (LM_1).

In order to scale-up our models, we implement a cut generation algorithm with cuts coming from the Benders decomposition of (LM_3). These cuts are added to (LM_1). We tested two implementations. First, we only generate cuts in the root node of the Branch-and-Bound tree. This implementation is named Cut & Branch. A second implementation consider adding cuts in the whole tree, each a fixed number of nodes explored. This implementation is named Branch & Cut. Since only few cuts are necessary to improve the LP gap, we limit the number of cuts added. In this experiment, we set the maximum number of cuts to nm. In the Branch-and-Cut scheme we add cuts each 1000 nodes of the Branch-and-Bound tree.

For each instance in Big(n, m), we solve the Cut & Branch, Branch & Cut and (LM_1) and compare the solution times, limited to 3 hours, and the MIPGAP returned by CPLEX. Figure 6 shows the results separated by cases when $n = 100$ and $n = 500$.

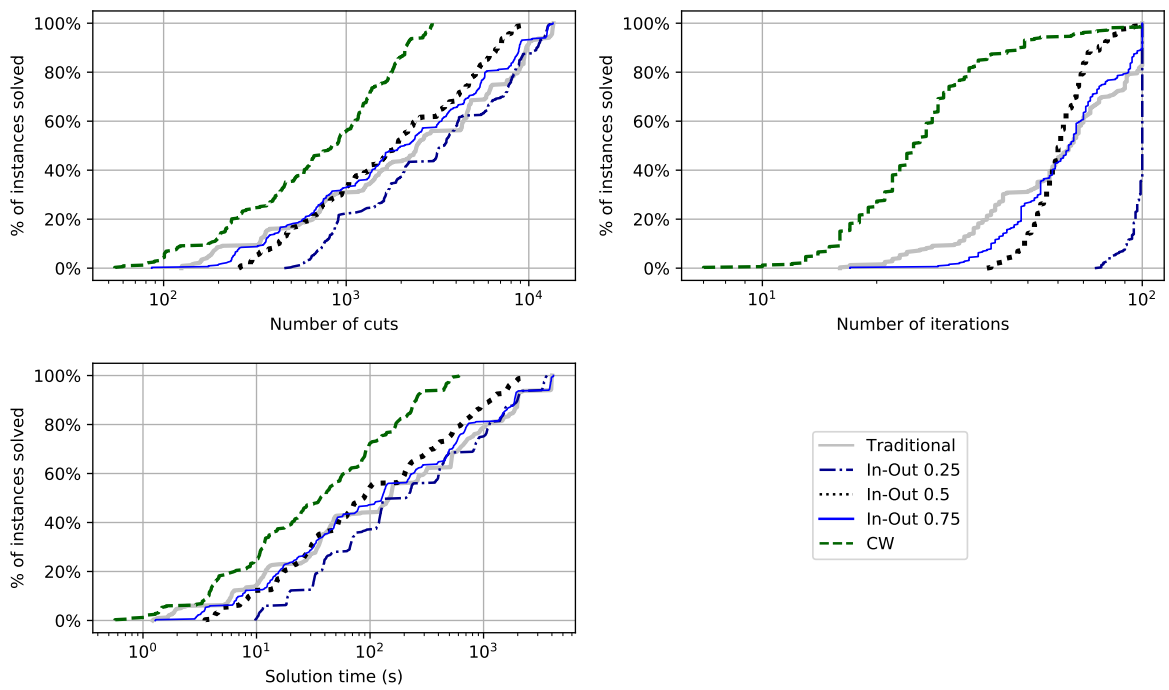

Figure 5: Performance of different implementations of Benders decomposition solving the linear relaxation of (LM_3)

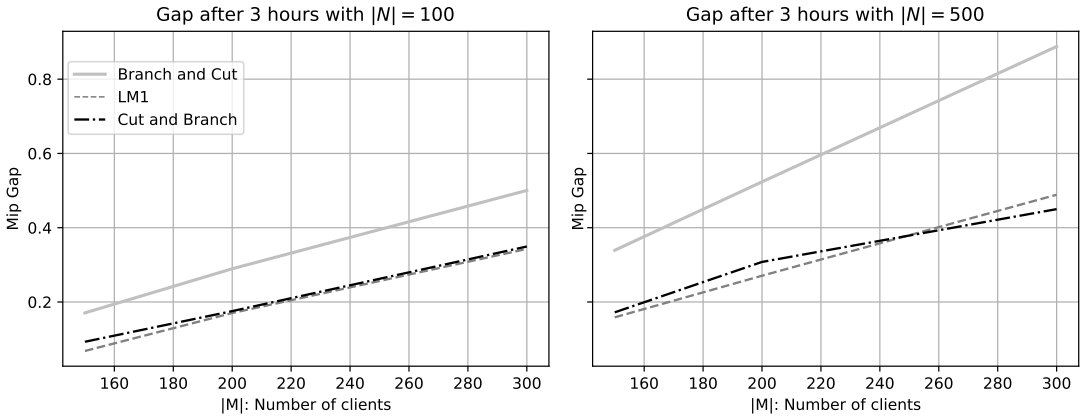

Figure 6: Gap after 3 hours in LM_1, and the different Branch and Cut implementations.

