UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTMENTO DE INGENIERÍA CIVIL INDUSTRIAL

ALGORITHMS FOR BI-LEVEL OPTIMIZATION IN RETAIL: PRICING, ASSORTMENT AND DEMAND COORDINATION

TESIS PARA OPTAR AL GRADO DE
DOCTOR EN SISTEMAS DE INGENIERÍA

SEBASTIÁN ANDRÉS DÁVILA GÁLVEZ

PROFESOR GUÍA:
FERNANDO ORDOÑEZ PIZARRO

PROFESOR CO-GUÍA:
VLADIMIR MARIANOV KLUGE

MIEMBROS DE LA COMISIÓN:
MARTINE LABBÉ
FRÉDÉRIC SEMET
ANDRÉS WEINTRAUB POHORILLE

Este trabajo ha sido parcialmente financiado por ANID-PFCHA/Doctorado Nacional/2016 # 21161328

SANTIAGO DE CHILE
2021

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE DOCTOR EN SISTEMAS DE INGENIERÍA
POR: SEBASTIÁN ANDRÉS DÁVILA GÁLVEZ
AÑO: 2021
PROFESOR GUÍA: FERNANDO ORDOÑEZ PIZARRO
PROFESOR CO-GUÍA: VLADIMIR MARIANOV KLUGE

ALGORITHMS FOR BI-LEVEL OPTIMIZATION IN RETAIL: PRICING, ASSORTMENT AND DEMAND COORDINATION

Esta trabajo de título se centra en las extensiones en modelamiento de la optimización de la línea de productos. Los comercios minoristas se enfrentan a muchas decisiones para optimizar el proceso de satisfacción de la demanda de los clientes. Gracias a las tecnologías actuales, las firmas pueden hacer uso de información diversa y relevante en su toma de decisiones. En concreto, las empresas minoristas pueden incorporar a su proceso de optimización información relativa a las decisiones de compra del cliente. Este tipo de estructura de problemas puede modelarse mediante programación binivel, donde el líder está representado por la empresa minorista y los seguidores por los clientes.

En concreto, en este trabajo se considera la localización de productos de una empresa con múltiples tiendas teniendo en cuenta el precio de reserva de los clientes y el coste de desplazamiento a una tienda. De este modo, los clientes pueden decidir viajar más lejos para comprar un producto más barato o comprar un producto que no está disponible en las tiendas más cercanas. Al tener esto en cuenta, se consigue que la asignación de productos se ajuste más a las preferencias de los consumidores, disminuyendo la rotación de productos y evitando mayores costes de inventario y/o descuentos en los productos. Un segundo problema considerado en esta tesis es un problema de coalición de consumidores, en el que un conjunto de consumidores decide agruparse para comprar productos aprovechando los precios al por mayor. Los productos están disponibles en cestas, y cada cliente tiene un precio de reserva diferente para ellos.

Formulamos ambos problemas como problemas de optimización de dos niveles y exploramos estrategias de generación de cortes para resolver estos problemas hasta la optimalidad para instancias del mundo real de forma eficiente. En particular, para el problema de localización de productos, lo formulamos como un problema equivalente de un nivel y evaluamos métodos de relajación lagrangiana y de generación de cortes para mejorar los tiempos computacionales. Introducimos nuevos cortes para este problema de dos niveles que mejoran los tiempos computacionales. Los experimentos computacionales sugieren que las desigualdades válidas reducen la brecha de la relajación lineal, y que incrustadas en el árbol Branch and Bound mejoran eficientemente el mejor algoritmo actual -la descomposición de Benders- conocido para el caso de una tienda.

Para el problema de la coalición de clientes, el objetivo es realizar la formación de la coalición. Además, se presenta una función general de precios al por mayor y una función de precios escalonados. Se considera una descomposición de Benders para resolver instancias a gran escala. Los experimentos computacionales establecen que añadir el corte de Benders en el nodo raíz tiene el mejor rendimiento.

ABSTRACT OF THE THESIS TO APPLY
TO THE DEGREE OF DOCTOR EN SISTEMAS DE INGENIERÍA
BY: SEBASTIÁN ANDRÉS DÁVILA GÁLVEZ
YEAR: 2021
ADVISOR: FERNANDO ORDOÑEZ PIZARRO
COADVISOR: VLADIMIR MARIANOV KLUGE

ALGORITHMS FOR BI-LEVEL OPTIMIZATION IN RETAIL: PRICING, ASSORTMENT AND DEMAND COORDINATION

This dissertation focuses on extensions of product line optimization modeling. Retail stores are faced with many decisions to optimize the process of satisfying customer demand. Thanks to current technologies, they can make use of diverse, relevant information in their decision making. Specifically, retail companies can incorporate into their optimization process information regarding the customer's purchasing decisions. This type of problem structure can be modeled by bi-level programming, where the leader is represented by the retail firm and the followers are the customers.

In particular, in this paper, we consider the product localization of a firm with multiple stores taking into account the customers' reservation price and the cost of traveling to a store. In this way, customers may decide to travel further to purchase a cheaper product or buy a product that is not available in stores closer to them. By taking this into account, it makes the allocation of products more in line with consumer preferences, decreasing product turnover and avoiding higher inventory costs and/or product discounts. A second problem considered in this dissertation is a consumer coalition problem, where a set of consumers decide to group together to buy products taking advantage of wholesale prices. The products are available in baskets, and each customer has a different reservation price for them.

We formulate both problems as bi-level optimization problems and explore cut generation strategies to solve these problems to optimality for real-world instances efficiently. In particular, for the product localization problem, we formulate it as an equivalent one-level problem and evaluate Lagrangian relaxation and cut generation methods to improve computational times. We introduce new cuts for this bi-level problem that improves computational times. Computational experiments suggest that valid inequalities reduce the linear relaxation gap, and embedded in Branch and Bound tree efficiently improve the best algorithm current -benders decomposition- known to the case of one store.

For the customer coalition problem, the focus is to perform coalition formation. A general wholesale pricing function and step price function is presented. A Benders Decomposition is considered to solve large-scale instances. Computational experiments establish that adds the benders cut in root node has the best yield.

To my family, my friends, especially to my daughter, who is my inspiration to continue to outstand, develop and grow me together to her.

Ackowledgements

I would like to begin this dissertation thankful to all people who have participated in one way or another in this long walk to start my career as a researcher. I thank professors who put their knowledge and passion into their courses and are always available to help and guide their students. DSI administration who always helped me in all the administrative procedures making my stay in the doctorate easier, besides motivating me to finish this process. Also, all the university staff who provide support us in everything necessary to make the day-to-day life more bearable.

To my friends who have given me their support words and advice. To my family, who always support and motivate me to continue my studies and reach my dreams. I thank my daughter's maternal family who always cheers me to progress. Especially to my daughter, although she does not understand that means this path yet, she pays attention to my work and is my light to follow.

To the DSI community. They are all very intelligent, capable, and entertaining people. I am grateful to have had the opportunity to get to know your work, opinions, and scientific rigor. I learned a lot from all of you. Especially to members of my office where I spent great parts of time, first studying together for our courses and after share and supported us in our projects thesis and life. They made this whole process so much easier, I really appreciate it.

To Martine Labbé and Frédéric Semet, for their disposition to incorporate me in thier projects and welcome me, first in Inria-Lille and later in ULB. Thank this I known a great group of researchers and better persons. I will be always grateful for his advice, not only regarding our joint research but also for how he cared about me having an integral experience as a researcher in Lille and Brussels. Without their vision, patience, and generosity when sharing their knowledge, this work would not have the results that it had.

To Fernando Ordoñez and Vladimir Marianov for the guidance and support through these last years. They invited me to work and learn about their interesting research. I learned a lot from their way of working, teams, rigor, and enormous knowledge in the area. I greatly appreciate the opportunity to have you as advisors. Besides, they are enormous people who are always willing to help, advise and support their students in whatever they require. I look forward to continuing to working with them and their research teams, who are not only colleagues but a great family.

Contents

Introduction **1**

1 A Review and background **8**

1.1 Bilevel programming . . . 8

1.1.1 Optimistic case . . . 10

1.1.2 Pessimist case . . . 11

1.1.3 Bi-level programming applications . . . 11

1.1.4 Solution methods . . . 12

1.2 Location problems . . . 14

1.2.1 Bi-level location problems . . . 15

1.2.2 Multi-product location problem . . . 17

1.3 Coalition problem . . . 18

1.3.1 Coalition structure generation (CSG) . . . 19

1.3.2 Payoff Distribution . . . 20

1.3.3 Applications . . . 22

1.4 Solution Method . . . 23

1.4.1 Lagrangian relaxation . . . 23

1.4.2 Cut generation . . . 29

2 Lagrangian relaxation for product line optimization with multiples sites **32**

2.1 Problem definition and formulations . . . 32

2.1.1 Problem description . . . 33

2.1.2 Bilevel model formulation . . . 33

2.1.3 Single-level model formulation . . . 35

2.2 Lagrangian relaxation approach . . . 36

2.2.1 Lagrangian Relaxation LR^1 . . . 37

2.2.2 Lagrangian relaxation LR^2 . . . 37

2.2.3 Lagrangian relaxation LR^3 . . . 38

2.2.4 Feasible solution . . . 39

2.3 Numerical Experiments . . . 40

2.3.1 Random instance . . . 40

2.3.2 Solving the Original problem . . . 41

2.3.3 Lagrangian Relaxation approach . . . 42

2.4 Conclusions . . . 42

3 Cut generation to product line optimisation with multiples site **44**

44section.3.1
3.2 Valid inequalities 45
3.3 Solution methods 47
3.3.1 Existing Benders decomposition method 48
3.3.2 Cut generation methods 48
3.3.3 Problem preprocessing 51
3.4 Computational experiments 51
3.4.1 Instances: description 51
3.4.2 Efficiency of problem formulation and benders decomposition on multi store instances 53
3.4.3 The effectiveness of constraints (3.7) 54
3.4.4 Evaluation of decomposition methods on multi store instances 54
3.5 Conclusions 56

4 Mixed-Integer Programming for Combinatorial Coalition Formation Problem 58
58section.4.1
4.2 Problem definition and formulations 60
4.2.1 Problem description 60
4.2.2 Model formulation 64
4.3 Solution methods 67
4.4 Computational experiments 73
4.4.1 Instances: description 73
4.4.2 Comparison between the different formulations 74
4.4.3 Evaluation of Bender's decomposition 75
4.5 Conclusions 78

5 Conclusion 80
5.1 Contributions 81
5.2 Future work 82

Bibliography 83

Appendix 94
5.2.1 Stabilization root node 94
5.2.2 Primal feasible heuristic 96

List of Tables

1 Reservation price r and travel cost d of the buyers. 5
2 Buyers utility for each price $(r - p - d)$. Unique price strategy (i) are numbers enclosed in squares. Separate problem strategy (ii) are the numbers enclosed in circles. The PLOMS strategy (iii) are the red numbers. 5

2.1 Set notation 33
2.2 Model parameters 33
2.3 Averages GAP, $\underline{GAP}$, $\overline{GAP}$ and t_{LP} (sec) for several test instances. $|\mathcal{K}| = 4$ and $p_j = 5\ \forall j \in \mathcal{J}$. 42
2.4 Average comparison of GAP of the different Lagrangian relaxation formulations with the MIP 42

3.1 Summary of the different formulations 47
3.2 Statistics of generated instances 52
3.3 Comparison of GAP, $RGAP$ and $t(s)$ between the formulations M2. M3 and M4. $|\mathcal{J}| = 4$. 53
3.4 Computational results of model M2. $|\mathcal{J}| = 4$ 53
3.5 Comparison of $RGAP$, TIME and $Rt(s)$ between formulations (M3) and (M3) plus (3.7). $|\mathcal{J}|$ and store capacity p = 5 54
3.6 Comparison of valid inequality (3.7) with default CPLEX cuts and $C\&B$. Store capacity $p = 5$ 55
3.7 Comparison of run TIME of instances [14]. Stores number $|\mathcal{J}| = 1$ 55
3.8 Comparison of GAP and run $TIME$s. Store capacity $p = 5$ 56
3.9 Comparison of GAP and run $TIME$s. Store capacity $p = 10$ 56

4.1 Notation (sets) 61
4.2 Notation (parameters) 61
4.3 Buyers' reservation prices 62
4.4 Price function 62
4.5 Comparison of GAP, $RGAP$ and t(s) between the formulations GPF and SPF. $|\mathcal{K}| = 12$ 75
4.6 Comparison of GAP and t(s) between of Benders Decomposition approach for SPF. $|\mathcal{K}| = 12$ 75

List of Figures

4.1 Step price function representation 65
4.2 Results summary of *B&C* and *C&B* 76
4.3 Percentage of solved instances by number nodes 77
4.4 Percentage of solved instances by time 77
4.5 Percentage of instances by gap 78

List of Algorithms

1 Cut loop stabilization procedure (4.8) 31
2 Lagrangian Relaxation method 37
3 Feasibility heuristic 40
4 Cut generation 50
5 Greedy Algorithm for Solving Problems (4.8) 71
6 Greedy Algorithm for Solving Problems (4.9) 71
7 Cut generation 95
8 Greedy Algorithm for Solving Problems (4.8) without constraint (4.6) 95
9 Greedy Algorithm for Solving Problems (4.9) without constraint (4.6) 96

Introduction

The marketplace is where agents bargain to deal with transactions of goods or services, usually through monetary resources. Nowadays, the marketplace can occur both in a physical and digital environment with different challenges for its participants. The first is the conventional marketplace, in which a company must decide what products and how many to allocate in a specific store and what will be the price of every item in a competitive environment [54]. Simultaneously, the customers need to move -or travel- to purchase the desired items, perhaps in several stores, where they can adopt a comparing-shopper behavior, considering the investment in time, transportation, and the actual price to be paid for the products at the stores [46]. We can assume that the customers have different preferences for the products, depending on variables such as tastes, quality, utility, performance, durability, distance to store, price, etc. Additionally, those preferences change based on people's age, gender, store location, social-economic, education, among others. The aforementioned interaction of factors partly depicts the existence of locations where many clustered companies -or brands - offer substitute and complementary goods, competing with other retailers; this increases the offer and demand at that marketplace [92]. From the point of view of the firm, the decision of assortment and pricing must adapt to the customers' preference, since a correct decision can be decisive in decreasing the inventory cost and the need for price discounts, encouraging the stockpile flow. Additionally, a scenario where the store price exceeds the reservation price (a theoretical maximum price that a customer is willing to pay on a product) is possible. In that case, if the customer can not purchase the product, it decreases the amount of prospects, and if pricing becomes underestimated, a greater number of buyers can be obtained, although the process would profit less than expected.

On the other hand, the digital format has its own challenges; first, it is necessary to design a simple and attractive marketplace for customers with a diversity of products, brands, prices and qualities; user-friendliness is also relevant and the possibility to apply different search filters that allow a correct matching for categorization; and multiple payment methods. This ability generates some of the digital marketplace advantages, which are the pricing and assortment, which can be customized to each user's needs, and the ability to use historically recorded data from the millions of interactions among the users on the digital marketplace platforms. Therefore, companies can collect diverse customer profiles and offer a particular assortment and pricing for everyone. Additionally, the possibility of diverse designs for pricing and assortment in these platforms is cheaper and more likely to yield better results than in the conventional marketplace. Furthermore, the digital platform offers a new opportunity for clients. From their perspective, these methods open new and favorable tools, such as simpler and faster item comparison, which makes it possible to select products to buy from anywhere in the world. Another favorable tool of digital platforms is the ability of buyers to associate to have access to wholesale prices, improving the marketplace performance. With the rise of social media, the opportunity of setting customer profiles appears as a challenge for retailers for the coming decades, and this can generate solid benefits for brands, reducing marketing costs and being able even to optimize the logistics involved.

In the two situations presented above, deciding the location and pricing of products and taking into account the ability of clients to group in the pricing decisions, pricing and assortment are significantly related decisions in the objective of maximizing net profit for the retail. In particular, when there is a competitive market with a dominant company, if there is a limit in pricing supremacy, the leading brand can increase the assortment benefits from the cost advantage (i.e., by reducing the unit cost a product). Whereas the firm that does not have a noticeable authority for pricing can reduce the assortment to balance the cost-saving policies. In addition, in the case of an outstanding retailer with a large market share, the brand can reduce the assortment when the products experience a highly asymmetric popularity situation, assuming the firm focuses on the most fruitful segment [4].

From a mathematical perspective, the seller and the buyers' interaction can be modeled by bi-level programming. In a bi-level programming problem, two agents make decisions that impact their individual private interests. In this setting, an agent called the leader makes its decisions first. Then, the second agent, known as the follower, reacts optimizing its own private interests taking into account the leader's decision; therefore, the leader must optimize its objective, assuming the follower will respond with its best reaction. In the marketplace context, the point of sale –or store- is the leader and buyers –or customers- are the followers [27]. In the specialized literature, diverse cases have been modeled through bi-level programming: e.g., pricing of multiple products, involving a unit-demand and buyers from a preferential rank [28]; location store problems under a buyers' comparison-shopping behavior [46], location store problems assuming varied buyers' purposes to travel in duopoly context [92] and an assortment of the products influenced by the buyers' preferences for a unit-demand environment [14], among others.

This thesis addresses two different problems: the challenges of assortment at the physical marketplace and the buyers' grouping on a digital marketplace. The first focuses on the Product Line Optimization with multiple stores (PLOMS), where a brand –or company- owning several stores and products has determined the assortment of the products to the stores considering the customer's preferences, thus reducing a virtual stockpile flow problem. The second deals with the Combinatorial Coalition Structure Problem (CCSP), which consists of a coordinator that decides how to group a set of customers that want to maximize their outcomes in a marketplace where a brand offers a product or good that could be complementary or substitute, in bundle formats.

Decomposition methods use the problem structure to decompose them into simpler optimization problems. optimization problems can decompose into different types of subproblems, depending on which constraints are removed, some being harder than others. Lagrangian relaxation relaxes constraints using Lagrangian multipliers, which leads to obtaining simpler problems. The Lagrangian relaxation finds the Lagrangian multipliers that provide the tightest relaxation in an iterative process that alternates between updating the Lagrangian multipliers and solving the simpler model until a stop criterion is reached. Moreover, some problems consider a single constraint set connecting two variables, say x and y. A specific cut generation strategy can be used to address problems with this structure. We consider a partition where variable x is kept in the first stage or master problem, while variable y is part of the subproblem. Each iteration consists of 1) solving the first stage problem, 2) using this solution to solve the subproblems, 3) generate cuts from the optimal solution to the subproblem, 4) add the generated cuts to the first stage problem in order to restart the process until some stop criterion is reached.

This thesis introduces a bi-level programming model to PLOMS that can be collapsed in a single level programming. To solve large-scale instances, we considered three approaches: Lagrangian relaxation, bender decomposition and cuts generation. For the CCSP, a mathematical programming model is presented that considers two types of pricing functions: an arbitrary decreasing price for an increasing quantity and a ladder price. For large-scale instances, a benders decomposition approach was applied to the ladder price function case.

Product Line Optimization with multiple stores

Stocktaking and product obsolescence costs have increasingly become significant in the retail industry. Reducing these costs requires a fast circulation of the inventory, which depends, in turn, on a good knowledge of the target customers' preferences. In contrast, when the products exhibited in stores do not match these preferences, a slow inventory replacement could trigger a markdown to get rid of the older inventory and free storage capacity. Indeed, matching the customer's preferences to the right products, sizes, colors, and brands is essential to avoid long-lasting storage of products that are not successful in terms of sales.

We address these issues in the case of a retailer which owns several stores and aims to select, for every branch, a subset of imperfect substitute products, i.e., products that serve the same customers' needs but differ in colour, appearance, flavor, design or some other attribute. A customer may have different reservation prices for these products but would purchase at most one of them. Such a subset must be optimal to fit the space restrictions and the neighboring purchasers' preferences, which can be included as a smart sale strategy. According to this, we allude to a JCPenney executive who stated that "assortments, allocations, discount pricing and other measures of the sort are all linked and optimized together at his company" [57].