The Branch & Cut implementation is the one with the worst performance. On the other hand, the Cut & Branch implementation can decrease the MIPGAP in the biggest instances. In instances with $n = 500$ and $m = 300$ this implementation reduces by 4% the MIPGAP. One possible improvement is to test using an small limit in the amount of cuts added in our branch-and-cut implementations.

7.5 Impact of preprocessing.

Our fourth set of experiments consists in measuring the improvement due to preprocessing for each instance in $RP(N, M_1, M_2)$. To do so, we measure the solution time with and without preprocessing for (LM_1), and the Cut-and-Branch and Branch-and-Cut algorithms. In order to compute the improvement

we use the Improvement factor

$$\frac{\text{Solution Time without Preprocessing}}{\text{Solution Time with Preprocessing}}.$$

Figure 7 shows the impact of preprocessing for each size of instances. Branch-and-Cut and Cut-and-Branch algorithms were the ones with higher impact in terms of solution time. This impact increases as the instance size increases. For (LM_1) this preprocessing is negligible in small instances. This is due to the fact that the time spent in preprocessing is similar to the time spent in solving the whole problem.

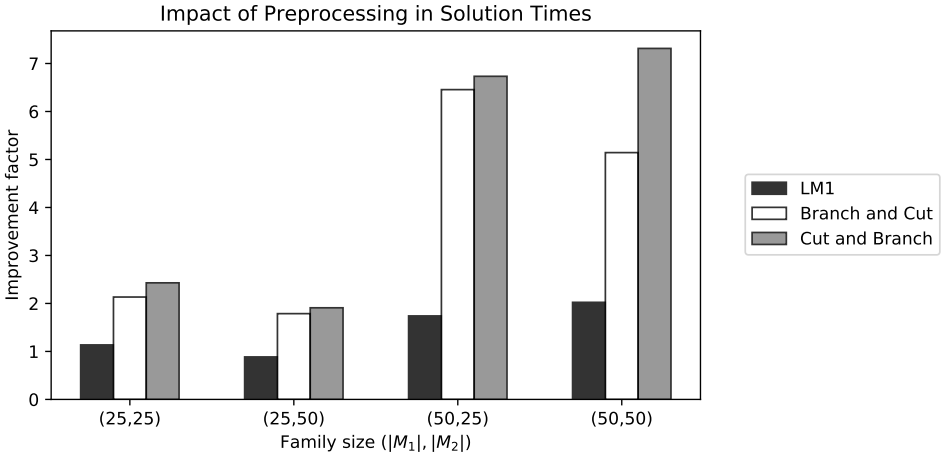

Figure 7: Improvement in solution times for (LM_1), Branch-and-Cut and Cut-and-Branch.

8 Concluding Remarks

In this paper, we have presented three novel MILP formulations for the single-minded bundle pricing problem. After obtaining an mixed-integer non-linear model we derive models (LM_1) and (LM_2) by different linearizations. A stronger model, (LM_3), is obtained by adding RLT-like valid inequalities, that significantly strengthen the gap of the LP relaxation. Some polyhedral results are shown and supported by computational results.

The main bottleneck, at this time, is solving the tighter but significantly heavier LP relaxation in large instances. Here we have studied a Benders Decomposition approach to scale-up our models with different stabilization methods in Branch-and-Cut and Cut-and-Branch schemes. The latter approach generates an improvement of 4% in cases tested in this paper. We believe an improvement can still be obtained by tuning the number of cuts generated.

Finally, we think that, given the combinatorial structure of this problem, some heuristics can be used to improve the optimality gap.

Acknowledgments

Victor Bucarey and Martine Labbé have been partially supported by the Fonds de la Recherche Scientifique - FNRS under Grant(s) no PDR T0098.18
Fränk Plein has been supported by his F.R.S.-FNRS research fellowship.
We would also like to thank Domenico Salvagnin for his valuable comments and discussion about this work.

A (LM_3) reformulation for Rich - Poor instances

$$
\begin{aligned}
(\mathrm{LM}_3 - \tilde{M}) \quad & \max \sum_{j \notin \tilde{M}} \sum_{i \in S^j} s_{ij} + \sum_{j \in \tilde{M}} \sum_{i \in S^j} p_i & \\
s.t. \quad & r_j = \sum_{i \in S^j} s_{ij} & j \notin \tilde{M} \\
& \sum_{i \in S^j} s_{ij} \leq b^j x_j & j \notin \tilde{M} \\
& \sum_{i \in S^j} p_i \leq b^j & j \in \tilde{M} \\
& s_{ij} \leq p_i & i \in N,\ j \notin \tilde{M} \\
& s_{ij} \geq p_i - U(p_i)(1 - x_j) & i \in N,\ j \notin \tilde{M} \\
& \sum_{i \in S^k} s_{ij} \geq r_k - b^k(x_k - x_j) & j, k \notin \tilde{M},\ j \neq k \\
& \sum_{i \in S^k} s_{ij} \leq b^k(x_k + x_j - 1) + \sum_{i \in S^k} p_i - r_k & j, k \notin \tilde{M},\ j \neq k \\
& s_{ij} \geq 0 & i \in N,\ j \in M \\
& p_i \geq 0 & i \in N \\
& x_j \in \{0, 1\} & j \in M
\end{aligned}
$$

B Generating Rich - Poor instances

Given positive integers n, m_1, m_2, such that $n > m_1$, the algorithm returns a set of clients, bundles, budgets and n products such that there exist m_1 relative rich clients and m_2 other clients.