The items assortment problem has attracted much attention. In [61], the authors introduce two formulations to support a retailer decisions on the composition of a set of similar products, using heuristic algorithms to solve the problems. A review of the literature on product selection is offered in [62], discussing cannibalization between products, different objectives, and buyers' classification, according to their purchasing power. They state that, to that date, none of the models had practical application. They also focus on the line of products from a manufacturer, even though the work applies to a retailer. The models require knowledge of the allocation of the products, considering the type of consumers in attribute space. The exhibition and storage room is not an issue here. In [12], a ranking-based formulation is presented for the Product Line Design (PLD) problem where the utilities generated by every customer are entered in the formulation as a constraint. In [94], a comparison between different heuristics for the PLD problem is performed. The authors present a new ranking-based formulation and solve it with Lagrangian Relaxation. These previously mentioned approaches [12] and [94] are compared in [14], which also introduces a strong formulation for [12]. A *Benders Decomposition* (BD), together with an efficient algorithm, is applied to solve the example, where the master problem defines what the available products are, and a separable slave problem solves each purchaser's decision. The efficiency of the BD is evaluated on examples with synthetic data, producing good results.

This study has not considered the case where the company has multiple stores. In the multi-store case, the option is to repeat the assortment in every store or solve a specific problem for each store. However, given that buyers' preferences could depend on their location, it is interesting to study a multi-store model. Moreover, in a multi-store case, it is possible to incorporate the cost travel because buyers could travel farther to find a better price or search for the product of greater preference. Therefore the firm must decide in which stores to allocate the product and at what price to maximize the profit.

We illustrate the advantages of the PLOMS model that decides the locations and prices of products in the multi-store case with a simple example, detailed in Tables 1 and 2. The PLOMS model is compared with (i) a unique price in all stores, (ii) a separate problem for each store, where prices are set subject to the buy decision of the closest set of customers and (iii) the PLOMS model. The example considers one product, three stores with a single capacity, and six customers which have reservation prices and take into account the travel cost. In this instance there are three possible prices: 110, 100 and 90. The reservation price and travel cost of buyers is presented in Table 1 and the buyers utility for the different prices is in Table 2. The pricing in every store and earnings to the company obtained by adding the prices of the items purchased, according to each strategy are the following: (i) for the unique price strategy a price of 90 for all stores gives a profit of 540, (ii) the separate problem strategy gives a price of 100 in both store m1 and the m2 and a price of 90 in m3 gives a profit of 570, and (iii) the PLOMS strategy sets a price of 100 in m1, 110 in m2 and 90 in m3 giving a total profit of 580. Notice in this example that a sub-optimal solution is obtained either when ignoring the buyer decision as in (i - unique price) or when only the buy decision of closest buyer as in (ii - separate problem). This example shows the advantage of considering the allocation and pricing of assortment in every store in a single multi-store model.

Combinatorial Coalition Structure Problem

While the PLOMS addresses an important challenge in a traditional marketplace, the marketplace has been evolving into digital platforms, opening new opportunities and greater challenges for the companies. For instance, the latter has defined new bargaining schemes, like different types of auctions offered in websites. An auction can be defined as a public sale, where –in general- a seller offers a product or service and every buyer bids in order to obtain that product or service, depending on certain rules that are well-known for the participants; an auctioneer chooses the winning purchaser and the value that he or she must pay. When buyers have high bargaining power, for example, if they could demand a large volume of items a reverse auction to obtain the best price could take place. In this scenario, the buyer announces the purchasing volume; each seller bids the price, and then, the auctioneer chooses –under specific rules known for all- who is the winning bidder and the purchasing price.

	d			
	m1	m2	m3	r
c1	24	2	31	113
c2	10	9	23	118
c3	7	16	30	115
c4	8	22	6	107
c5	15	17	7	116
c6	13	24	5	104

Table 1: Reservation price r and travel cost d of the buyers.

	110			100			90		
	m1	m2	m3	m1	m2	m3	m1	m2	m3
c1	-21	$\color{red}{1}$	-28	-11	$\textcircled{11}$	-18	-1	$\boxed{21}$	-8
c2	-2	-1	-15	$\color{red}{8}$	$\textcircled{9}$	-5	18	$\boxed{19}$	5
c3	-2	-11	-25	$\textcircled{\color{red}{8}}$	-1	-15	$\boxed{18}$	9	-5
c4	-11	-25	-9	-1	-15	1	9	-5	$\boxed{\textcircled{\color{red}{11}}}$
c5	-9	-11	-1	1	-1	9	11	9	$\boxed{\textcircled{\color{red}{19}}}$
c6	-19	-30	-11	-9	-20	-1	1	-10	$\boxed{\textcircled{\color{red}{9}}}$

Table 2: Buyers utility for each price $(r - p - d)$. Unique price strategy (i) are numbers enclosed in squares. Separate problem strategy (ii) are the numbers enclosed in circles. The PLOMS strategy (iii) are the red numbers.

Consequently, the reverse auction model allows better bargaining from the purchaser, which can occur when the buyer requires a large demand amount of items or goods. Therefore, if many different buyers wish to obtain identical items, it would be beneficial to coordinate a single larger purchase to reach the cheaper price per unit, rather than buying in separately [21]. The evolution of e-commerce has improved the buyers' grouping through platforms that gather the clustered buyers' preferences and divide them into coalitions so as to make a single purchase, which reduces the cost per unit.

In this sense, the combinatorial reverse auction or group-buying problem involves a mixture of varied bids on different combinations of items, according to their personal preferences [21]. If the buyers can buy several items at once, this enables them to purchase a bundle of items from the lowest selling price [71]. The Combinatorial Coalition Formation Problem (CCFP) is the problem of deciding how to group buyers with individual preferences to purchase bundles so that the maximum social surplus is obtained [90].

The focus of this thesis is to solve the CCFP efficiently, i.e., how to form a group of buyers when they buy in a bundle. A CCFP was studied in [89] where buyers pay for a bundle of products that can be complement or substitute; buyers have reservation prices established for bundles, and there exists a discount in the price of items according to the purchasing size. Hence, in this situation, the coalition is for a bundle and not for a single item. To the best of our knowledge there is no evidence of an efficient mathematical model to solve this problem. For this reason, we suggest two formulations: integer programming for the case of a non-increasing general price function and mixed-integer programming for the case of a ladder cost function. We study the use of benders decomposition to solve large-scale instances of the second formulation. Furthermore, to prove the performances of the methods, we conducted computer experiments.

Contributions

This work aims to provide novel mathematical programming approaches for two relevant issues for the marketplace: the optimal allocation and pricing of products for a firm with multiple stores considering the buyers' preference, and optimal buyer group formation to take advantage of wholesale prices for bundle purchases. We remark that despite this thesis presenting diverse novel mathematical formulations to situations mentioned above, this thesis's main contributions are algorithmic.

For the first problem, we propose a Stackelberg game formulation for the multi-store case with travel cost (PLOMS). Here the firm is the leader, and the buyers are followers. This single-leader multi-follower game can be collapsed into a single level integer programming formulation which, in the case of only one store, is equivalent to the model proposed in [61]. Additionally, we adapt a family of known valid inequalities that appear in [118] for the facility location problem with buyer's preference. We proposed the following decomposition methods to solve this PLOMS

(i) Lagrangian relaxation: where the hard constraints relaxed are: the preference constraint, the buy only one product constraint and linking constraint;
(ii) Benders decomposition: where extend the approach present in [14] for the multi-store which is used as a benchmark; and
(iii) Cut generation: where both a cut and branch (C&B) and branch and cut (B&C) approaches are investigated on how to add the valid inequalities efficiently.

Our extensive computational experiments show improvement over the current state of the art using the C&B approach.

For the CCFP we introduced a nonlinear model when there is a non-increasing price function as a function of quantity. Given the buyers are discrete, we can use an approximate piece-wise linear function to transform it into linear integer programming. Moreover, a price function in the form of a decreasing ladder often represents wholesale prices that can be linearized through incremental [40] and multi-choice formulation [121] derived from mixed-integer programming, besides a strengthened formulation. In order to solve the last model, a Benders decomposition is presented, where the binaries and variables continuous variables are contained into the master problem and slave problem, respectively. The slave problem is separable in $|\mathcal{K}|$ sub-problems, where $\mathcal{K}$ is the products set. In order to generate the cuts, the integer solution of the slave problem is analytically obtained while its linear relaxation with an exact algorithm is solved. To add the cuts only to the root node (C&B) or along to the branching tree (B&C) are studied. Computational results for large-scale instances show that the (C&B) obtained the best results.

Thesis Structure

This doctoral thesis opens with Chapter 1, focusing on the literature review about bi-level programming for localization problems and its main solution methods. In Chapter 2, the PLOMS problem and a Lagrangian relaxation approach to solve it are described. The results show that while these Lagrangian relaxation approaches efficiently solve the sub-problems, it is possible to reach maximum optimality. Chapter 3 presents valid cuts for our topic and an adaptation of the Benders decomposition used for the single-store instance. The results are compared to a generation of cuts, and the results prove that it produces a better result for the multi-store case. In Chapter 4, the CCFP explains mathematical programming stage.

Two different types of the cost function are addressed: a general decreasing function and a decreasing staircase function. For the second type of cost function, a Benders decomposition can be considered. The results show that it is possible to solve large problems through this approach. Finally, relevant conclusions, potentially associated works and future research topics are shown in Chapter 5.

Chapter 1

A Review and background

In this chapter, a revision of the literature performed in this doctoral research is summarised, providing an insight background to aboard the problem treat in chapters follows. The Section 1.1 presents a general perspective to *bi-level programming*, such that the main theoretical results, applications, and solution methods. The Section 1.2 presents some models to location problems that can be developed since bi-level programming. In the Section 1.3, the general concepts about the coalition problem are described. Finally, the last Section 1.4 boards the solution methods to the large-scale used in the chapters following: *Lagrangian relaxation* and *cut generation*.

1.1 Bilevel programming

In different real-life situations, the decision of two agents faces each other and can be opposed. In certain cases, these agents have an asymmetric relationship, which means a certain hierarchy exists between them. Therefore, an agent named *leader* can decide before the other agent known as a *follower*. Then the follower makes his decision considering the decision made by the leader. Such a situation can be modeled through bi-level programming. In this case, part of the constraints of mathematical programming is another optimization problem. Bi-level programming has been used in different contexts, such as balance in network design, prices in networks, and energy distribution. The following presents the key concepts and the main applications of Bilevel programming.

Suppose there are two decision-makers denoted by player P1 and player P2 where each player's decision affects, in some way, the decision of the other player. The objectives of both players are opposed, and there exists an asymmetric relationship between them. Then, the game is played sequentially, which means player P1 makes the first decision, then player P2 watches what P1 did and makes his decision.

The situation raised above exposes a phenomenon that is replicated in many real-life situations, such as environmental policies, [10] design of transport networks [25], electricity market [66], competitive in a duopoly market, in security strategies, among others. The first to pose this situation mathematically was Stackelberg in 1934 [123]. In that work, he exposes a situation where two companies, a leader and a follower, sell a homogeneous product sharing the same market. They must say how many products each enter the market. Decisions are made sequentially, with the leader deciding first. This problem is currently known as Stackelberg's game and is part of what is known as Bi-level Programming.

A bi-level Programming Problem (BLPP) can be defined as a mathematical program where part of the constraints of an optimization problem, known as the upper-level, is another optimization problem known as the lower-level [110]. According to this definition, each level has its objective function and set of constraints, i.e., each level represents the optimization problem that each agent must solve. In general, it is not symmetrical between the two levels since the upper-level has complete information about the lower-level problem (otherwise, it is said that there is uncertainty). In comparison, the lower-level only observes the leader's decision and thus optimizes its strategy. Commonly the objectives of a private agent are opposed to controlling authority.

There are certain advantages of formulating certain situations through a bi-level model and not by a single level. In [114] mention three main advantages of bi-level formulations compared to a single-level

(1) can be used to analyze two different objectives at the same time, which can even be conflicting during the decision-making process.
(2) in practice, the bi-level multi-criteria decision-making method can better reflect this situation.
(3) can explicitly represent mutual action between two players.

From a game theory point of view, BLPP can be considered a game where two players decide according to a hierarchical order. The first player, called the leader, decides his strategy and communicates to the second player, known as the follower. Then given the leader's choice, the follower chooses his best response. Therefore, the leader's task is to determine the best decision, considering the follower's best response [79]. A general formulation of a bi-level program is1:

$$\begin{aligned} \min_{\substack{x_u \in \mathcal{X}_u,\\ x_l \in \mathcal{X}_l}} \; & F(x_u, x_l) \\ s.t.\; & G(x_u, x_l) \leq 0 \\ & \min_{x_l \in \mathcal{X}_l} f(x_u, x_l) \\ & \quad s.t.\; g(x_u, x_l) \leq 0 \end{aligned}$$

where $x_u \in \mathbb{R}^{n_1}$ are the *upper-level variables* and $x_l \in \mathbb{R}^{n_2}$ are the *lower-level variables*. Similarly, the function $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ is the *upper-level objective function* and $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ is the *lower-level objective function*. Meanwhile, $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_1}$ and $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_2}$ are the *upper-level* and *lower-level* constraints respectively.

Definition 1.1.1 (Inducible Region) Let $\Psi : R^{n_1} \times R^{n_2}$ be a map set of values,

$$\Psi(x_u) = \left\{ x_l : x_l = \arg\min_{x_l \in \mathcal{X}_l} \{ f(x_u, x_l) : g(x_u, x_l) \leq 0 \} \right\} \tag{1.1}$$

that represents the set of optimal solutions of the lower-level; then we can define the Inducible Region as1:

$$IR = \{(x_u, x_l^*) : G(x_u, x_l^*) \le 0,\ x_l^* \in \Psi(x_u)\}$$

In general, we consider the assumption that for each upper-level solution, a lower-level solution exists1: the set $IR \neq \emptyset$.

The main mathematical features of BLPP are:

- Let $x_l(x_u)$ be the implicit way of representing the lower-level variable. If $x_l(x_u)$ can be represented explicitly, then the BLPP problem can be transformed into a single-level. In general, this is not possible.
- If the optimal solution of the relaxed problem (without the lower-level objective function f) is an IR point, then the solution is optimal for the BLPP.
- BLPP is inherently difficult to resolve, being typically non-convex and non-differentiable. BLPP is a strongly NP-hard problem even for the simplest case, such as bi-level linear programming [78].
- Even if F and f are continuous and G, g is compact, no solution can be found.

In case there are multiple optimal solutions for the lower level, the higher-level solution is ambiguous. Two possibilities to solve this can be considered according to the behavior of the follower that we assume. The follower can have two behaviors1: cooperative or non-cooperative. In the first one, the leader selects the best solution concerning the follower's decision; this case is optimistic. On the other hand, if the leader has no chance to influence or conjecture about the follower's decision, he can assume that the follower will select the worst solution concerning his objective function, then the leader must lessen the risk of such selection. The latter is known as the leader's pessimistic case and is more complicated than the optimistic one because, in general, a *min-max* problem must be solved.

1.1.1 Optimistic case

In this case, given a set of multiple optimal solutions of the lower-level, the leader expects the follower to play cooperatively, that is, select the optimal solution from the set $\Psi^o(x_u)$ that obtains the best value of the objective function F. The follower decision function can be expressed as follows.

$$\Psi^O(x_u) = \underset{x_l \in \mathcal{X}_l}{\arg\min}\{F(x_u, x_l) : x_l \in \Psi(x_u)\} \tag{1.2}$$

Hence the lower-level can be replaced by:

$$x_l = \Psi^O(x_u) \tag{1.3}$$

Given $\Phi = \{(x_u, x_l) : G(x_u, x_l) \le 0,\ g(x_u, x_l) \le 0\}$ we can ensure an optimal solution for the optimistic case according to the following Theorem:

Theorem 1.1.1 If the functions F, f, G and g are smooth enough, if $\Phi \neq \emptyset$; and compact, and the Mangasarian-Fromowitz qualification constraints [1] is sustained at all points, then it can be guaranteed that the bi-level problem, for the optimistic case, will have an optimum as long as there is a feasible solution.

1.1.2 Pessimist case

In this case, given a set of multiple optimal solutions of the lower-level, the leader waits for that follower to play of cooperative way, i.e., that he select the optimal solution of the set $\Psi^P(x_u)$ with the worst value of objective function F of the Leader. Therefore, the decision function of the follower can be express in the following way.

$$\Psi^P(x_u) = \underset{x_l \in \mathcal{X}_l}{\arg\max}\{F(x_u, x_l) : x_l \in \Psi(x_u)\} \tag{1.4}$$

Hence, the lower-level can be represented for:

$$x_l = \Psi^P(x_u)$$

For the optimistic case, a lower-level with a convex problem, it is possible to reduce the bi-level problem a single-level used the correspondent variational inequalities of the lower-level problem [110]. On the other hand, the pessimistic case obtaining is not possible a simple single-level.

We can assure an optimal solution for the pessimistic case with the following theorem:

Theorem 1.1.2 If the functions F, f, G and g are sufficiently smooth, $\Phi \neq \emptyset$; and compact, the mapping of value sets Ψ^P, is semi-continuous for the entire upper-level decision vector, then the problem is guaranteed to have an optimum for the pessimistic case of BLPP.

1.1.3 Bi-level programming applications

Given the characteristics of the bi-level programming mentioned above, this modeling framework has been utilized in several applications [110]. Below is presented some applications example:

- **Stackelberg Security Games1:** There exist two types of players, the Defender, and the Attacker. There are n targets that could be attacked for Attacker, and the Defender has m resources to protect it where $n > m$. There are a set of pure strategies for each player where the Defender covers m of n targets. Later the Attacker observes and attacked one target to optimize his reward. The payment function is determined about pure strategy, i.e., if a target attacked is covered, then the Defender (Attacker) obtained a reward (penalty), in counter case a penalty (reward). The payment expected is obtained when it is extended to a mixed strategy. Therefore, the Defender commits a mixed strategy considering the Attacker's possible responses for maximized his utility. Meanwhile, the Attacker observed the mixed strategy played by the Defender, and then Attacker plays in response.
- **Competitive Facilities Location Problem:** In the classic models of location, there is a firm that sells homogeneous products, and it must select a subset of facilities since a set of locations possesses such that satisfies the demand of the clients that be assumed inelastic. In this case, all the facilities belong to a firm. For the case, a Competitive Facility Location Problem (CFLP) be had two firms that compete to capture a set of clients to meet the objective, in general, are opposite. The client is allocated to the

facility that gives a better utility. In other words, a firm captures a client if its facility is more attractive to any other that has opened its competitor. Thus, we can model CFLP as a Stackelberg game, where the leader decides where to open its facilities to maximize its utility, considered the follower's objective function as the consumers' preferences. Meanwhile, the follower gives the facilities opens for the leader and the consumers' preferences, and it decided where to locate its facilities to maximize its utility.

- **Equilibrium of network design:** The aim is to select the arcs' capacity in the highways urban' network design considered that the congestion produced in each of them produces a delay the users. In this context, the paradox Braess [23] said that the increase in the arcs' capacity (or the number of arcs) could increase each user's delay the network. Due to this effect, the problem aims to find the best Equilibrium between the expanse cost and the trip delay. In this way, network design decision impacts the route selected for the users as in the yield of that network.
- **Pricing problem in a network:** The problem consists of given a network where a subset of arcs has tolls and another subset of arcs without tolls. The tolls are located by an authority. Moreover, there exist users trips for this network. Each user selects the shortest path that minimizes his toll payments. In the basic version there is no congestion. The more sophisticated versions, additionally, incorporate both the network design and other tolls pricing. Besides, it has considered population concerns about time perception and includes discrete choice models [26].
- **Electricity sector:** Every day the generator companies of electricity must select the sale price and present them to the market operator. The operator of the market makes the best purchase. In this way, each generator company aims at maximizing its benefits. The market operator minimizes its cost, determined by the amount of energy purchase, and the marginal price finally chosen. Each agent's decision must satisfy several constraints such as1: to assure the investment and the flow of energy, recourse limits, satisfy the demand, limits in the rate of energy penetration rate, among others [26]. For this case, models have been developen with multiple leaders and one follower.
- **Pricing problems1:** An important decision in retail is the pricing problem. The retail aim is to maximize its utility. For this purpose, the retail must consider the value that the clients give to its products. If omitting this information, the retailer can overestimate the price, obtaining fewer sales and a lower rotation rate of its stock. On the other hand, if the price is underestimated, it can have a higher demand, but the profit will diminish. The firsts works on this line of research were developments in the airline industry, on the subject of ticket pricing.

1.1.4 Solution methods

The following Section shows the main algorithms to solve bi-level programming problems. Given the difficulty of the bi-level formulation, most of the algorithms are for the cases of linear, quadratic, and convex models, of which lineal cases are the most studied [35].

Bi-level programming nonlinear.

- **Karush-Kuhn-Tucker condition:** One of the most common methods to solve BLPP with a linear or quadratic lower-level is to replace that level by its Karush-Kuhn-Tucker conditions (KKT). On the semi-cooperative case, a formulation is obtained of a single-level. For the case of a linear or convex quadratic functions, the Lagrangian constraint is linear. Meanwhile, the complementary slackness condition increases the number of constraints exponentially; therefore, this method tends to fail on a large-scale. A strategy to solve this inconvenience is to use Branch and Bound. First, relax the complementary slackness constraints in the root node and solve. If the solution does not satisfy complementary slackness, two children are built, one when $\lambda_i = 0$ (where λ is the vector of Lagrange multipliers) as an additional constraint; and another with the constraint $g_i = 0$ (where g_i represents the slack variable). As in each node a relaxed problem is solved , each node's optimal solution corresponds to a lower bound of the sub-tree (in the case that upper-level is a minimization problem). Methods based in that scheme have been used for BLPP, [51], quadratic-linear functions [9] and quadratic problem [3, 43].

- **Descent method:** In this case, a descent direction is searched that decreases the objective function of lower-level, saving the new feasible points; a feasible point is a point that belongs to ϕ and is lower-level optimal. However, it is not simple to find a feasible direction. If the lower-level has a unique solution then x_l can be defined as an implicit function of $x_l(x_u)$, and in this way, the problem left only in terms of $x_u \in R^{n_1}$. The main difficulty is the gradient (or sub-gradient) availability in the objective function [37] infeasible points. For this purpose, in [86] an approximation method of this gradient is proposed. Meanwhile, in [106] the problem is formulated without constraints in the upper-level, an auxiliary problem to find the descent direction. In [119] this work is extended to bi-level convex quadratic programming (both objective functions are quadratic and linear constraints).
- **Penalty Methods:** [1, 2] propose to replace the lower-level by the following penalty problem:

$$\min_{x_l} \rho(x_u, x_l, r) = f(x_u, x_l) + r\phi(g(x_u, x_l)$$

  where r is a positive scalar and ϕ is positive if x_l belong the relative interior and $+\infty$ if x_l belongs a the relative bound of $x_l : g(x_u, x_l) \leq 0$. However, each iteration must find a global optimal of the modified problem that is not necessarily sufficiently easier than the original problem [37]. Later, in [75] proposed a method of double penalty where they replaced the objective function of the lower-level by the constraint $\nabla x_l \rho(x_u, x_l, r) = 0$. After each r, this new problem is solved to apply other penalty to the constraints.

Mixed-integer bi-level programming

- **Extreme points:** in the case of linear bi-level programming (L-BLPP), where all functions are linear and the set X is a polyhedron, the set of solutions (in case it is not empty) contains at least a vertex of the constraint set $\Phi \cap X$. A simple method can be devised to solve the L-BLPP based on enumerating the vertices. [29] proposed, for the problem without constraints of the upper-level and a unique solution for the lower-level, an algorithm that explores a decreasing number of lower-level problem bases. Numeric experiments showed that it is relatively slow. Subsequently, [17] proposed the "*K-th best method*". They considered the bases of Φ that order way increase respect the value of the upper-level's objective function. Finally, the k^{th} lowest index corresponds to the method's global optimal finish. Others methods based on this idea are the proposed by

[99, 34, 116].

- **Lagrangian relaxation:** [100] show a method based in Lagrangian relaxation for the case in which both the lower-level and upper-level have continuous and binary variables ($x_u, x_l \in \{0,1\}$ and $x_u^{'}, x_l^{'} \geq 0$). Besides, the upper-level constraints do not depend on the decision variables of the lower-level (i.e., $G(x_u) \leq 0$). The method has three steps. First, it solves the upper-level without considering the lower-level objective function to obtain the upper-level decision variables ($\overline{x}_u$). Second, given the first step's results, it fixes the variables ($\overline{x}_u$) and solves a Lagrangian relaxation of the lower-level. In this case, they replace the constraint $x_l \in \{0,1\}$ by $x_l \leq x_l^2$ obtaining an equivalent formulation. The last constraint is relaxed to get the sub-problem. Then it has a quadratic function objective and linear constraints. Finally, they generate a cut to improve the feasible solution. They improved the current bounds available in known instances of a competitive facilities location using this method.
- **Decomposition techniques:** [103] proposed a decomposition method based in Benders Decomposition to a problem with upper-level MIP and continuous lower-level. The process is decomposed in a Restricted Master Problem (RMP) and Slave Problem (SP). Then, the algorithm begins to fix the values of integer variables and build the current SP. The problem is reformulated to a linear MIP using the KKT conditions. The solution to this problem gives information about the active constraints used for the respective linear problem. The above gives an upper bound of the original problem (in the maximization problem). The dual problem of the current linear problem is used to build a cut to the RMP. Then, the RMP is solved and is obtained at a lower level (in the case of maximization). Later, the optimality conditions are verified. If the RMP feasible solution is updated, the lower-level otherwise, building a new SP gives RMP solution. The key is that the SP is a linear bi-level that is solved through a reformulation using the KKT conditions. In many cases, the process is inefficient and has the wrong approximation for problems of large size. In [50] used is a method for solving a traffic network design problem, while in [31] extends the method used in [103] to solve a facility location problem with equal constraint.

1.2 Location problems

Location problems consists in locating a set of facilities to satisfy the demand of the clients under a constraint set (e.g., facility capacity, each client must be served, each client must be within a certain distance, etc.) such that the cost is minimized (installation costs, travel cost, among others.). The location problem has been widely studied because, from a theoretical perspective, it is an NP-hard problem has different methods have been developed to solve it. It has great versatility to solve problems in both private and public sectors. Examples of these are installing warehouses, production plants, landfills, fire stations, and health centers [44]. The main variants of the location problem are briefly described below.

In general, when the facility's production capacity is large enough to satisfy the total demand, we can skip a capacity constraint. For this reason, the majority of problems have uncapacitated and capacitated versions.

- **P-median:** Consists of minimizing the total distance between demand and their closest facility, given a fixed number of facilities to locate among candidate locations.

- **Single Plant location Problem:** Single Plant location Problem was introduced by Kaufman & et. (1977) Kaufman et al. [80]. The aim is to select a subset of plants from a set of possible locations and allocate a single open plant to the clients that minimizes the location and allocation cost. In the classic version, both location and allocation decisions are taken by a single agent; consequently, the criterion to location and allocation are aligned [30].
- **Set Covering:** In this case, the aim is to minimize the number of facilities open to serve all demand points. To this end, a customer can be served by more than one installation. A version with capacity restriction is studied in [39].
- **Maximum covering**: Unlike the previous case, the number of facilities is limited, and all the demand cannot be served, so it must select a fixed number k of facilities to cover the maximum demand. As in the set covering, the covered clients can be served by more than one facility.

The last two problems have neither fixed cost of installation nor travel cost between facility and clients.

- **Hub location:** In this case, it is considered a network to locate a set of facilities known as hubs. The aim is to find which non-hub node is allocated to each hub such that it minimizes total transport cost between origin and destination into the network, taking into account that trip between hubs has a lower unit cost. Instead of attending each origin-destination pair with a unique arc, a network with hubs provides fewer arcs that link each origin-destination pair taking advantage of scale economies. In a single allocation hub location problem, each non-hub must be attended by a single hub forming a star-shaped network. In the case of the multiple allocation hub location problems, each trip between the origin-destination pair van use more than one path, always through the hubs, decreasing the travel cost. Each origin-destination path has at least a hub, and there exists a discount among all the hub's pairs.

Besides the uncapacitated and capacitated problems described above, there exists stochastic and dynamic versions. In the first, some items have uncertainty, e.g., demand, travel time, among others. So, the uncertainty must be considered in the problem. In the second, it incorporates changes in the time that can impact the change of open/closed facility or the re-allocate of clients.

1.2.1 Bi-level location problems

Competitive facility location problem

In this case, the location problem incorporates the original model, the explicit presence of competition where all agents share the same market. For this purpose, the good or product is considered homogeneous . Hotelling (1929) [70] presented the first notions of the problem considering a linear market with two firms, and the consumers distributed uniformly on the line, been attended by their closest firm.

In general terms, the entry to the market of the agents can be simultaneous or sequential. In simultaneous cases, first, the agents select where to open the facilities, and later, they decide the production level or their prices. FIn the case sequential, a leader firm open its facilities first, then follower firm opens facilities consider the leader's decision.

The clients selection of a facility has been modeled by deterministic or random utility rules. In the first case, the client is always attended by the firm that maximizes the customer's utility, determined by certain attributes, e.g., nearness. In the random utility case, the clients select which facility attend with certain probability [126, 87].

Facilities location with user preferences

If the firm distributes the products to its clients or forces them to be served by a specific plant, the non-competitive models make sense. However, if we consider that customers choose where to get their services, they may want to go to any facility of their preference. Besides, each customer may have preferences by age, tastes, work, among others. For this purpose, models have been established that consider the customers' preference, e.g., the *Single Plant Location with User Preferences* (SPLP). The model was introduced by Hanjoul & Peeters (1987) [64]. In [65] they propose a BLPP formulation that is transformed in different ways to a single-level formulation obtaining in its strongest version the same formulation as Hanjoul & Peeters. [30, 117] propose a new family of valid inequalities used in a Branch & Cut algorithm.

Another variant of this model was developed by Krarup et. (1983) [77] where the possible set of installations is replaced by two levels, the upper one by deposits and the lower one by satellites. It must be decided which deposits and satellites to open and which deposits-satellite to serve each customer. The objective is to reduce costs. This version seeks to take advantage of economies of scale in hierarchical networks' design and operation [55].

[55] considered a two-level uncapacitated facility location problem with single assignment constraints (TUFLP-S) where the upper-level selects the deposits, and the lower-level selects the satellites. Each satellite can be attached to at most one deposit. They use Lagrangian Relaxation along with a local search heuristic to obtain a higher and lower bound. Finally, they embed that solution into a Branch and Bound. To update the multipliers, they use a bundle method. As initial multipliers, they use the dual variables of the relaxed constraint.

Facilities location with partial external financing

In this case, part of the plant's products is transported to an investor with a limited budget. The investor receives the products and can lower the prices in order to maximize the profit. In this way, a bi-level model can be elaborated where the upper-level corresponds to the agent that locates the plant at minimum cost. At the lower level is the investor who seeks to maximize its utility. So, in this case, the leader's problem is an SPLP, and the follower solves a knapsack problem [83].

[83] reduce the problem to a series of SPLP that allows obtaining a lower-level for the leader's objective function. Also, they design an exact algorithm and an approximation algorithm.

1.2.2 Multi-product location problem

The assortment selection problem consists in finding a product subset, and allocating the products in the subset to a limited, integer number of shelves in a supermarket so that revenues are maximized. It is addressed in [127]. The demand for the product depends on shelf space, price, advertising, promotions, and store attraction, among other factors, and a dynamic programming solution method is used. In [42], the authors use conjoint analysis to decide the product assortment, based on each product's fixed and variable costs, as well as its cannibalization effects on other products, rather than storage and shelf space. Each customer segment purchases the product that provides her with the best utility. The problem is solved using a heuristic. Demand substitution means that a consumer prefers to buy a product that is an imperfect substitute of her best choice, rather than not purchasing at all is addressed in [125]. Their problem also considered exogenous demand, supplier selection, shelf space limitations, and inventory management considerations. In [57], the product assortment is optimized, together with the pricing and inventory decisions, not considering either space limitations or preferences variation across the region of interest. They consider multiple periods and customer segments and determine optimal prices and inventory levels for the subset of products in each period. A deterministic utility function depends on the reservation and selling prices and makes customers purchase their best choice. They also do the exercise of calibrating consumers' reservation prices. In [58], the optimal product assortments and pricing for multiple product categories are found when these are complementary (nachos, cheese spread, and guacamole). Consumers can choose to purchase the primary product (nachos) and mix it with complementary categories (a brand of cheese spread or guacamole). Customers choose according to a deterministic utility function.

The problem of assortment planning and pricing in a competitive setting is addressed in [15] using a multinomial logit model for consumers' demand. The paper [95] solves selecting a mutual substitute product line by a retailer who also selects the price among a set of discrete prices. They include limited shelf space and dynamic substitution, making consumers purchase products that are not necessarily their best choice when unavailable. They consider one line of products without differentiating by store and solve it using a genetic algorithm. In [72] the authors optimize shelf-space planning in a store, taking into account products that are substitutes of each other (e.g., different brands) and the effects of not listing products or replacing them with other products, and the effect of these actions on demand for other products. Their main problem is allocating products to shelves and deciding how much space to allocate to each product.

A frequent practice is optimizing shelf space and determining a set of products to be displayed that are not substitutes for each other, i.e., items that consumers could purchase together [33]. Other authors that address the same problem with different variants are [49, 72, 84, 73], who offer a review on the subject. A later review is presented by [85], which states that "no dominant solution has yet emerged for assortment planning, so assortment planning represents a wonderful opportunity for academia to contribute to enhancing retail practice."

Reviews of planning of mutual substitute products assortment can be found in [108], on the planning of product lines, and [97] on retail store operations, including a good Section on product assortment. There exists commercial software, which solves product allocation problems to stores using simple thumb rules of thumb [73].

In [32], markdowns are dealt with. They assume multiple stores owned by different chains, served by a single warehouse over a time horizon during which products can take different discrete prices from a set. Store owners must follow a set of rules to maintain fair competition and inventory turnover. There is just one product, and the demand is stochastic. von Stackelberg pricing was the subject of [24], who assume customers that purchase sets of products and are not capable of computing their exact utilities, as they are computationally bounded.

Somewhat related is the competitive facility location problem [see 45]. However, the von Stackelberg game players are the firms locating their stores instead of firms and customers. For a review of bi-level models for competitive location, [see 5]. Finally, [16], address the problem of product allocation among brick-and-mortar stores and online stores belonging to the same chain. None of the reviewed papers considers customers' geographical distribution, the possibility of customers purchasing at different stores, and the fact that different stores belonging to the same retailer can offer different product lines with possibly different markdowns.

1.3 Coalition problem

Diverse firms sell products whose price depends on the demand. A single customer cannot usually take advantage of this fact because she would need a few units of the product. However, buyers can agree to buy a large number of units and benefit from discounts. Moreover, customers require not a single product, but several. In the last case, a mechanism to reduce prices of several products is to form a coalition, i.e., put together many buyers that wish to buy a similar set of products to do a unique purchase at a lower unit price. Many platforms have been developed to coordinate the buyers with similar taste, assuming the challenges to satisfy members of the coalition. This challenge is known in the operation research literature as a coalition problem that contemplates two-stages: coalition formation called *Coalition Structure Generation* (CSG) and defining the price paid by each buyer, which is the solution of a cooperative game theory.

The process of coalition involves the following steps:

- *Forming a coalition structure* the agents are grouped, which is done endogenously (e.g., through some bargaining procedure) or exogenously (e.g., by a system designer). This procedure is called *coalition structure*. Typically, the goal is to maximize welfare or minimize the agents' incentive to deviate from their coalitions.
- *Solving the optimization problem of each coalition* where the members of a coalition should coordinate their activities such that the coalition's performance is maximized.
- *Dividing each coalition's reward among its members*, where they will need to agree on how to divide these benefits amongst themselves. The distribution to reward must satisfy certain desirable criteria such as *fairness* (where each agent's reward reflects its contribution to the game) or *stability* (where no group of agents can selfishly benefit by forming their own coalition).

In this circumstance, a *solution concept* specifies (i) allocating the agents to a coalition and (ii) how the payoff of every coalition is divided among its members. Following that order, first, we describe the coalition structure problem and later, the problem of the distribution to the reward.

1.3.1 Coalition structure generation (CSG)

The CSG problem focuses on partitioning the agents' set into mutually disjoint coalitions so that the total reward from the resulting coalitions is maximized [102]. The CSG assumes that there is a coordinator that designs the formation of the coalition and has three important characteristics1: (i) the players are goal-directed and short life; (ii) the coordination is between members of the same coalition, but not members of other coalitions, and in general (iii) the organization structure is flat, i.e., there is no hierarchical order between agents [20]. Given its difficult computational treatment, different algorithms as heuristic and exact methods have been developed. Mathematically, it can be defined as follows.

Let $A = \{1, \ldots, n\}$ be a set of agents. The value of coalition $S \subseteq A$ is given by a *characteristic function* $v : 2^{|A|} \to \mathbb{R}$. Without loss of generality, we assume $\forall S \subseteq A,\ v(S) \geq 0$ and $v(\emptyset) = 0$ holds. In [105] it is shown that it is possible to normalize the coalitions values so that all values are non-negative, and that the obtained game is strategically equivalent to the original game. Now, assume a set of allowed coalitions is given as $AC \subset 2^A$. If coalition S is not in AC, we assume the value of $v(S)$ is $-\infty$. Each agent belongs to exactly one coalition, and some agents may be alone in their own coalitions. Considering this, formally, a CS can defined as followed:

Definition 1.3.1 (Coalition Structure) CS^S for agents S is a partition of S into disjoint and exhaustive coalitions, where $CS^S = \{S_1, S_2 \ldots\}$ satisfies the following conditions1:

$$\forall i, S_i \in AC,\ \forall i, j\ (i \neq j),\ S_i \cap S_j = \emptyset,\ \bigcup_{S_i \in CS^S} S_i = S.$$

Another important task is how to define the value of each coalition correctly. For that, the *characteristic function* is typically used. In this case, the value of the coalition only depends on the coalition members. A partition function game is used. The partition function game is given by a pair, (A, v), where A is the set of agents, and v is a function called the characteristic function, that maps each coalition, C, to its value, $v(C)$. Note that when the value of a coalition incorporates externalities, that means the value does not depend only on members of its proper coalition, but also members of other coalitions, and the models can not be represented with characteristic functions. Formally: $v : 2^A \to \mathbb{R}$. The partition function game is given by a pair, (A, w), where A is the set of agents and w is a function called the *partition function* that maps each embedded coalition, (C, CS), to its value, $w(C, CS)$, and $w : EC \to \mathbb{R}$. We will write $w(C, CS)$ instead of $w(C, CS)$ for brevity.

Note that if the characteristic function is super-additive, i.e., $v(S_1 \cup S_2) \leq v(S_1) + v(S_2)$ holds for any two disjoint coalition S_1 and S_2, and $S_1 \cap S_2 = \emptyset$, solving the CSG becomes trivial, because Super-additivity means that any pair of coalitions is better off by merging into one. In fact, the grand coalition (the coalition of all agents) is optimal.

From the point of view of mathematical programming, the coalition structure problem is reducible to complete set partitioning (NP-hard problem) in which there exist a finite set of elements, and every subset of them is associated with a value. The goal is to partition this set into disjoint subsets to maximize the sum of subset values. Currently exists two main approaches to solving this problem: dynamic programming and anytime algorithm.