Algorithm 4 Generate relative rich - poor instances

Require: m_1, m_2, n

Step 1: Create set of clients $M = \{1, \ldots, m_1, m_1 + 1, \ldots, m_1 + m_2\}$ and products $N = \{1, \ldots, n\}$

Step 2: Create Budgets. For $j \in \{1, \ldots, m_1\}$ $b_j = U(1000, 5000)$. For $j \in \{m_1 + 1, \ldots, m_1 + m_2\}$ generate $b_j = U(10, 1000)$.

Step 3: Partition the set of products in m_1 components. Each disjoint subset is allocated to one of the m_1 first clients.

Step 4: Generate random bundles for the rest of the clients.

Step 5: For each of the first m_1 clients check if the condition is satisfied. If it is not the case, randomly either reduce the bundles of clients that intersect their bundles, or decrease their budget.

return N, M, S, b with exactly m_1 relative rich clients.

References

[1] Goker Aydin and Jennifer K Ryan. "Product line selection and pricing under the multinomial logit choice model." In: *Proceedings of the 2000 MSOM conference*. Citeseer. 2000.

[2] Maria-Florina Balcan and Avrim Blum. "Approximation algorithms and online mechanisms for item pricing." en. In: *Proceedings of the 7th ACM conference on Electronic commerce - EC '06*. Ann Arbor, Michigan, USA: ACM Press, 2006, pp. 29–35. DOI: 10.1145/1134707.1134711. URL: http://portal.acm.org/citation.cfm?doid=1134707.1134711.

[3] Walid Ben-Ameur and José Neto. "Acceleration of cutting-plane and column generation algorithms: Applications to network design." In: *Networks: An International Journal* 49.1 (2007), pp. 3–17.

[4] Patrick Briest and Piotr Krysta. "Single-minded unlimited supply pricing on sparse instances." In: *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics. 2006, pp. 1093–1102.

[5] Michele Conforti and Laurence A Wolsey. "Facet separation with one linear program." In: *Mathematical Programming* (2018), pp. 1–20.

[6] Amos Fiat and Amiram Wingarten. "Envy, Multi Envy, and Revenue Maximization." In: Sept. 2009. DOI: 10.1007/978-3-642-10841-9_48.

[7] Alexander Grigoriev, Joyce van Loon, Maxim Sviridenko, Marc Uetz, and Tjark Vredeveld. "Optimal bundle pricing with monotonicity constraint." In: *Operations research letters* 36.5 (2008), pp. 609–614.

[8] Venkatesan Guruswami, Jason D Hartline, Anna R Karlin, David Kempe, Claire Kenyon, and Frank McSherry. "On profit-maximizing envy-free pricing." In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2005, pp. 1164–1173.

[9] Ward Hanson and Kipp Martin. "Optimizing multinomial logit profit functions." In: *Management Science* 42.7 (1996), pp. 992–1003.

[10] Rohit Khandekar, Tracy Kimbrel, Konstantin Makarychev, and Maxim Sviridenko. "On hardness of pricing items for single-minded bidders." In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2009, pp. 202–216.

[11] John O Ledyard. "Optimal combinatoric auctions with single-minded bidders." In: *Proceedings of the 8th ACM conference on Electronic commerce*. ACM. 2007, pp. 237–242.

[12] Joyce van Loon. *Algorithmic Pricing*. Universitaire Pers Maastricht, 2009.

[13] Stefan Mayer and Jochen Gönsch. "Consumer choice modelling in product line pricing: reservation prices and discrete choice theory." In: *Operations Research Proceedings 2011*. Springer, 2012, pp. 547–552.

[14] Garth P McCormick. "Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems." In: *Mathematical programming* 10.1 (1976), pp. 147–175.

[15] Margaret P Pierson, Gad Allon, and Awi Federgruen. "Price Competition Under Mixed Multinomial Logit Demand Functions." In: *Management Science* 59 (2013), p. 8.

[16] Fränk Plein. "Analysis of a Problem in Product Pricing, Mathematics." en. MA thesis. Brussels: Université Libre de Bruxelles, 2017. URL: http://hdl.handle.net/2013/ULB-DIPOT:oai:dipot.ulb.ac.be:2013/260879 (visited on 01/07/2018).

[17] Paat Rusmevichientong, Benjamin Van Roy, and Peter W Glynn. "A nonparametric approach to multiproduct pricing." In: *Operations Research* 54.1 (2006), pp. 82–98.

[18] Kalyan T Talluri and Garrett J Van Ryzin. *The theory and practice of revenue management*. Vol. 68. Springer Science & Business Media, 2006.