1.3.2 Payoff Distribution

Besides the formation of CSG, various works have been center on how to distribute the payoff. In this case, the solution concept is based on cooperative games. Unlike the non-cooperative game, in the cooperative game there is no possibility of forcing alliance among agents, although the agents can compete with each other in the coalition to improve their benefits. The payoff distribution will depend on whether exists transferable utility or not. To exemplify the idea consider a marketplace context where there exists a set of buyers who have reservation prices about the items available by a firm and the items have any price function. In the cases of the non-transferable utility, only can be members of a coalition the players with a reservation price greater than the price of the item. In contrast, when there exists transferable utility, the players with reservation prices less than payoff can belong in a coalition if it helps to improve the coalition value, e.g., when there exists wholesale price where the price depends on the size of the purchase.

The more interesting case is when there is utility. Now, there are two possible objectives to be achieved: fairness and stability. The fairness searches that the payoff of agents reflect its contribution to the game. Meanwhile, the stability searches that members of a coalition have not an incentive for forming their own coalition. For fairness, a solution method has been developed known as *Shapley value* [107] For fairness, *Core* [59] present a solution. The problem of determining whether there exists a stable solution in NP-hard [38].

Shapley value

The fairness concept behind the distribution payoff relates to defining a fair payoff for each player, commensurate with each player's contribution to the value of the coalition. Accordingly, Shapley proposed in 1953 [107] a measure to define each player's coalition's contribution known as *Shapley value.* In order to establish this measure, one must first define the *marginal contribution* as follows1:

$$m_i^v(S) = \begin{cases} v(S) - v(S \setminus \{i\}), & \text{if } i \in S; \\ v(S \cup \{i\}) - v(S), & \text{if } i \notin S. \end{cases}$$

where the *gain* of incorporating the player i in the coalition S, is the contribution of player i to the coalition S. Note that the marginal contribution can be calculated in all possible permutations into the coalition. Therefore, it is possible to calculate each player's average contribution, known as the Shapley value. Formal definition as follows.

Definition 1.3.2 The Shapley value on a subset A of $G(N)$ is a solution function ϕ on A whose components $\phi_1(v), \phi_2(v), \ldots, \phi_n(v)$ in the game v on A are defined by

$$\phi_i(v) = \sum_{s1:i \in S} \frac{(s-1)!(n-s)!}{n!}[v(S) - v(S \setminus \{i\})] \tag{1.5}$$

where the sum of the global coalition S containing player i and s generically represents the number of players in coalition S.

The Shapley value can be interpreted as the expected payoff to player i under a randomization scheme where the players' arrival is equiprobable to coalition S. From another point of view, the Shapley value can be calculated as the sum over all coalition S not containing the player i, as presented below:

$$\phi_i(v) = \sum_{s1:i \notin S} \frac{s!(n-s-1)!}{n!}[v(S \cup \{i\}) - v(S)] \tag{1.6}$$

Shapley values are used in a wide range of applications because, in part, it satisfies important properties as

- **Efficiency:** if $f_1(v) + f_2(v) + \cdots f_n(v) = v(N)$ for every v from A; is interpreted as a combination of the feasibility $(f_1(v) + f_2(v) + \cdots f_n(v) \leq v(N))$ i.e., the distribution must be at most the total payoff of grand coalition, and collective rationality $(f_1(v) + f_2(v) + \cdots f_n(v) \geq v(N))$ that summary as distributed all value of coalition;
- **Null player:** function solution f satisfies $f_i(v) = 0$ whenever $v \in G(N)$ and i is a null player;
- **Equal treatment:** if $f_i(v) = f_j(v)$ for every $v \in G(N)$ and every pair of players (i, j) are interchangeable in v;
- **Additivity:** if $f(u + v) = f(u)f(v)$ for every pair of games u and v from $G(N)$;
- **Dummy player:** if $f_i(v) = v(\{i\})$ whenever $v \in G(n)$ and i is a dummy player in v; and
- **Anonymity:** if for every permutation π of N, $f_i(\pi v) = f_{\pi(i)}(v)$ for very game $v \in G(n)$ and every player $i \in N$.

The Shapley value is a unique solution function that satisfies the first four properties, [107].

Core for coalition structure

A distribution payoff stability seeks to ensure that players have no incentive to leave the grand coalition. To understand the concept, we need to define the *kernel.* A vector $x \in \mathbb{R}^n$ is an imputation if it satisfies $x_t \geq v(i)$ and $\sum_t x_t = v(N)$. Thus, formally the kernel is defined as:

Let $i, j \in A$, $x = (x_1, \ldots, x_n)$ is a payoff vector. Define $\delta_{ij}(x)$ to be the maximal excess, with respect to x of a coalition containing i but not j; i.e,

$$\delta_{ij}(x) = \max\{v(S) - \sum_{k \in S} x_k : S \subset A \setminus \{j\}, i \in S\}$$

then the kernel $K(A, v)$ is the set of all imputations x such that, for any pair of players (i, j) either:

- $\delta_{ij}(x) = \delta_{ji}(x)$, or
- $\delta_{ij}(x) > \delta_{ji}(x)$ and $x_j = v(\{j\})$, or
- $\delta_{ij}(x) < \delta_{ji}(x)$ and $x_i = v(\{i\})$.

Definition 1.3.3 (Core for coalition structure) The core for the coalition structure CS^A of all agents A is the set of payoff vectors, where each element (denoted as y) satisfies the two following conditions:

- $\forall S \in AC$, $\sum_{i \in S} y_i \leq v(S)$ (non-blocking condition)
- $\sum_{i \in A} y_i = V(CS^*)$ (efficiency condition)

In effect, an imputation belongs to the core if no coalition has an incentive to separate from the grand coalition.

In addition, y is said to dominate x if $y_t > x_t$ for all $i \in S$ and $\sum_{t \in S} y_t \in v(S)$. That mean that y dominates x if it is best for all members of S and is feasible, i.e., $\sum_{t \in S} y_t \leq v(S)$. Therefore, the *Dominance Kernel* $DC(v)$ is the set of non-dominated imputations. Note that x is dominated by y across the coalition S, so $\sum_{t \in S} x_t < \sum_{t \in S} y_t \leq v(S)$. Therefore, x is not in the kernel then it is possible to find an imputation y that dominates x.

Unfortunately, the core is sometimes empty; i.e., there are no imputations that are not dominated. Also, determining whether there is a stable solution is NP-complete [38]. The first proposed concept of stability was by von Neumann and Morgenstern [96]. They defined a stable solution as a set of imputations such that no imputation in the stable set dominates another stable set, and if every imputation outside the stable set is dominated by an imputation belonging to the stable set. Other solutions such as nucleolus, kernel, and the related bargaining set to focus on minimizing the incentive for a coalition or an individual to separate from the proposed assignment(s) [124].

1.3.3 Applications

Although this thesis addresses the coalition problem from a marketplace perspective, these are not the only applications. There are a wide variety of applications of coalitions problems such as1:

- autonomous sensors can improve their surveillance of certain area;
- virtual power plants can reduce the uncertainty of their expected energy outputs [20];
- cognitive radio networks can increase their throughput, and buyers can obtain lower prices through bulk purchasing [111].
- teamwork formation to perform tasks. Each activity needs certain skills, and each player has a different level for each skill ([115]).

For this thesis, the focus of the coalition problem is the buyer group forming. In this circumstance, there exists a set of buyers and a set of products that buyers wish to purchase. The products have a wholesale price, but every buyer requires a few units. Hence, the buyers, through the coordinator, can form groups to reach a better price. The coordinator recollects all the necessary information (e.g., reservation price, preference, and amount required) to define the coalition structure and distribution payoff according to the criteria of stability and fairness.

A variant of this problem occurs when selling bundles of products known as *Combinatorial Coalition Formation* (CCF). In [88] is presented a CCF where the clients can buy more than a unit per item. In this case, the reservation price is per bundle, and the buyers can buy more of one unit of the item. A virtual reservation price by each item is calculated and then reflected in a bundle price. If the set of sub-coalitions are compatible, i.e., each buyer belongs to all sub-coalition or none, a coalition is formed. Otherwise, they must then calculate the reservation price again and find the sub coalition set until they are compatible. They proved that if the algorithm finds a set of sub-coalitions compatible for linearly decreasing price, the coalition found is optimal. Besides, they calculated the core with the respective payoff for each buyer. Meanwhile, [89] shown a greedy algorithm to (CCF) that incorporate goods complements or substitutes, and heterogeneous preference of buyers. They propose a Coalition for bundles instead of products. The reservation price is for a bundle, not for an item, reflecting if the items are complementary or substitute. The proof can be reduced to Complete Set Partitioning, therefore, is NP-hard. Finally, they propose a cost-sharing that accomplishes stable properties, which establish that buyers pay their reservation price if that is less to price and value greater in another case which establishes to a particular method.

Additionally, others variants to the CSG include shipping cost [112] and location [98], limit stock [74], such attribute in Social Network [22]. Also there could be constraints, for instance, physical limitation, legal banishments, and social relationship [60], social ridesharing [18, 19]. Based on Myerson, consider the feasible set is restricted to a graph. Therefore each coalition is associated with an undirected graph defined over the set of agents, and the coalition is feasible only if the subgraph induced over the nodes in a coalition is connected [18, 122, 19].

1.4 Solution Method

1.4.1 Lagrangian relaxation

Lagrange's method is based on the idea formulated by the mathematician Lagrange in 1785. In optimization models, there are often some *difficult* constraints that make the problems difficult to solve. If the difficult constraints are omitted, the sub-problem obtained can be easily solved. However, the solution to this problem can be infeasible in the original problem. To find a feasible solution to the original problem, we can penalize the subproblem each time the difficult constraints are violated.

Using this idea, let us consider the following problem :

$$\begin{aligned}(P1)\ \ z^* &= \min_{x\leq 0} c^T x\\ s.t. \quad & Ax \leq b\\ & Dx \leq d\end{aligned}$$

where, $c \in \mathbb{R}^n$, (A, b) and (D, d) are matrices $m \times (n+1)$ and $r \times (n+1)$ respectively and z^* is the optimal value of the problem solution (P1). Suppose that $Ax \leq b$ are difficult constraints and $Dx \leq d$ are easy constraints.

We can relax the difficult constraints and penalize them in the objective function in case they are violated. Then the objective function is as follows1:

$$\mathcal{L}(\lambda, x) = \min_{x\geq 0, \lambda\geq 0} c^T x + \lambda^T (b - Ax)$$

Then, given a $\lambda \in \mathbb{R}^m_+$ is to solve the sub-problem:

$$\begin{aligned}(P2)\ \ \phi(\lambda) = \quad & \max \quad \mathcal{L}(\lambda, x)\\ & s.a. \quad Dx \leq d\end{aligned}$$

where the function ϕ is known as the dual function and is determined by the sub-problem's optimal value. Now be a x feasible in (P1), some $\lambda \in \mathbb{R}^m$ and x^* an optimal solution of (P1), then it remains that:

$$\phi(\lambda) \leq \mathcal{L}(\lambda, x) \leq c^T x \text{ y } \phi(\lambda) \leq (\lambda, x^*) \leq c^T x^* = z^*$$

Therefore, we have $\phi(z) \leq z^*$. By setting the value of λ and solving the sub-problem (P2), we get a lower bound for the problem (P1). Therefore, we try to find the best lower bound ϕ^* of z^*. Then, to find ϕ^* we must select the λ that solves the Lagrangian Dual Problem (LDP):

$$\phi^* = \max\{\phi(\lambda) : \lambda \geq 0\}$$

To find ϕ^*, there are different methods such as the sub-gradient method, volume method, descent method, among others.

In case $Ax = b$, then the Lagrange multipliers are unconstrained, and then the dual problem remains as:

$$\phi^* = \max_{\lambda\in\mathbb{R}^m}\{\phi(\lambda)\}$$

An important lagrangian relaxation theorem for integer problems was developed by Geoffrion [56]. This theorem is based on the convex hull, denoted by $conv((X))$, of points in $\mathbb{X}$ where the $conv(\mathbb{X})$ is the set of all convex combinations of points in (X), i.e. :

$$conv(\mathbb{X}) = \{\overline{x} : \overline{x} = \sum_i \alpha_i x^i, \sum_i \alpha_i = 1, \alpha_i \geq 0, x^i \in \mathbb{X}, \forall i\}$$

Theorem 1.4.1 (Geoffrion, 1974) Let (P1), (P2), and (LDP) be the integer version described above. Let ϕ^* be the optimal solution value of the dual Lagrangian problem. Then:

$$\begin{aligned}\phi^* = \quad & \min \quad c^T x\\ & s.a. \quad Ax \leq b\\ & \qquad\ \ x \in conv(\mathbb{X})\end{aligned}$$

The above theorem can be said that (LDP) is the convexification of the entire problem (P1).

Lagrangian Relaxation vs. Linear Relaxation

Another relaxation that we have studied is linear relaxation (LP). As we previously studied linear relaxation, it also gives us a measure of the integer problem. So a coherent question is which of these relaxations is better?

Let us first look at the Lagrangian relaxation:

$$\begin{aligned}\phi^* &= \max\{\phi(\lambda) : \lambda \geq 0\}\\ &= \max\{\min\{c^T x + \lambda^T(b - Ax) : Dx \leq d, x \geq 0 \text{ and integer}\}\}\\ &\geq \max\{\min\{c^T x + \lambda^T(b - Ax) : Dx \leq d, x \geq 0\}\} = Z_{LP}\end{aligned}$$

To see the latter let us consider the dual of the internal problem for which $\mu \in \mathbb{R}^k$ is the multiplier of the constraint $Dx \leq d$, and the dual of the dual.

$$\begin{array}{llcllcll} \min & c^T x + \lambda^T(b - Ax) & \text{Dual} & \max & \lambda^T b + \mu^T d & \text{Dual} & \min & c^T x \\ s.a. & Dx \leq d & \rightarrow & s.a. & \lambda^T A + \mu^T D \geq c & \rightarrow & s.a. & Ax \leq b \\ & x \geq 0 & & & \lambda, \mu \geq 0 & & & \end{array}$$

We can observe that the linear relaxation (Z_{LP}) of the sub-problem ($\phi(\lambda)$) is a lower level of the Lagrangian relaxation ($\phi^* \geq Z_{LP}$). If the linear relaxation gives us integer solutions, then it is fulfilled by equality (i.e., both the optimum value of Lagrangian relaxation and the optimum value of linear relaxation are equal ($\phi^* = Z_{LP}$). This case is known as the property of integrality (linear relaxation obtains integer solutions). Given the above, we can conclude that the level that gives us Lagrangian relaxation is at least as good as linear relaxation.

In general, we can say that for the case in which the variables are integer, the Lagrange method can be more effective than linear relaxation in problems that have complicated constraints [57] due to:

- LDP can take advantage of a special sub-problem structure to separate the sub-problem.
- The number of constraints is reduced in the sub-problem (compared to the original problem).
- ϕ^* can be tighter or at least as good as Z_{LP}.
- A workable solution can be built from the solution of the sub-problem.

Finally, three fundamental questions must be answered when performing a Lagrangian relaxation:

(1) What restrictions should we relax in order to get the best score?
(2) What are the best values of Lagrange multipliers?
(3) How can we obtain feasible (incumbent) solutions to the original problem?

The first question depends on the structure of the problem. For example, with certain relaxations, we can separate the original problem into two easier ones. It also depends, if when relaxing the constraints, it obtains a subproblem with an integrality property (it should not have this property because I can find a better level than the linear relaxation). Also, with which relaxed constraint is obtained, an easier subproblem.

The second question depends on the methods we use to solve the LDP. In this work, we focus on the subgradient method.

Finally, the third one again depends on the structure of the problem; however, heuristics are used to find it in general.

Subgradient method

The subgradient method is an iterative method that in each step solves the sub-problem (P2) and updates λ, ensuring that the new iteration belongs to a subgradient direction with an appropriate step. In general, we can describe the subgradient method as follows [57]:

- Select starting point $\lambda^0 \geq 0, n = 0$
- Build a sequence of points $\{\lambda^n\} \geq 0$ which eventually converges to the optimal solution using the following rule:
$$\lambda^{n+1} = \max(0, \lambda^n + s_n d^n)$$
- $n = n + 1$ with s_n a positive scalar is called step length and d^n is a vector called the direction of step, which must be determined in each iteration. In this case $d^n := b - Ax^{(n)}$.
- Until some stop condition is reached.

Some stop criteria are $|\lambda^{(k)}| < \varepsilon$ or $|b - Ax^{(k)}| < \varepsilon$ (complementary slack) $Z_L(\lambda^{(k)} - Z^*) < \varepsilon$, $|\lambda^{(k)} - \lambda^{(k-1)}| < \varepsilon$ where ε is a small number defined in advance or some mix between them.

The simplest versions of the subgradient start with $\lambda^0 = 0$. Other versions suggest starting with the dual variables corresponding to the relaxed constraint.

In Held et al. (1974) [67] they show that the sufficient condition for the step to give us convergence is as follows1: As $k \to \infty$, if

$$s_k \to 0 \text{ y } \sum_{j=1}^{k} s_j \to \infty$$

One formula often used to select the s_k is

$$s_k^2 = \frac{\beta_k (Z_L(\lambda^{(k)}) - Z^*)}{\|b - Ax^{(k)}\|_2^2} \tag{1.7}$$

where $\beta_k \in (0, 2]$ and Z^* is the best feasible solution found for the original problem. Usually β_k is initialized to 2 and halved if $Z_L(\lambda_k)$ does not change in some iterations.

The Kiev-method is a variant of the subgradient method that takes into account the information from the last two iterations to actualize the Lagrange multipliers1:

The Kiev-method is described below:

where $0 < \beta < 1$ and $\delta^t = b - x^{(t)}$. Let s_t be from the t iteration step,

$$s_{t-1} = \gamma_{t-1} \frac{z^*_{LB} - z_{UB}(\lambda^{t-1})}{\|G^{t-1}\|}$$

with $0\gamma_{t-1} \leq 2$. If s_t not changes into $\bar{t}$ iterations then $\gamma_t = \alpha\gamma_{t-1}$ with $0 < \alpha \leq 1$.

The optimal value of the sub-problem in each iteration can have an oscillating behavior. To avoid that, in [57] several modulations to the subgradient are proposed to diminish this effect. The main modulations are presented below.

To obtain a better convergence in [57], they suggest the following strategy. Let μ_k be the Lagrange multiplier in the iteration of the k subgradient method. Then, from p iterations of the subgradient method, they compare the best and the value obtained from z_{ub}. If the difference is greater than 1%, then μ is reduced by half. To the contrary, if the difference is less than 0.1%, then μ is increased by 50%. The above is summarised as follows1:

$$\mu_k = \begin{cases} 0.5\mu_{k-1}, & \text{if } \overline{UB} - \underline{UB} > 0.001 \cdot \underline{UB}, \\ 1.5\mu_{k-1}, & \text{if } \overline{UB} - \underline{UB} < 0.0001 \cdot \underline{UB}, \\ \mu_{k-1}, & \text{otherwise.} \end{cases}$$

where $UB = \{z^t_{UB} : t = k-p+1, k-p+2, \ldots, k\}$, $\overline{UB} = max(UB)$ and $\underline{UB} = min(UB)$. The authors suggest $p = 20$ and $\mu_0 = 0.1$.

Volume method

Barahona and Anbil [7] introduced the volume method. This method is based on Definition 1.4.1, which shows that (LDP) is the convexification of the entire problem (P1). Thus, the method seeks to find both α and λ.

Definition 1.4.1 (Near Optimal Solution) Suppose $\hat{\lambda} \in \Omega^* = \{\hat{\lambda} \in \mathbb{R}^m_+ : \phi(\hat{\lambda}), \forall \lambda \in \mathbb{R}^n_+\}$ and let $\mathbb{X} = \{\overline{x} \in conv(\mathbb{X}) : c\overline{x} + \hat{\lambda}(b - A\overline{x}) \leq cx + \hat{\lambda}(b - Ax), \forall x \; conv(\mathbb{X})\}$. Then a $\hat{x} \in \mathbb{X}(\hat{\lambda})$ is said to be the near optimum of the integer problem of (P1), if $A\hat{x} \geq 0$ and $\hat{\lambda}(A\hat{\lambda} - b) = 0$.

The volume method, as well as the sub-gradient method, is iterative. One of the main differences is that the volume method finds a primary solution close to the optimal.

Next, the volume method is described as it is presented in [57] :

Step 0 Set $\overline{\lambda} := \lambda^0$. Solve the sub-problem (P2). Let $\overline{x}$ be the solution of the sub-problem; and $\lambda^{k+1} = \max\{0, \overline{\lambda} + s_k d^k\}$, $k = 1$.

Step 1 Determine direction, $d^k = b - A\overline{x}^k$. Solve the sub-problem (P2). Let x^k and $\phi(\lambda^k)$ be the solutions obtained. Update $\overline{x} = \alpha x^k + (1-\alpha)\overline{x}$ to $\alpha \in (0, 1)$.

Step 2 If $\phi(\lambda^k) > \overline{\lambda}$, update $\overline{\lambda}$ and $\overline{\phi}$ as

$$\overline{\phi} := \phi(\lambda^k), \;\; \overline{\lambda} := \lambda^k$$

Then, $k = k + 1$.

The same step suggested in the subgradient method is used for the step:

$$s_k = \mu_k \frac{z^*_{LB} - \overline{\phi}}{\|d^k\|^2}$$

where $0 < k < 2$. To update μ_k in [7] follow three types of iterations.

- Every time $\overline{\phi}$ does not improve; it is counted as a *red* iteration. It is suggested that a smaller step is needed.
- If $\phi(\lambda^k) > \overline{\phi}$ and $d^k s^k < 0$ suggests that a longer step of d^k should give a smaller value of $\phi\lambda^k$. They call this type of iteration yellow.
- Finally, if $\phi(\lambda^k) > \overline{\phi}$ and $d^k s^k > 0$ will follow a larger step. It's known as a *green* iteration.

Thus μ^k is updated as follows.

$$\mu_k = \begin{cases} (0.66)\mu_{k-1}, & \text{after a sequence of 20 red iterations} \\ (1.1)\mu_{k-1}, & \text{if the k-th iteration is green} \\ \mu_{k-1}, & \text{otherwise} \end{cases}.$$

To determine the value of α in [7] they suggest two strategies; fix the value by several iterations and then decrease it under some criteria or solve the next problem of a dimension :

$$\begin{array}{ll} \min & \left\| b - A(\alpha x^k - (1-\alpha)\overline{x}) \right\| \\ s.t. & \frac{a}{10} \le \alpha \le a \end{array}$$

In [8] they start with a value of $\alpha = 0.1$, in case that the last 100 iterations the value of $\phi(\lambda^k)$ does not have an increment greater than 1% then it is reduced to half; otherwise, it is maintained. The value of a is set if it is less than 10^{-5}.

If $x^0, x^1, \ldots, x^k$ is a sequence of solutions of the sub-problem (P2), then we have to

$$\overline{x} = (1-\alpha)^k + (1-\alpha)^{k-1}\alpha x^1 + \cdots + (1-\alpha)\alpha x^{k-1} + \alpha x^k$$

for $\alpha \in (0.1)$. In addition the coefficients x^k, $k = 0.1, \ldots, k$ are also between 0 and 1:

$$\alpha \sum_{i=0}^{k-1} (1-\alpha)^i + (1-\alpha)^k = 1$$

which can be demonstrated by induction in k with $k = 1.2, \ldots$ [57]. Therefore, $\overline{x}$ is convex combination of $\{x^0, x^1, \ldots, x^k \subset \mathbb{X}$. Then from the theorem 2.3 it can be deduced that the solution found by the volume method estimates the solution of (LDP).

Like the subgradient method the stop criterion is whether $d^k \le 0$ and $\overline{\lambda}(b - A\overline{x}) = 0$ in this case $\overline{x}$ is the optimal solution of the problem (Q1) and $\overline{\lambda}$ is the optimal solution of (LDP). In general, this criterion is relaxed to $|b - A\overline{x}| < \varepsilon$ and $|c\overline{x} - b| < \varepsilon$ for a small ε value.

1.4.2 Cut generation

Benders decomposition

Consider the following formulation with n integer variables and p continuous variables to describe Benders Decomposition's classic version.

$$\min c^T x + h^T y \tag{1.8}$$
$$s.t.\ Ax = b \tag{1.9}$$
$$Bx + Dy = b \tag{1.10}$$
$$y \geq 0 \tag{1.11}$$
$$x \geq 0 \quad \text{and integer} \tag{1.12}$$

with complicating variables $x \in \mathbb{R}^{n_1}$, such to the constraint (1.9) where $A \in \mathbb{R}^{m_1 \times m_2}$ is a known matrix and $b \in \mathbb{R}^{m_1}$. The set of constraint (1.10) connects the variables x and y where $B \in \mathbb{R}^{m_2 \times n_1}$, $D \in \mathbb{R}^{m_2 \times n_2}$ and $d \in \mathbb{R}^{m_2}$. The objective function minimizes the total cost with the cost vectors $c \in \mathbb{R}^{n_1}$ and $h \in \mathbb{R}^{n_2}$. Fixing the values of the variables $\overline{x}$, we obtain a reformulated Linear Program (LP):

$$\min_{\overline{x} \in \mathbb{X}} \left\{ c^T \overline{x} + \min_{y \geq 0} \{ h^T y : Dy = d - B\overline{x} \} \right\} \tag{1.13}$$

where $\overline{x}$ is belongs to the set $\mathbb{X} = \{x | Ax = b, x \geq 0 \text{ and integer}\}$. The inner problem called subproblem (SP) is a continuous linear problem that can be reformulated consideреing the dual variable π associated with the constraint set $Dy = d - B\overline{x}$:

$$v(SP(\overline{x})) = \max_{\pi \in \mathbb{R}^{m_2}} \left\{ \pi^T (d - B\overline{x}) : \pi^T D \leq h \right\} \tag{1.14}$$

The primal and dual problem's function optimal objective values are equivalent by strong duality theorem. Then it is possible to replace the dual problem in (1.13). Then, the problem is reformulated as follows:

$$\min_{\overline{x} \in \mathbb{X}} \left\{ c^T \overline{x} + \max_{\pi \in \mathbb{R}^{m_2}} \{ \pi^T (d - B\overline{x}) : \pi^T D \leq h \} \right\}$$

Note the feasible space $F = \{\pi | \pi^T D \leq h\}$ of the dual problem does not depend of variable $\overline{x}$.It can use the extreme and rays extreme to define the polyhedra that describe the feasible space F. Hence, if F is not empty, then the inner problem can be either unbounded or feasible for any arbitrary choice of $\overline{x}$. For the first case, given an extreme ray set Q of F, there is a direction r_q, $q \in Q$ for which $r_q^T (D - B\overline{x}) > 0$. Thus, to avoid unbounded solutions, the following cuts can be added to restrict the movement in this direction:

$$r_q^T (D - B\overline{x}) \leq 0 \ \ \forall q \in Q \tag{1.15}$$

The solution to F is a point π_e, $e \in E$ where E is a set of extreme points of F. Therefore, if all cuts (1.15) are added to the outer problem and considering that the value of the inner problem will be one of its extreme points, then the model (1.13) can be reformulated as1:

$$\min_{\overline{x} \in \mathbb{X}} c^T x + \max_{e \in E} \{ \pi_e^T (d - B\overline{x}) \} \tag{1.16}$$
$$s.t.\ r_q^T (d - Bx) \leq 0 \ \ \forall q \in Q \tag{1.17}$$

Note that it is possible to reformulate the above model in a linear problem obtaining a MILP with many constraints known as Benders Master Problem (MP).

$$(MP) \quad \min c^T x + \nu \tag{1.18}$$
$$s.t. \; Ax = b \tag{1.19}$$
$$\nu \geq \pi_e^T (d - Bx) \;\; \forall e \in E \tag{1.20}$$
$$0 \geq r_q^T (d - Bx) \;\; \forall q \in Q \tag{1.21}$$
$$x \geq 0 \;\text{ and integer} \tag{1.22}$$

Due to the large number of constraints generated by points and rays extreme of the set F, a cut generation approach can be used to solve it efficiently. The method starts with a reduced number of constraints, progressively adding the necessary cuts. It is possible to check the current gap during the process to prove the convergence of the method. Indeed, in each iteration, a relaxation to MP is solved, then the objective function is a valid lower bound of the MP. On the other hand, if the solution $\overline{x}$ is feasible in the subproblem, then it is a feasible point in the MP. Thus, the sum of both $c^T\overline{x}$ and the objective solution associated with the subproblem provides an upper bound to the MP. Following with the method, the relaxed master problem (RMP) with a subset of constraint (1.20) and (1.21) are defined. Solving RMP yields an optimal solution (x^*, ν^*). The model (1.14) is solved as a separate problem to determine the most violated constraint by the current optimal solution is used.

$$(RMP) \quad \min c^T x + \nu \tag{1.23}$$
$$s.t. \; Ax = b \tag{1.24}$$
$$\nu \geq \pi_e^T (d - Bx) \;\; \forall e \in I_1 \subseteq E \tag{1.25}$$
$$0 \geq r_q^T (d - Bx) \;\; \forall q \in I_2 \subseteq Q \tag{1.26}$$
$$x \geq 0 \;\text{ and integer} \tag{1.27}$$

Thus, solving SP provides $v(SP(\overline{x}))$ the value of optimal solution and $\overline{\lambda} = \overline{\pi}$, the optimal solution that attains that value. One of the three following cases may occur:

1. If $v(SP(\overline{x})) \leq \nu^*$ then x^* is an optimal solution to MP.
2. If $v(SP(\overline{x})) > \nu^*$ then adding the constraint $\nu \geq \overline{\pi}^T(d - Bx)$ to RMP and reoptimize.
3. if $v(SP(\overline{x}))$ is unbounded then there exists an extreme ray $\overline{r}$ such that $\overline{r}(d - Bx^*) > 0$. Add $\overline{r}(d - Bx) \leq 0$ to RMP and reoptimize.

Unfortunately, in many cases, the cut generation presents slow convergence, mainly because the cuts added are not tight enough, which increases the progressive number of constraints increasing the computer time to solve each iteration. For this reason, many works aim to find a smart way to select better cuts in each iteration for reducing computational time. Other aspects that influence the performance of the method are ineffective initial iterations, zigzagging behavior of the primal solutions and slow convergence at the end of the algorithm, and upper bounds that remain unchanged in successive iterations because equivalent solutions exist [101].

Particularly, the branch-and-cut embedded bender decomposition method sometimes generates cuts that are not as tight, not improving the gap between the upper bound and the dual lower bound. In particular, the traditional cut generation scheme proposed by [82] performs poorly because the dual lower bound computed at each iteration can be erratic, resulting in slow convergence.

To improve this performance, stabilization procedures have been proposed at the root node, whose purpose is to reduce the number of times the splitting problems are solved. It has been studied that the convergence of the global strategy is highly dependent on the world chosen to be separated at each iteration [48]. Therefore, one way to improve the performance is by implementing a simple stabilization process as proposed by [11].

In this case, for the root node, we assume a fractional solution a_{out} obtain of the master problem, and a feasible solution a_{in} for the original MILP. Then, instead of trying to separate a_{out}, one can try to separate an intermediate point between the a_{out} and the feasible point a_{in} obtained by a convex combination of the two, i.e:

$$a_{sep} = \lambda a_{out} + (1 - \lambda) a_{in}$$

where $\lambda \in]0; 1]$. Note that when $\lambda = 1$, it is equivalent to performing no stabilization. While the smaller the value of *lambda*, the more aggressive the stabilization performed, since point one introduced in the separation problems is closer to a feasible solution. Algorithm 1.4.2 shows the pseudocode of this stabilization procedure.

Algorithm 1 Cut loop stabilization procedure (4.8)

```
 1: **Input** Current (fractional) solution $a_{out}$, Feasible solution $a_{in}$
 2: Set value of $\lambda \in ]0, 1]$
 3: Construct $a_{sep} = \lambda a_{out} + (1 - \lambda) a_{in}$
 4: Solve the separation problem
 5: **if** *separation problem is unbounded* **then**
 6:     Generate feasibility cut
 7: **else**Generate optimality cut
 8: **end if**
 9: **if** $\exists$ *generated cut which is violated by* $a_{sep}$ **then**
10:     add cut to master problem
11: **end if**
12: **if** *No cuts have been added* **then**
13:     Update 'interior' point: $a_{in} = a_{sep}$
14: **end if**
```

Similarly, the absence of stabilization can lead to slow convergence, as is the case with Kelley's scheme; too aggressive stabilization can cause one to spend too much time solving separation subproblems that do not generate violated cuts. Therefore, it is important to properly tune the value of *lambda* for good performance of the stabilization procedure.

Chapter 2

Lagrangian relaxation for product line optimization with multiples sites

To differentiate from others, retailers need to adjust their decisions according to the customers' demands and needs. Nowadays, seeing the amount of data collected, it is possible to meet this challenge in different areas. The question that arises here, therefore, is how to use this information. Particularly, companies have to decide where to allocate each product - which is known as the *Product Line Design Optimization* (PDLO)- assuming that there are varied preferences in each local area, such as taste, age, social, economic, or others. At the same time, people may go to closest-from-home stores, searching for a lower price or purchase the product they need, assuming the cost of travel. If the companies could use this information, they would reduce possible inventory costs due to slow product turnover, which may imply lower prices to increase turnover, this way, reducing its profits.

The aforementioned problem can be represented by an approach of bi-level programming, which can frame the effects of two agents that must make decisions, directly affecting each other and having conflict objectives. In this case, one agent makes the decision, first, and the second agent acts accordingly. Given this, the firm decides to allocate the products in its store; subsequently, each customer decides, assuming that they know where the product can be found.

In this chapter we study the use of a Lagrangian relaxation solution method for the PLOMS problem. We begin presenting the problem, its bi-level formulation and three different single-level formulations in Section 2.1. In Section 2.2, the proposed solution methods are described. Computational experiments that compare the different formulations and methods is explained in Section 2.3. Finally, we present our conclusions and possible lines of future work in Section 2.4.

2.1 Problem definition and formulations

This section presents a description of the *Product Line Optimization in Multiple Stores* (PLOMS) problem considered in this work. We provide three different formulations for the

PLOMS problem and introduce valid inequalities that are used later.

2.1.1 Problem description

A company or brand owns a set $\mathcal{J}$ of stores geographically distributed throughout a region. A set $\mathcal{K}$ indicates all products in a category, e.g., TV sets of specific screen size. All the products in this set are imperfect substitutes of another one and differ in secondary features and price. Let $\pi_{j,k}$ be the unit price of product $k \in \mathcal{K}$ at store $j \in \mathcal{J}$. There are capacity constraints, indicating that each store $j \in \mathcal{J}$ can display (and offer) up to p_j products of the set $\mathcal{K}$.

Moreover, there is a set $\mathcal{I}$ of consumers or potential clients. The travel cost from consumer's location i and the target store j is d_{ij}. Each purchaser $i \in \mathcal{I}$ is interested only in those products in a set $\mathcal{K}_i \subset \mathcal{K}$ and has a reservation price r_{ik} per product k in $\mathcal{K}_i$. The customers will buy at most one unit of product at one store, provided that the full cost (price plus travel cost) of the purchase does not exceed the reservation price for that product. Each consumer chooses the product and store that maximises their utility (i.e., the surplus obtained), subtracting the product's full cost from the reservation price. The firm must then decide what products in $\mathcal{K}$ to display at each store to maximise its revenue.

Tables 2.1 and 2.2 present the set and parameter notation used in this work.

Table 2.1: Set notation

$\mathcal{I}$	set of clients
$\mathcal{J}$	set of stores
$\mathcal{K}$	set of products
$\mathcal{K}_i$	set of products client $i \in \mathcal{I}$ is interested in

Table 2.2: Model parameters

d_{ij}	round-trip distance between client $i \in \mathcal{I}$ and store $j \in \mathcal{J}$
p_j	maximal number of products assigned to store $j \in \mathcal{J}$
$\pi_{j,k}$	unit price for product $k \in \mathcal{K}$ in store $j \in \mathcal{J}$
r_{ik}	reservation price of client $i \in \mathcal{I}$ for product $k \in \mathcal{K}$

2.1.2 Bilevel model formulation

This problem can be formulated as a linear bilevel program, where the company acting as an upper-level decision-maker decides the allocation of products to each store. At the lower level, the customers decide which products to purchase. For this purpose, let us first introduce two sets of binary decision variables. For $i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}_i$, x_{ijk} is a binary variable equal to 1 if client i selects product k at mall j. For $j \in \mathcal{J}, k \in \mathcal{K}$, y_{jk} is a binary variable equal to 1 if product k is present at mall j, 0 otherwise.

Then, we have the following bilevel optimization formulation with multiple independent followers:

$$\begin{aligned}
\max \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}_i} \pi_{jk}x_{ijk} && && \text{(2.1a)}\\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}} y_{jk} \le p_j && (j\in\mathcal{J}) && \text{(2.1b)}\\
& y_{jk}\in\{0,1\} && (j\in\mathcal{J}, k\in\mathcal{K}) && \text{(2.1c)}\\
& \text{for each } i\in\mathcal{I} \text{ we have} \\
& \quad \max \ \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}_i}(r_{ik}-\pi_{jk}-d_{ij})x_{ijk} && && \text{(2.1d)}\\
& \quad \text{s.t.} \ \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}_i} x_{ijk}\le 1 && && \text{(2.1e)}\\
& \qquad x_{ijk}\le y_{jk} && (j\in\mathcal{J}, k\in\mathcal{K}_i) && \text{(2.1f)}\\
& \qquad x_{ijk}\in\{0,1\} && (i\in\mathcal{I}, j\in\mathcal{J}, k\in\mathcal{K}_i)\ . && \text{(2.1g)}
\end{aligned}$$

Here the upper level objective function (2.1a) maximizes the profit of the company subject to constraint (2.1b) which limits the units to display in each store $j \in \mathcal{J}$ and the purchase decision of each customer obtained from the lower level problem. For each $i \in \mathcal{I}$, the lower level maximizes the customers utility (2.1d) (i.e., the reservation price minus the price and travel cost), subject to constraint (2.1e) that limit purchase to at most one product and constraint (2.1f) that forces customer i to purchase product j in store k only if is available. Meanwhile, the constraints (2.1c) and (2.1g) declare the binary nature of the variables $\mathbf{y}$ and $\mathbf{x}$, respectively.

Note that customer $i \in \mathcal{I}$ would not purchase a product for which the reservation price r_{ik} is smaller than the price π_{jk} plus the travel cost d_{ij}. It is interesting to remark that the optimal solution would never have the same product in one store with different markdowns. If one product had two prices in the same store, buyers would purchase only the least costly option since it would give them the most utility. We can define the set of products/locations that are attractive to customer i as: $\mathcal{T}_i = \{(j,k) | r_{ik} - \pi_{jk} - d_{ij} \ge 0\}$. With this notation we can set $x_{ijk} = 0$ if $(j,k) \notin \mathcal{T}_i$. We can express (2.1a), (2.1d) and (2.1e) as:

$$\begin{aligned}
&\sum_{i\in\mathcal{I}}\sum_{(j,k)\in\mathcal{T}_i} \pi_{jk}x_{ijk}\\
&\sum_{(j,k)\in\mathcal{T}_i} (r_{ik}-\pi_{jk}-d_{ij})x_{ijk}\\
&\sum_{(j,k)\in\mathcal{T}_i} x_{ijk} \le 1\ .
\end{aligned}$$

Note that constraints (2.1f) are in the second level problem. If they were considered in the first level problem, then the follower problem's solution is independent of the leader decisions. In this case, the first level problem is either feasible, with a value determined by the second level problem, or there is no solution to the problem.

2.1.3 Single-level model formulation

Since customers purchase at most one product, we assume they will purchase the product with the best utility from the subset $\mathcal{A} = \{(j,k)|y_{j,k} = 1\}$ of products available by the company in each store, which means they will select the pair $(j^*, k^*)_i = \arg\max\{r_{ik} - \pi_{jk} - d_{ij}|(j,k) \in \mathcal{T}_i \cap \mathcal{A}\}$. Considering this, we can establish an optimality constraint that identifies the purchase decision of each customer and allows us to transform the bi-level model into a single-level model

We now replace the second level optimization problem by setting constraints on the first level and second level variables that force the second level variables to be optimal. For all $i \in \mathcal{I}$, we rank the pairs $(j,k) \in \mathcal{T}_i$ according to the following order relation:

$$(j,k) \prec (j',k') \text{ if and only if } r_{ik} - \alpha_{jk}\pi_{j,k} - d_{ij} \leq r_{ik'} - \alpha_{j'k'}\pi_{j',k'} - d_{ij'}$$

This means that (j',k') is greater than (j,k) if it provides a larger reward for customer $i \in \mathcal{I}$. Then, we define the set of preferred products to $(j,k) \in \mathcal{T}_i$ for client $i \in \mathcal{I}$ by $\mathcal{B}_{ijk} = \{(j',k') \in \mathcal{T}_i|(j,k) \preceq (j',k')\}$. Note that $(j,k) \in \mathcal{B}_{ijk}$. Given a vector of integer variables y_{jk} for $j \in \mathcal{J}, k \in \mathcal{K}$, a vector x feasible for the second level problem (2.1d)-(2.1g) is optimal for this problem if and only if it satisfies the following set of constraints:

$$\sum_{(j',k') \in \mathcal{B}_{ijk}} x_{ij'k'} \geq y_{jk} \qquad i \in \mathcal{I}, (j,k) \in \mathcal{T}_i \qquad \text{(c-opt)}$$

Alternatively, we define the set of products that are not preferable to $(j,k) \in \mathcal{T}_i$ for client $i \in \mathcal{I}$, that is $\mathcal{W}_{ijk} = \{(j',k') \in \mathcal{T}_i|(j,k) \succ (j',k')\}$, so that $\mathcal{W}_{ijk} \cap \mathcal{B}_{ijk} = \emptyset$ and $\mathcal{T}_i = \mathcal{W}_{ijk} \cup \mathcal{B}_{ijk}$. The set of not preferable products, including (j,k), is denoted by $\mathcal{W}^*_{ijk} = \mathcal{W}_{ijk} \cup \{(j,k)\}$. Then,

$$\begin{aligned} \sum_{(j',k') \in \mathcal{B}_{ijk}} x_{ij'k'} + \sum_{(j',k') \in \mathcal{W}_{ijk}} x_{ij'k'} &\geq y_{jk} + \textstyle\sum_{(j',k') \in \mathcal{W}_{ijk}} x_{ij'k'} \quad i \in \mathcal{I}, (j,k) \in \mathcal{T}_i \\ \sum_{(j',k') \in \mathcal{T}_i} x_{ij'k'} &\geq y_{jk} + \textstyle\sum_{(j',k') \in \mathcal{W}_{ijk}} x_{ij'k'} \quad i \in \mathcal{I}, (j,k) \in \mathcal{T}_i \qquad (2.2) \end{aligned}$$

By constraints (2.1e), the left-hand side of (2.2) is smaller than or equal to 1, and the previous set of constraints can be rewritten as

$$y_{jk} + \sum_{(j',k') \in \mathcal{W}_{ijk}} x_{ij'k'} \leq 1 \qquad (i \in \mathcal{I}, (j,k) \in \mathcal{T}_i) \qquad \text{(c-opt-worse)}$$

Finally, constraint (c-opt-worse) can be strengthened as follows:

$$\sum_{(j',k') \in \mathcal{W}_{ijk}} x_{ij'k'} + \sum_{(j',k') \in \mathcal{W}_{ii'jk}} x_{i'j'k'} + y_{jk} \quad \leq 1 \qquad (i, i' \in \mathcal{I}, (j,k) \in \mathcal{T}_i) \quad \text{(c-opt-worse2)}$$

where $\mathcal{W}_{ii'jk} = W_{i'jk} \cap \mathcal{B}_{ijk}$. Note that this inequality is satisfied when $\sum_{(j',k') \in \mathcal{W}_{ijk}} x_{ij'k'} = 1$, because, in that case, for (j',k') and $(j,k) \in \mathcal{T}_i$, it holds that $y_{j'k'} = 0 \; \forall (j',k') \succeq (j,k)$. Then, $\sum_{(j',k') \in \mathcal{W}_{i'jk} \cap \mathcal{B}_{ijk}} x_{i'j'k'} = 0$.

Although, logically, the previous three sets of constraints are equivalent, we later show that these formulations lead to differences in solution times. This gives the following equivalent single-level integer programming formulation:

$$
\begin{aligned}
(M1)\ \ \max\quad & \textstyle\sum_{i\in\mathcal{I}}\sum_{(j,k)\in\mathcal{T}_i}\pi_{j,k}x_{ijk} && && (2.3a)\\
\text{s.t.}\quad & \textstyle\sum_{k\in\mathcal{K}} y_{jk}\le p_j && (j\in\mathcal{J}), && (2.3b)\\
& \textstyle\sum_{(j',k')\in\mathcal{B}_{ijk}} x_{ij'k'}\ge y_{jk} && (i\in\mathcal{I},(j,k)\in\mathcal{T}_i) && (2.3c)\\
& x_{ijk}\le y_{jk} && (i\in\mathcal{I},(j,k)\in\mathcal{T}_i), && (2.3d)\\
& \textstyle\sum_{(j,k)\in\mathcal{T}_i} x_{ijk}\le 1 && (i\in\mathcal{I}), && (2.3e)\\
& x_{ijk}\in\{0,1\} && (i\in\mathcal{I},(j,k)\in\mathcal{T}_i), && (2.3f)\\
& y_{jk}\in\{0,1\} && (j\in\mathcal{J},k\in\mathcal{K}). && (2.3g)
\end{aligned}
$$

If constraint (2.3c) is replaced by constraint (c-opt-worse), the model becomes (M2), and if the same constraint is replaced by constraint (c-opt-worse2), the model is (M3).

2.2 Lagrangian relaxation approach

Lagrangian relaxation (LR) has shown to be an efficient method when solving location problems [55], [44]. The Lagrangian relaxation method relaxes a set of hard constraints, the violation of which is penalised in the objective function by weighting them with the Lagrangian multipliers. To find the best Lagrangian multipliers (Dual Lagrangian problem), a popular approach is the subgradient method [63].

We propose and compare three variants of LR for model (M2). For these three variants, we use the Kiev subgradient method described in Algorithm 2), which considers the last two iterations to update the Lagrangian multiplier. We now describe this method, which is later used as the function `subgradient-method`() in the full Lagrangian relaxation method:

$$
\begin{aligned}
\Lambda^t =& \omega && \text{if } t=0\\
\Lambda^t =& \Lambda^{t-1} - s_0\frac{\delta^t}{||\delta^t||} && \text{if } t=1\\
\Lambda^t =& \Lambda^{t-1} - s_{t-1}\frac{\beta\delta^{t-1}+(1-\beta)\delta^{t-2}}{||\delta^t||} && \text{otherwise}
\end{aligned}
$$

where Λ^t is the Lagrangian multiplier vector in iteration t, ω represents the vector of dual variables of the relaxed constraints, δ^t is the gradient in iteration t, and $0<\beta<1$. Let s_t be the step of the iteration t, $s_{t-1}=\gamma_{t-1}\frac{z^*_{LB}-z_{UB}(\Lambda^{t-1})}{||g^{t-1}||}$ where z^*_{LB} be is the value of the best lower bound (feasible), and $z_{UB}(\Lambda^{t-1})$ is the value of the iteration's upper bound. With $0<\gamma_{t-1}\le 2$. If z^*_{LB} does not change in $\bar{t}$ iterations, then $\gamma_t=\alpha\gamma_{t-1}$ with $0<\alpha\le 1$.

In Algorithm 2, `no-improvement`() is true if z^*_{LB} has not improved in a preset number of recent iterations (we used 30). $\{x_{UB}, y_{UB}\}$ is the solution of `LR`() with the current vector Λ of multipliers, and Z_{UB} is its value. The function `feasible`() finds a feasible solution $\{x_{LB}, y_{LB}\}$, with an objective value Z_{LB}. Z^*_{LB} is the best feasible solution found. In the remainder of this Section, we describe three different ways of relaxing the PLOMS problem and explain the feasibility heuristic used and.

Algorithm 2 Lagrangian Relaxation method

```
    $\Lambda = \Lambda^0$                                          ▷ dual variables of the relaxed constraints
    $t = 0$, $\pi = 0.1$, stop = False
    while $t < t_{max}$ or stop = False do
        $\{Z_{UB}, (x_{UB}, y_{UB})\} = \texttt{LR}(\Lambda)$          ▷ returns the solution of the iteration
        $\delta = \texttt{gradient}(x_{UB}, y_{UB})$                    ▷ calculates the gradient
        $\{Z_{LB}, (x_{LB}, y_{LB})\} = \texttt{feasible}(x_{UB}, y_{UB})$    ▷ returns a feasible solution
        if $Z_{LB} > Z^*_{LB}$ then
            $Z^*_{LB}, \{x^*_{LB}, y^*_{LB}\} = Z_{LB}, \{x_{LB}, y_{LB}\}$
        end if
        if no-improvement() then
            $\pi = 0.9 \cdot \pi$
        end if
        if $\delta \cdot \Lambda < \varepsilon$ then
            stop = True
        else
            $\Lambda = \texttt{subgradient-method}(Z_{UB}, Z^*_{LB}, \delta)$     ▷ Kiev-subgradient method
        end if
        $t = t + 1$
    end while
```

2.2.1 Lagrangian Relaxation LR^1

In the first variant of Lagrangian relaxation we relax constraints (c-opt-worse) of formulation (M2) and add them to the objective with nonnegative Lagrangian multipliers λ_{ijk}. The formulation of the Lagrangian relaxation (LR^1) is

$$(\text{LR}^1)\ L^1(\lambda) = \max \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \pi_{jk} x_{ijk} \tag{2.4}$$

$$+ \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk} \left(1 - \sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} - y_{jk}\right)$$

$$= \max \left\{ \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \left[\left(\pi_{jk} - \sum_{(j'k')\in\hat{\mathcal{B}}_{ijk}} \lambda_{ij'k'} \right) x_{ijk} - \lambda_{ijk} y_{jk} \right] \right\} \tag{2.5}$$

$$+ \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk}$$

subject to constraints (2.3b), (2.3d)-(2.3g), where $\hat{\mathcal{B}}_{ijk} = \mathcal{B}_{ijk} \setminus \{(j,k)\}$ is the set of pairs that are strictly better than pair (j,k). This Lagrangian relaxation cannot be decomposed. The optimal solution to (LR^1) is found using Branch and Bound.

2.2.2 Lagrangian relaxation LR^2

In this variant, constraints (c-opt-worse) and (2.3d) of formulation (M2) are relaxed and weighted in the objective by nonnegative Lagrangian multipliers λ_{ijk} and γ_{ijkl}. The formulation of the Lagrangian relaxation (LR^2) is

$$\begin{aligned}
(\mathrm{LR}^2)\quad L^2(\lambda,\gamma) =& \max \sum_{i\in\mathcal{I}} \sum_{(j,k,l)\in\mathcal{J}_i} \pi_{jk} x_{ijk} + \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \gamma_{ijk}\left(y_{jk} - x_{ijk}\right) && (2.6)\\
&+ \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk} \left(1 - \sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} - y_{jk}\right)\\
=& \max\left\{ \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \left(\pi_{jk} - \sum_{j'k'\in\hat{\mathcal{B}}_{ijkl}} \lambda_{ij'k'} - \gamma_{ijk}\right) x_{ijk} \right\} && (2.7)\\
&+ \max\left\{ \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \left(\gamma_{ijk} - \lambda_{ijk}\right) y_{jk} \right\} + \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk}
\end{aligned}$$

subject to constraints (2.3b), (2.3e)-(2.3g). Recall that $\hat{\mathcal{B}}_{ijk} = \mathcal{B}_{ijk} \setminus \{(j,k)\}$.

This Lagrangian relaxation problem can be decomposed into separate subproblems, one to decide the product allocation variables (y_{jk}) and the other the customer choice variables (x_{ijk}). The formulation of the subproblem for variables (x) is written as:

$$(\mathrm{LR}^2)_{x_i} \quad L^2_{x_i}(\lambda,\gamma) = \max \sum_{(j,k)\in\mathcal{J}_i} f^x_{ijk} x_{ijk} \qquad (2.8)$$

subject to constraints (2.3e), (2.3f), where the index i is fixed for each subproblem and $f^x_{ijk} = \pi_{jk} - \sum_{j'k'\in\hat{\mathcal{B}}_{ijkl}} \lambda_{ij'k'} - \gamma_{ijk}$. The optimal solution to each subproblem $(L^2_{x_i})$ is obtained by making each customer $i \in \mathcal{I}$ choose the pair $(j,k) \in \mathcal{J}_i$ with the best value of f^x_{ijk}.

The formulation of the subproblem (y) is as follows:

$$(\mathrm{LR}^2)_{y_j} \; L^2_{y_j}(\lambda,\gamma) = \max \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} f^y_{jk} y_{jk}\ ,$$

subject to constraints (2.3b), (2.3g), where the index j is fixed for each subproblem and $f^y_{jk} = \gamma_{jk} - \lambda_{jk}$. The optimal solution to each subproblem $(\mathrm{LR}^2_{y_j})$ is found when the first p_j products with best $f^y_{jk} > 0$ values are selected for each store $j \in \mathcal{J}$. Both subproblems $(\mathrm{LR}^2)_{x_i}$ and $(\mathrm{LR}^2)_{y_j}$ have the integrality property; therefore, the optimal value of each (LR^2) is equal to the optimal value of its linear relaxation.

2.2.3 Lagrangian relaxation LR^3

The last Lagrangian relaxation variant is obtained, relaxing (c-opt-worse) and (2.3e) from formulation (M2), with nonnegative Lagrangian multipliers λ_{ijkl} and θ_i. The formulation of the Lagrangian relaxation (LR^3) is

$$\begin{aligned}
(\text{LR}^3)\ L_3(\lambda,\theta) =& \max \sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{T}_i} \pi_{jk} x_{ijk} + \sum_{i\in\mathcal{I}} \theta_i \left(1 - \sum_{(j,k)\in\mathcal{J}_i} x_{ijk}\right) \quad (2.9)\\
&+ \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk} \left(1 - \sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} - y_{jkl}\right)\\
=& \max\left\{\sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \left[\left(\pi_{jk} - \sum_{(j'k')\in\hat{\mathcal{B}}_{ijkl}} \lambda_{ij'k'} - \gamma_{ijk}\right) x_{ijk}\right]\right\} \quad (2.10)\\
&- \max\left\{\sum_{i\in\mathcal{I}} \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk} y_{jk}\right\} + \sum_{i\in\mathcal{I}} \left[\theta_i + \sum_{(j,k)\in\mathcal{J}_i} \lambda_{ijk}\right]
\end{aligned}$$

subject to constraints (2.3b), (2.3d), (2.3f), (2.3g), where $\hat{\mathcal{B}}_{ijkl} = \mathcal{B}_{ijk} \setminus (j,k)$. To find the optimal solution to the subproblem, for each pair (j,k) define

$$g^y_{jk} = \sum_{i\in\mathcal{T}_{jk}} \max\{\pi_{jk} - \sum_{(j'k')\in\hat{\mathcal{B}}_{ijk}} \lambda_{ij'k'} - \theta i, 0\} - \lambda_{ijk}$$

where $T_{jk} = \{i \in \mathcal{I} | (j,k) \in \mathcal{T}_i\}$ represents the set of clients that have product k in store j among their preferences. Then, for each $j \in \mathcal{J}$, choose the p_j products with best value $g^y_{jk} > 0$. Once this is done, variables x are assigned as follows:

$$x_{ijk} = \begin{cases} 1 & \text{if } y_{jk} = 1 \text{ and } \pi_{jk} - \sum_{(j'k')\in\hat{\mathcal{B}}_{ijk}} \lambda_{ij'k'} - \theta i > 0 \\ 0 & \text{otherwise .} \end{cases}$$

Note that the subproblem (LR^3) has the integrality property. Therefore, the optimal value of (LR^3) is equal to the optimal value of the linear relaxation of the original problem.

We highlight that the Lagrangian relaxations for model (M1) are similar to those shown above for (M2). In the previous experiments, we note that the results for (M1) are slightly worse than those for (M2). For the case of (M3), in earlier computational experiments, the multiplier update occupies a lot of time, harming the performance of the Lagrangian relaxation method, which is explained by a large number of constraints (c-opt-worse2).

2.2.4 Feasible solution

The subgradient algorithm requires a feasible solution on each iteration to update the Lagrangian multipliers. The heuristic describes below use the solution of the subproblem to find a feasible solution.

Note that for the three formulations mentioned above, the set of pairs store/product selected (y_{UB}) is feasible at each iteration since satisfying the capacity constraint. Therefore, we require only set up the purchasers' decision to obtain a feasible solution, which means they must decide what product and which store maximizes their utility. Hence, we define the set the set of pair store/product $P_a = \{(j,k) | y^{UB}_{jk} = 1\ (j,k) \in \mathcal{J} \times \mathcal{K}\}$ at the current iteration. The customer choice procedure is illustrated in Algorithm 3.

Algorithm 3 Feasibility heuristic

```
1: for $i \in I$ do
2:     $\mathcal{F}_i \leftarrow P_a \cap \mathcal{T}_i$
3:     for $(j,k) \in$ order-clients($\mathcal{F}_i$) do        ▷ orders the couples in $\mathcal{F}_i$ by decreasing utility
4:         $(x_{ijk}^{LB}, y_{jk}^{LB}) \leftarrow (1,1)$        ▷ picks the best
5:         break
6:     end for
7: end for
8: $Z_{LB} \leftarrow$ objective($x_{LB}, y_{LB}$)        ▷ returns the value of objective function
```

The feasible solution obtained by this heuristic is not a tight bound. Hence, two simple local changes to improve it are used. In the first local change, we checked if eliminating a duple $(j,k) \in P_a$ improves the feasible solution. The second local change consists of checking if adding a pair (i,j) to P_a improves the solution, provided that there is spare capacity at store j. With these two simple local changes, we obtained a reasonable, feasible solution.

The feasible solution obtained by this heuristic is not a tight bound. Hence, two simple local changes to improve it are used. In the first local change, we tested if removing a pair $(j,k) \in P_a$ improves the feasible solution. The second local change consists of checking if adding a pair (i,j) to the set P_a improves the solution, provided that there is spare capacity at store j. With these two simple local changes, we obtained a reasonable and feasible solution.

Furthermore, it is also possible to find a feasible solution by solving the original problem without constraints (2.3b) and only for the set P_a. In the previous experiment, we observed that the heuristic described above provides, in less time, a better solution than solving the original reduced model to optimality.

2.3 Numerical Experiments

This Section presents computational results obtained with the original problem's different formulations and the Lagrangian relaxation approach. The experiments were run on an Intel Xeon multicore machine with 16 gigabytes RAM running the CentOS release 6.7 and a Linux operating system, with 32 processors running at 2.00 gigahertz. Note that GUROBI was set to use a single processor, and multiple processes were run simultaneously on the machine.

2.3.1 Random instance

We first implemented a random instance generator, which we describe next. Let $\mathcal{I}$ be the number of clients, $\mathcal{J}$ be number of products, $\mathcal{K}$ be number of stores, $\underline{r}$ and $\overline{r}$ be the minimum and maximum reservation price respectively, and $\underline{\pi}$ and $\overline{\pi}$ be the minimum and maximum price respectively. For each product k and client i, we chose $r_{ik} \in \{\underline{r}, \overline{r}\}$ uniformly distributed. For each store j and product k, we chose $\pi_{jk} \in \{\underline{\pi}, \overline{\pi}\}$. The travel cost d_{ij} from client i to store j was also randomly distributed in $\{\underline{d}, \overline{d}\}$. The distribution is uniform.

2.3.2 Solving the Original problem

Recall that if a product is not attractive for any client then it is not considered. That is the set of possible products is $\mathcal{K} = \{k | \exists i \in \mathcal{I}, j \in \mathcal{J} \text{ s.t. } (j,k) \in \mathcal{T}_i\}$. Before solving the three formulations for the original problem ((M1), (M2) and (M3)), we perform the prepossessing that decreases the number of constraints.

For the models (M1) ((M2)), the following prepossessing was applied:

- If $|\mathcal{W}_{ijk}| = 0$ ($|\mathcal{B}_{ijk}| = \mathcal{T}_i$) then the pair (j,k) is the worst for client i. Then constraint (2.3c) ((c-opt-worse)) becomes $y_{jk} \leq 1$, and it can be removed.
- If $|\mathcal{W}_{ijk}| = |\mathcal{T}_i| - 1$ ($|\mathcal{B}_{ijk}| = 1$), then the pair (j,k) is the best option for client i. Then, constraint (2.3c) ((c-opt-worse)) becomes $y_{jk} \leq x_{ijk}$, but by (2.1f) $y_{jk} \geq x_{ijk}$, and it can be removed and replaced by the equivalent constraint $y_{jk} = x_{ijk}$.

For the model (M3), the following prepossessing was applied:

- If $|\mathcal{W}_{ii'jk}| = 0$ and $|\mathcal{W}_{ijk}| = 0$, then the pair (j,k) is the worst for client i. Then constraint (c-opt-worse2) becomes $y_{jk} \leq 1$, and it can be removed.
- If $|\mathcal{W}_{ii'jk}| = 0$ and $|\mathcal{W}_{ijk}| \neq 0$, then the constraint (c-opt-worse2) becomes equal to the constraint (c-opt-worse) and can be removed.
- If $|\mathcal{W}_{ii'jk}| \neq 0$ and $|\mathcal{W}_{ijk}| = 0$, then the constraint (c-opt-worse2) becomes equal to the constraint (c-opt-worse) and can be removed.
- If $|\mathcal{W}_{ijk}| = |\mathcal{T}_i| - 1$ and $|\mathcal{W}_{ii'jk}| = 0$, then the pair (j,k) is the best option for client i. Then, constraint (c-opt-worse2) becomes $y_{jk} \leq x_{ijk}$, but by (2.1f) $y_{jk} \geq x_{ijk}$, both are removed and replaced by the equivalent constraint $y_{jk} = x_{ijk}$.

Computational study

The MIPs were solved by GUROBI 7.5.1 in single-thread mode. The tests were run using a limit of 60 minutes of CPU time. The codes were implemented in Python. In this case, $\underline{\pi} = 400$ and $\overline{\pi} = 1000$; $\underline{r} = 500$ and $\overline{r} = 2000$; $\underline{d} = 10$ and $\overline{d} = 100$; $\underline{d} = 1$ and $\overline{d} = 0.2 \cdot |\mathcal{K}|$.

We solved instances with 200, 300 or 600 clients; 80 or 150 products; and 4 stores with at most 5 products displayed in each store. For each problem size, we solved for 10 randomly generated instances. We compare the solution quality and time for models $(M1)$, $(M2)$ and $(M3)$ in Table 2.3. We use the following notation: GAP is the average of gap of the each instance where $gap = 100\frac{UB-LB}{LB}\%$; $\underline{GAP}$ is the average of $\underline{gap}$ of each instance where $\underline{gap} = 100\frac{LB-\overline{LB}}{\overline{LB}}\%$; $\overline{GAP}$ is the average of $\overline{gap}$ of each instance where $\overline{gap} = 100\frac{UB-\overline{LB}}{\overline{LB}}\%$. For a given instance, UB is the best upper-bound value found, LB is the best lower-bound value found and $\overline{LB}$ is the best lower-bound value found between all models. Finally, t_{LP} is the average spent time (sec.) to obtain the optimal solution of the linear relaxation of the model considered.

$(M3)$ has the best $\overline{GAP}$ in all cases that obtain the solution; that is, it found the best upper bound between all formulations. However, this model's number of optimality constraints is much larger than the corresponding number in models $(M1)$ and $(M2)$. Due to this large number of constraints, the time required to obtain the linear relaxation solution is greater

Table 2.3: Averages GAP, $\underline{GAP}$, $\overline{GAP}$ and t_{LP} (sec) for several test instances. $|\mathcal{K}| = 4$ and $p_j = 5\ \forall j \in \mathcal{J}$.

Instance		$(M1)$				$(M2)$				$(M3)$			
$\|\mathcal{I}\|$	$\|\mathcal{J}\|$	GAP	$\underline{GAP}$	$\overline{GAP}$	t_{LP}	GAP	$\underline{GAP}$	$\overline{GAP}$	t_{LP}	GAP	$\underline{GAP}$	$\overline{GAP}$	t_{LP}
200	80	3.08	-0.19	2.88	11.21	2.00	-0.05	1.95	8.16	6.94	-4.05	1.28	332.12
200	150	3.62	-0.13	3.49	34.66	3.48	-0.24	3.24	27.72	3.01	-0.52	2.46	1206.09
300	80	4.24	-0.22	4.01	27.33	3.12	-0.02	3.10	27.14	2.31	-0.09	2.22 ^{1}	1383.68
300	150	5.46	-0.07	5.38	89.93	5.95	0.00	5.95	87.07	–	–	–	3575.15
600	80	6.05	-0.19	5.85	149.02	5.24	0.00	5.24	96.02	–	–	–	3600.00
600	150	7.50	0.17	7.68	545.80	7.03	0.31	7.36	503.80	–	–	–	3600.00

than the experiment's time (3600 sec.).

Since in the results, model $(M2)$ obtains a better GAP and $\overline{GAP}$ in almost all cases and is faster linear relaxation than $(M1)$, we use $(M2)$ from now on. In the following Section, we use Lagrangian relaxation on this model in three versions.

2.3.3 Lagrangian Relaxation approach

We run experiments using the same random test set described above. Table 2.4 shows the summary of the computational experiments. We compared the results of formulation $(M2)$ with different Lagrangian approaches. Both LR^2 and LR^3 did not obtain good results due to the integrality property. On the other hand, in the large instance, LR^1 obtained better $\overline{GAP}$ than $(M2)$, but worse $\underline{GAP}$ than $(M2)$; that is, the upper bound with LR^1 is better than $(M2)$, but the lower bound is worse than $(M2)$.

Table 2.4: Average comparison of GAP of the different Lagrangian relaxation formulations with the MIP

Instance		$(M2)$			(LR^1)			(LR^2)			(LR^3)		
$\|\mathcal{I}\|$	$\|\mathcal{J}\|$	GAP	$\underline{GAP}$	$\overline{GAP}$	GAP	$\underline{GAP}$	$\overline{GAP}$	GAP	$\underline{GAP}$	$\overline{GAP}$	GAP	$\underline{GAP}$	$\overline{GAP}$
200	80	2.00	-0.05	2.88	5.49	-1.25	4.17	11.83	-6.46	4.59	11.76	-6.41	4.59
200	150	3.48	-0.24	3.49	4.72	-1.33	3.32	15.00	-9.24	4.35	14.78	-9.08	4.35
300	80	3.12	-0.02	4.01	5.62	-1.47	4.06	12.98	-7.05	4.99	13.40	-7.40	4.99
300	150	5.95	0.00	5.38	4.90	-1.13	3.71	16.21	-8.97	5.78	16.29	-9.02	5.78
600	80	5.24	0.00	5.85	5.72	-1.69	3.92	14.28	-7.09	6.17	14.43	-7.22	5.85
600	150	7.03	0.31	7.36	2.81	-0.98	2.11	16.81	-7.15	6.23	16.83	-7.32	6.23

2.4 Conclusions

In most cases, the problems of assortment planning do not consider the cost of travel for customers. Furthermore, most of the preliminary works considered that the brand covers only one store. The model explained in this chapter incorporates the travel cost as part of the purchaser's utility and a company that covers more than one store. The customers can decide to travel farther away to purchase at a lower price or perhaps obtain an unavailable product at their neighbouring store. Previously published models cannot embrace this behaviour, and, in our thoughts, they lead to assortments that attract more customers. A bi-level formulation is proposed, considering the interaction between the choice of the client and the brand. Due to its structure, we can obtain a single level formulation. In this case, three different single-level formulations are suggested.

The computer results exhibit that formulation (M3) is the tightest. However, it generates a large number of constraints that make it inefficient on large-scale instances. Because of this handicap in (M3), the model (M1) was chosen for large-scale development. A Lagrangian relaxation approach is presented to resolve it. The computer results show that it yields favourable results for large instances.

There are several directions to expand the research in this chapter. It was reasonable for simplicity reasons to consider that clients buy at least one unit of the products. However, it is more common that customers will buy different products. It is also interesting to consider the case in which customers buy a bundle of products. There are several possibilities: the purchases can pay for all of the products in the bundle in one mall or different malls and incorporate the travel cost in their decision. From an algorithmic point of view, other lines or research are related to methods that exploit the characteristics of the model (M3). Additionally, extra preprocessing could be attempted in order to reduce redundant constraints and the size and depth of the problem.

Chapter 3

Cut generation to product line optimisation with multiples site

3.1 Introduction[1]

In chapter 2, we introduced the multi-product location case. It describes the decision made by a firm to allocate products in a group of stores, which hold a limited capacity to exhibit them, along with the customers' choice of buying the best and most useful product, including its full reservation price, product and travel cost. Therein a bi-level formulation is presented, which is collapsed to a single level formulation with mixed-integer optimisation.

In this chapter, we focus on the solution method through a single-level formulation. We based this work on two main ideas. First, a ranking-based formulation is presented in [12] for the Product Line Design (PLD) problem where each client's utility is introduced in the formulation as a constraint. These previous approaches [12] and [94] are compared in [14], which also introduces a strong formulation for [12]. A Benders Decomposition (BD), together with an efficient algorithm, is used to solve the problem, where the master problem defines the available products, and a separated slave problem solves each client's purchase decision. The efficiency of the BD is evaluated on synthetic data, providing good results. Furthermore, note that the structure of the PLOMS is similar to that of the *Facility Location Problem with Clients' Preferences* (FLPCP). In the FLPCP, the supplier should decide what facility to open, considering that each client will choose the facility according to her preferences. Among the main differences between FLPCP and PLOMS, it is remarkable that in the first, (i) facilities have an opening cost; (ii) the clients can purchase at any facility and; (iii) the facilities have unlimited capacity. Valid inequalities for the FLPCP are developed in [30] and [118], for their use within a Branch & Cut solution procedure.

We adapt valid inequalities that have been used for the Facility Location problem with Preferences to solve our problem using Branch and Cut ($B\&C$) and Cut and Branch ($C\&B$)

[1]This work has been partially funded by grants ANID-PFCHA/Doctorado Nacional/2016 # 21161328; INRIA Associated Team BIPLOS; CONICYT PIA AFB180003; FONDECYT 1190064; Fonds de la Recherche Scientifique -FNRS under Grant PDR T0098.18, and the the supercomputing infrastructure of the NLHPC (ECM-02) of the Universidad de Chile. M.L.C. and H.X.

methods. We compare the $B\&C$ and $C\&B$ with existing Benders' decomposition methods that have been used only for the single-store case.

Section 3.2 presents the problem together with some valid inequalities. In Section 3.3, the proposed solution methods are described. Computer testing comparing the different formulations and methods is presented in Section 3.4. Finally, we present our conclusions and lines for future work in Section 3.5.

3.2 Valid inequalities

The collapsing of the two levels of the problem into one gives the following equivalent single level integer programming formulation:

$$
\begin{array}{llll}
(M1) \ \max & \sum_{i\in\mathcal{I}}\sum_{(j,k)\in\mathcal{T}_i}\pi_{j,k}x_{ijk} & & (3.1a)\\
\text{s.t.} & \sum_{k\in\mathcal{K}} y_{jk} \le p_j & (j\in\mathcal{J}), & (3.1b)\\
& \sum_{(j',k')\in\mathcal{B}_{ijk}} x_{ij'k'} \ge y_{jk} & (i\in\mathcal{I},(j,k)\in\mathcal{T}_i) & (3.1c)\\
& x_{ijk} \le y_{jk} & (i\in\mathcal{I},(j,k)\in\mathcal{T}_i), & (3.1d)\\
& \sum_{(j,k)\in\mathcal{T}_i} x_{ijk} \le 1 & (i\in\mathcal{I}), & (3.1e)\\
& x_{ijk}\in\{0,1\} & (i\in\mathcal{I},(j,k)\in\mathcal{T}_i), & (3.1f)\\
& y_{jk}\in\{0,1\} & (j\in\mathcal{J},k\in\mathcal{K}), & (3.1g)
\end{array}
$$

Remember that the set of constraints (3.1c) models the decision problem of the buyers, representing the fact that they purchase the best available combination of product/location, in terms of utility. In words, constraint (3.1c) requires that, given a buyer i and a combination product/location (j,k), if the product k is available at store j, i.e., $y_{k,l}=1$, then the buyer must purchase this product/location unless there is another combination with a better utility. Considering the set $\mathcal{W}_{ijk}$, which is the complementary set of $\mathcal{B}_{ijk}$, constraint (3.1c) can be replaced for the following constraint:

$$
y_{jk} + \sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} \le 1 \qquad i\in\mathcal{I},(j,k)\in\mathcal{T}_i \tag{3.2}
$$

where, in contrast with the previous one, states that if the pair product/location (j,k) is available, the buyers will choose no product/location with less utility than (j,k).

Here, we introduce additional valid inequalities for the PLOMS, which are constructed considering the interaction of more than than one client $i\in\mathcal{I}$, to help define tighter equivalent formulations for the problem.

We strengthen constraint (3.2) by considering the set of product/locations that are worse than (j,k) for a second client i' but not for i. That is the set $\mathcal{W}_{ii'jk} = W_{i'jk}\cap\mathcal{B}_{ijk}$. Using this set, similar to [30], we obtain the following set of stronger inequalities:

$$\sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} + \sum_{(j',k')\in\mathcal{W}_{ii'jk}} x_{ij'k'} + y_{jk} \;\; \leq \;\; 1 \qquad i,i' \in \mathcal{I}, (j,k) \in \mathcal{T}_i \; . \tag{3.3}$$

Note that this inequality is satisfied when $\sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} = 1$, because in that case, $y_{\tilde{j},\tilde{k}} = 0$ for any $(\tilde{j},\tilde{k}) \succeq (j,k)$. This means that $x_{i\tilde{j}\tilde{k}} = 0$ for any $i \in \mathcal{I}$, giving $\sum_{(j',k')\in\mathcal{W}_{i'jk}\cap\mathcal{B}_{ijk}} x_{ij'k'} = 0$.

Constraint (3.3) can be generalized to multiple clients, as shown in [30]. Given $i_1,\ldots,i_s \in \mathcal{I}$, and $(j,k) \in \mathcal{T}_{i_1}$, then the following inequalities are valid for PLOMS:

$$\sum_{(j',k')\in\mathcal{W}_{i_1jk}} x_{i_1j'k'} + \sum_{t=2}^{s} \sum_{(j',k')\in B_{i_1jk}\cap\left(\bigcap_{q=2}^{t}\mathcal{W}_{i_qjk}\right)} x_{i_tj'k'} + y_{jk} \leq 1 \; . \tag{3.4}$$

Below, we present valid inequalities that do not arise from strengthening constraint (3.1c). The next two sets of valid inequalities are stated in propositions that establish relationships between two customers' variables.

Considering the sets $\mathcal{B}_{ijk}$, $\mathcal{B}_{i'jk}$ of the best alternatives for (j,k) by customers i, i', respectively and $\mathcal{B}_{i'jk} \subseteq \mathcal{B}_{ijk}$, if the purchaser i acquires the pair (j,k) means there exists no pair $(j',k') \in \mathcal{B}_{ijk}$ available, hence the purchaser i' also acquires the pair (j,k) (because $\mathcal{B}_{i'jk} \subseteq \mathcal{B}_{ijk}$ then there exists no either a product available with best utility for her). The above is formalized in the following proposition.

Proposition 3.2.1 Let x,y be a feasible solution for M1. Then for $i,i' \in \mathcal{I}$, $(j,k) \in \mathcal{T}_i$, we have

$$x_{ijk} \leq x_{i'jk} \quad \text{if} \quad \mathcal{B}_{i'jk} \subseteq \mathcal{B}_{ijk} \tag{3.5}$$

Proof: Assume that $\mathcal{B}_{i'jk} \subseteq \mathcal{B}_{ijk}$. If $x_{ijk} = 1$ then, with (3.1e), we have that $x_{ij'k'} = 0$ for all $(j',k') \in \mathcal{B}_{ijk} \setminus \{(j,k)\}$. This implies that $y_{j',k'} = 0$ for all $(j',k') \in \mathcal{B}_{ijk} \setminus \{(j,k)\}$ by using (3.1c) and that $\mathcal{B}_{ij'k'} \subseteq \mathcal{B}_{ijk} \setminus \{(j,k)\}$ for any such (j',k'). From $x_{ijk} = 1$, (3.1d) and (3.1c) we get $1 = y_{jk} \leq \sum_{(j',k')\in\mathcal{B}_{i'jk}} x_{ij'k'}$. This is a contradiction, since $(j',k') \in \mathcal{B}_{ijk} \setminus \{(j,k)\}$ by the hypothesis and, as concluded above, $y_{j'k'} = 0$. Therefore $x_{i'jk} = 1$ completing the proof. $\square$

Proposition 3.2.1 generalizes a result in [30] to the case in which the sets of preferred products are different for each client. When $\mathcal{B}_{ijk} = \mathcal{B}_{i'jk}$, this result can be obtained as a corollary when repeating Proposition 3.2.1.

Corollary 1 Let x, y be a feasible solution for M1. Then for $i,i' \in \mathcal{I}$, $j \in \mathcal{J}$, $k \in \mathcal{K}$,

$$x_{ijk} = x_{i'jk} \quad \text{if} \quad \mathcal{B}_{ijk} = \mathcal{B}_{i'jk} \tag{3.6}$$

The following result requires the definition of the set $\mathcal{B}_{ii'jk} = \mathcal{B}_{ijk} \cap \mathcal{T}_{i'}$ of product/location pairs that are preferred to (j,k) for i that are profitable for i'. Note that this set $\mathcal{B}_{ii'jk}$ is empty if $\mathcal{T}_i \cap \mathcal{T}_{i'} = \emptyset$.

Proposition 3.2.2 Let $x,\ y$ be a feasible solution for M2. Then for $i, i' \in \mathcal{I}$, $(j,k) \in \mathcal{T}_i$, we have

$$\sum_{(j',k')\in\mathcal{W}_{ijk}} x_{ij'k'} + \sum_{(j',k')\in\mathcal{B}_{ii'jk}} x_{i'j'k'} \leq 1 \ . \tag{3.7}$$

Proof: Note that if $\mathcal{B}_{ii'jk} = \emptyset$, (3.7) is true, since it is implied by (3.1e). Let us assume, therefore, that $\mathcal{B}_{ii'jk} \neq \emptyset$. If the first sum is equal to one, then there exists $(j',k') \in \mathcal{W}_{ijk}$ such that $x_{ij'k'} = 1$. This means, considering (3.2) for $(\hat{j},\hat{k}) \succeq (j,k)$, that $y_{\hat{j}\hat{k}} = 0$ for all $(\hat{j},\hat{k}) \in \mathcal{B}_{ijk}$. In particular, $y_{\hat{j}\hat{k}} = 0$ for all $(\hat{j},\hat{k}) \in \mathcal{B}_{ii'jk}$. This, and (3.1d) imply that the second sum is zero. Consider now that the second sum is equal to one. Then, there exists $(j',k') \in \mathcal{B}_{ii'jk} \subseteq \mathcal{B}_{ijk}$ such that $x_{i'j'k'} = 1$. Because of (3.1d), $y_{j'k'} = 1$, and therefore $x_{i\hat{j}\hat{k}} = 0$ for all $(\hat{j},\hat{k}) \in \mathcal{W}_{ij'k'}$ due to equation (3.2). This implies that the first sum is zero, since $(j',k') \in \mathcal{B}_{ijk}$ means $\mathcal{W}_{ijk} \subseteq \mathcal{W}_{ij'k'}$. $\square$

Using the above valid inequalities, we construct two additional equivalent formulations for the PLOMS. We note that (3.5) has at most $\sum_{i\in\mathcal{I}} |\mathcal{T}_i|$ total constraints, while (3.7) could have up to $|\mathcal{I}| \sum_{i\in\mathcal{I}} |\mathcal{T}_i|$ total constraints.

We denote by M3 the problem that considers valid inequalities (3.5) and replaces (3.1c) in M1 with (3.3). This formulation include constraints that model interactions between pairs of consumers. We also define M4 as the problem that incorporates valid inequalities (3.5) and replaces (3.1c) in M1 with (3.4), modeling interactions between sets of customers. These formulations are summarized in Table 3.1. Since constraints (3.2), (3.3), and (3.4) are

Table 3.1: Summary of the different formulations

Model	(3.1a)	(3.1b)	(3.1c)	(3.1d)	(3.1e)	(3.2)	(3.3)	(3.4)	(3.5)
M1	X	X	X	X	X				
M2	X	X		X	X	X			
M3	X	X		X	X		X		X
M4	X	X		X	X			X	X

increasingly stronger constraints, the corresponding formulations are tighter formulations of the PLOMS. Our approach to solve large instances of M3 and M4 will consider subsets of constraints (3.3) and (3.4), respectively. In addition, we consider the effect of including inequalities (3.7) in these formulations. Since there is a large number of these constraints, we add them using cutting plane approaches, similar to [118], as we see below.

3.3 Solution methods

3.3. SOLUTION METHODS

The formulations M2, M3 and M4 of the PLOMS problem introduced in the previous table, represent mixed-integer problems that can be directly handled by a commercial solver. To efficiently solve large instances, we investigate different decomposition strategies for the

PLOMS. This Section presents a Benders decomposition strategy that is applicable to M2, problem reductions for M3 and M4, and a cut generation method for inequalities (3.7). We finalize with some variable and constraint simplifications for PLOMS.

3.3.1 Existing Benders decomposition method

[14] present a Benders decomposition approach, using a formulation which is similar to M2, but applies when there is only one store. The master problem in the method prescribes which products are to be placed in the store, letting the customer's purchase decision as the second stage problem, which is separable in $|\mathcal{I}|$ independent sub-problems.

An efficient algorithm to solve these sub-problems is presented. This solution method first assigns to each client the available product with the highest price, using for each customer a greedy algorithm to solve the dual problem, which generates the Bender's Cuts that are added as lazy constraints. To the best of our knowledge, this is the most efficient solution method available for the one store problem, and we use this solution method as a benchmark for the solution methods proposed here.

In order to adapt this Benders decomposition method to the multiple-store case, we let the master problem solve the product availability problem in all stores and pass the solution to the master problem; the sub-problems depend on each customer's purchase decision. In the case of the M2 formulation, these sub-problems remain separable in $|\mathcal{I}|$ sub-problems and can still be solved efficiently, using the algorithm suggested in [14]. In this adaptation of the Benders decomposition method, each sub-problem can generate a Benders' optimality cut in every iteration.

The fact that formulations M3 and M4 include constraints that involve multiple clients, breaks the sub-problems separability and generates sub-problems that cannot be solved with the method suggested in [14]. Adapting this solution method of solving these formulations in the multiple-store case is not straightforward, and become a matter of future research. We therefore, only consider this Benders decomposition method in formulation M2, and also refer to it as M2-BD.

3.3.2 Cut generation methods

To reduce the problem size, here we introduce relaxations to problems M3 and M4 by considering only a subset of constraints (3.3) and (3.4). We also present a cut generation approach that gradually incorporates constraints (3.7). Cut generation approaches can either be used to add cuts only at the root node of the branch and bound tree, known as a Cut and Branch ($C\&B$) approach or used to add cuts throughout the branch and bound tree as needed, a Branch and Cut ($B\&C$) approach.

The other potentially large set of constraints, (3.5), does not generate computational difficulties in our experiments, either because they are preceding constraints or because the condition $\mathcal{B}_{i'jk} \subseteq \mathcal{B}_{ijk}$ is difficult to satisfy.

A subset of constraints (3.3)

Note that for a particular constraint (3.3) to be different from the constraints in (3.2), it is necessary that $\mathcal{W}_{ii'jk} \neq \emptyset$. Given the definition of $\mathcal{W}_{ii'jk}$, this set can be large when the attractive products for two customers overlap significantly. Therefore, the subsets of constraints (3.3) that are selected, correspond to pairs of clients $i, i' \in \mathcal{I}$ with $i \neq i'$ that have a large set of common attractive product/locations, i.e. large $c_{ii'} = |\mathcal{T}_i \cap \mathcal{T}_{i'}|$.

To find a set of pairs of clients that have a large number of common attractive product/locations we consider the following optimization problem.

$$\begin{array}{lll} \max & \sum_{i\in\mathcal{I}} \sum_{\substack{i'\in\mathcal{I}\\ i\neq i'}} c_{i,i'} z_{i,i'} & \\ \text{s.t.} & \sum_{\substack{i'\in\mathcal{I}\\ i\neq i'}} (z_{i,i'} + z_{i',i}) = 1 & (i \in \mathcal{I}) \\ & \quad z_{i,i'} \in \{0,1\} & ((i,i') \in \mathcal{I}, i \neq i'). \end{array}$$

The optimal solution for this optimization problem indicates which pair of clients are to be used to build the subset of constraints (3.3). We include one such constraint for each pair (i, i') such that $z_{i,i'} = 1$. Note that the number of constraints generated are equal to the number of constraints in (3.2), since we generated the constrains for $(i, i'), (j, k) \in \mathcal{T}_i$ and $(i, i'), (j, k) \in \mathcal{T}_{i'}$.

A subset of constraints (3.4)

There is a set of constraints (3.4) for every possible group of clients $\{i_1, \dots, i_r\}$. To identify which groups of them to use to generate the subset of (3.4), we used the following procedure, introduced in [30].

1. Let $(j, k) \in \mathcal{J} \times \mathcal{K}$ and let $C = \{i_1 \in \mathcal{I} : \exists i_2 \in \mathcal{I} \mid \mathcal{W}_{i_1jk} \cap \mathcal{W}_{i_2jk} = \emptyset,\ \mathcal{W}_{i_1jk} \cap \mathcal{B}_{i_2jk} \neq \emptyset,\ \mathcal{W}_{i_2jk} \cap \mathcal{B}_{i_1jk} \neq \emptyset\}$.
2. We consider the graph obtained from associating a node to each element of C and an edge to each pair $(i_1, i_2) \in C \times C$ such that $\mathcal{W}_{i_1jk} \cap \mathcal{W}_{i_2jk} = \emptyset,\ \mathcal{W}_{i_1jk} \cap \mathcal{B}_{i_2jk} \neq \emptyset,\ \mathcal{W}_{i_2jk} \cap \mathcal{B}_{i_1jk} \neq \emptyset$.
3. We search for a clique $\{i_1, \dots, i_r\}$ in this graph and replace the inequalities $\sum_{(j',k')\in\mathcal{W}_{i_tjk}} x_{i_tj'k'} + y_{jk} \leq 1$ for $t = 1, \dots, r$, with the tighter inequality $\sum_{t=1}^{r} \sum_{(j'k')\in\mathcal{W}_{i_tjk}} x_{i_tj'k'} + y_{jk} \leq 1$.
4. Nodes $i_1, \dots, i_r$ are removed form the graph and the process is repeated with the remaining nodes until a graph with no edges is obtained.

This procedure modifies constraints (3.2) in step 3 by replacing them with constraints of the form (3.4). Since at every iteration we are introducing tighter constraints, the resulting subset of (3.4) implied constraints (3.2).

Cut generation

For the cut generation strategy, we solve the problem M4 adding a subset of constraints (3.7). Once a solution to this problem is obtained, we check whether any of the remaining constraints (3.7) is violated. For each client, we generate at most one of the violated constraints, including it in the formulation. Then, we re-optimize and repeat this procedure until the optimal solution satisfies all constraints (3.7). A similar cut generation strategy for constraints (3.4) was not competitive.

A critical part in constructing an effective cut generation strategy is to be able to quickly check if there are violated constraints. For the case of constraint (3.7) we begin by noticing that it is not necessary to check all inequalities indexed in $(j,k) \in \mathcal{T}_i$ for a given pair $i, i' \in \mathcal{I}$. For this, we define $\sigma_i(j,k)$ as the position of pair (j,k) in the set of preferences $\mathcal{T}_i$ (in increasing order with respect to the utility of client i). Define also, $(j,k)^i_{\min} = \arg\min_{(j,k)\in S}\{\sigma_i(j,k)\}$ and $(j,k)^{i'}_{\max} = \arg\max_{(j,k)\in S\cap\mathcal{T}_{i'}}\{\sigma_i(j,k)\}$, where $S = \{(j,k)|x_{ijk} > 0\ (j,k) \in \mathcal{T}_i\}$. We now show that for any $(j,k) \notin [(j,k)^i_{\min}, (j,k)^i_{\max}]$ variable x satisfies constraint (3.7).

By definition we have that $x_{ij'k'} = 0$ for all $(j',k') \in \mathcal{W}_{i(j,k)^i_{\min}}$, which implies that $\sum_{(j'k')\in\mathcal{W}_{i(j,k)^i_{\min}}} x_{ij'k'} = 0$ hence inequality (3.7) is satisfied. Likewise, $x_{i'jk} = 0$ for all $(j,k) \in \mathcal{B}_{ii'(j,k)^{i'}_{\max}}$ which implies that $\sum_{(j'k')\in\mathcal{B}_{ii'(j,k)^{i'}_{max}}} x_{ij'k'} = 0$, hence, inequality (3.7) is satisfied.

The process of generating cuts by only verifying the range $[(j,k)^i_{\min}, (j,k)^i_{\max}]$ for each $i, i' \in \mathcal{I}$ is described in Algorithm 7 below. Note that we add at most one valid inequality for each client in each iteration. These inequalities were added to the problem as lazy constraints.

Algorithm 4 Cut generation

```
1: $stop = True$
2: for $i \in \mathcal{I}$ and $stop = True$ do
3:     $S = \{(j,k)|x_{ijk} > 0\ (j,k) \in \mathcal{T}_i\}$
4:     $jk^i_{\min} = \arg\min_{(j,k)\in S}\{\sigma_i(j,k)\}$
5:     $jk^i_{\max} = \arg\max_{(j,k)\in S}\{\sigma_i(j,k)\}$
6:     if $y_{(j,k)^i_{\max}} < 1$ then
7:         for $i' \in \mathcal{I} \setminus \{i\}$ do
8:             $(j,k)^{i'}_{\max} = \arg\max_{(j,k)\in S\cap\mathcal{T}_{i'}}\{\sigma_i(j,k)\}$
9:             for $(\hat{j},\hat{k}) \in \mathcal{T}_i \cap [(j,k)^i_{\min}, (j,k)^{i'}_{\max}]$ do
10:                if $\sum_{(j',k')\in\mathcal{W}_{i\hat{j}\hat{k}}} x_{ij'k'} = y_{(j,k)^i_{max}}$ then
11:                    break
12:                end if
13:                if $\sum_{(j',k')\in\mathcal{W}_{i\hat{j}\hat{k}}} x_{ij'k'} + \sum_{(j',k')\in\mathcal{B}_{ii'\hat{j}\hat{k}}} x_{ij'k'} \leq 1$ then
14:                    add cut$(i, i', \hat{j}, \hat{k})$
15:                    $stop = False$
16:                    break
17:                end if
18:            end for
19:        end for
20:    end if
21: end for
```

3.3.3 Problem preprocessing

To speed up the solution for these models, we remove or simplify the constraints that are easy to check, reducing the problem size. In particular we conduct the following simplifications for model M2:

- If $|\mathcal{W}_{ijk}| = 0$ ($|\mathcal{B}_{ijk}| = \mathcal{T}_i$) then the pair (j, k) is the worst for client i. Constraint (3.2) becomes $y_{jk} \leq 1$ and can be removed.
- If $|\mathcal{W}_{ijk}| = |\mathcal{T}_i| - 1$ ($|\mathcal{B}_{ijk}| = 1$) then the pair (j, k) is the best option for client i. Constraint (3.2) becomes $y_{jk} \leq x_{ijk}$, but by (3.1d) $y_{jk} \geq x_{ijk}$. Both constraints are removed and replaced by $y_{jk} = x_{ijk}$.

Similarly, we conducted the following simplification for model M3:

- If $|\mathcal{W}_{ii'jk}| = 0$ and $|\mathcal{W}_{ijk}| = 0$, then the pair (j, k) is the worst for client i. Constraint (3.3) becomes $y_{jk} \leq 1$ and can be removed.
- If $|\mathcal{W}_{ii'jk}| = 0$ and $|\mathcal{W}_{ijk}| \neq 0$, then constraint (3.3) become equal to constraint (3.2) and could be removed.
- If $|\mathcal{W}_{ii'jk}| \neq 0$ and $|\mathcal{W}_{ijk}| = 0$, then constraint (3.3) becomes equal to constraint (3.2) and can be removed.
- If $|\mathcal{W}_{ijk}| = |\mathcal{T}_i| - 1$ and $|\mathcal{W}_{ii'jk}| = 0$, then the pair (j, k) is the best option for client i. Constraint (3.3) becomes $y_{jk} \leq x_{ijk}$, but by (3.1d) $y_{jk} \geq x_{ijk}$, so both are remove and replace by $y_{jk} = x_{ijk}$.

3.4 Computational experiments

We now present the computational tests. All the procedures and algorithms have been written using the language Python and, for MIP problems, we used IBM ILOG CPLEX version 12.9. The experiments were performed in an Intel Xeon 32 multicore processors, 2.00 GHz speed each and 128 GB of RAM, running the CentOS release 6.7 Linux operating system. We begin by describing the synthetic data that was used in our experiments. Our computational results explore the strength of the different formulations and the of existing Benders decomposition strategies (SubSection 3.4.2), the effectiveness of the constraints (3.7) in SubSection 3.4.3, and the comparison of the proposed decomposition methods and a benchmark method on the multiple-store case (SubSection 3.4.4).

3.4.1 Instances: description

The data sets were adapted from Beasley's OR-Library [11]. Values d_{ij}, r_k and r_i^k were built, based on data files *pmed10*, *pmed25* for the uncapacitated warehouse location problem. $|\mathcal{I}|$ nodes were selected randomly as clients. To select $|\mathcal{J}|$ nodes as stores, we solved an uncapacitated *p-median problem*. The travel cost between each client i and each store j are trivial to obtain. The remaining parameters were determined as follows: $\pi_k \in [1.5\overline{d}, 5\overline{d}]$ and $r_i^k \in [2\underline{d_i} + \overline{\pi}, 2\overline{d_i} + \overline{\pi}]$, where $\overline{d}$ is the average of all travel costs, $\overline{\pi}$ is the average of all products prices; $\underline{d_i}$ and $\overline{d_i}$ the travel cost between the client i and the closest and farthest store, respectively. Finally, we fixed a capacity p_j, which is the number of products to display, arbitrarily for each store. Without loss of generality, we fixed all stores with equal capacity, i.e $p_j = p \; \forall j \in \mathcal{J}$.

For the number of customers, stores, products, and capacities, we used the values $|\mathcal{I}| = 100, 160, 250, 400$, $|\mathcal{J}| = 4, 8, 12$, $|\mathcal{K}| = 20, 30, 40, 50, 80$ and $p_j = p = 5, 10 \; \forall j \in \mathcal{J}$. We using small values of capacities to be able to evaluate better the performance of the methods, as preliminary experiments showed that large capacities decrease the computational time.

Each instance so defined, was run using five different scenarios. Table 3.4.1 shows statistics of the generated instances, obtained over the five scenarios. The Table is divided in three blocks displaying the same statistics for 4, 8 and 12 stores. Each block is divided in seven columns. The first three show minimum, average and maximum number of clients (i) per pair (j, k), i.e. (product, store), while columns four, five and six show the converse. Column seven shows the average number of clients with the same set $\mathcal{T}_i$.

Table 3.2: Statistics of generated instances

Instance		4 stores							stores 8							stores 12						
		clients x ($j.k$)			items x clients			same clients	clients x items			items x clients			same clients	clients x items			items x clients			same clients
$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{K}\rvert$	min	avg	max	min	avg	max		min	avg	max	min	avg	max		min	avg	max	min	avg	max	
	20	12.2	20.8	28.4	12.2	16.6	28.0	0.6	12.0	20.5	29.2	12.0	32.8	84.0	0.2	11.2	20.5	28.8	11.2	32.8	56.0	0.2
	30	11.4	20.6	29.8	11.4	24.7	40.0	0.0	11.0	20.7	29.6	11.0	49.8	120.0	0.0	11.0	20.7	30.4	11.0	49.6	80.0	0.2
100	40	10.8	20.7	31.4	10.8	33.2	55.6	0.0	9.2	20.3	30.2	9.2	64.9	164.8	0.0	8.6	20.3	30.8	8.6	64.9	109.6	0.0
	50	12.2	21.5	31.6	12.2	43.0	72.0	0.0	8.4	20.8	33.8	8.4	83.3	209.8	0.0	9.8	20.8	32.6	9.8	83.3	143.6	0.0
	80	10.2	21.1	34.6	10.2	67.4	110.4	0.0	7.6	20.3	31.6	7.6	130.1	327.4	0.0	8.8	20.3	32.2	8.8	130.1	216.4	0.0
	20	22.8	34.1	45.4	22.8	17.1	28.0	1.6	20.2	33.6	45.4	20.2	33.6	84.0	1.0	19.6	33.6	45.8	19.6	33.6	56.0	1.4
	30	19.6	32.1	43.0	19.6	24.1	40.0	0.2	17.8	30.9	43.0	17.8	46.3	120.0	0.0	17.8	30.9	42.6	17.8	46.3	79.8	0.0
160	40	21.4	33.7	46.0	21.4	33.7	56.0	0.0	15.2	32.5	45.4	15.2	65.1	167.4	0.0	18.6	32.5	46.8	18.6	65.1	110.6	0.0
	50	22.2	34.3	46.4	22.2	42.9	72.0	0.0	16.6	33.5	45.8	16.6	83.7	213.0	0.0	19.2	33.5	46.8	19.2	83.7	143.6	0.0
	80	21.6	34.2	49.4	21.6	68.4	111.4	0.0	15.4	33.5	48.0	15.4	133.8	326.0	0.0	17.8	33.5	46.4	17.8	133.8	217.6	0.0
	20	30.8	52.3	67.2	30.8	16.7	28.0	3.4	33.4	51.3	66.6	33.4	32.8	84.0	3.6	31.4	51.3	67.4	31.4	32.8	56.0	2.6
	30	30.4	49.6	66.2	30.4	23.8	40.0	0.0	28.2	49.4	67.0	28.2	47.4	120.0	0.0	30.0	49.4	66.8	30.0	47.4	80.0	0.0
250	40	33.2	50.8	71.4	33.2	32.5	56.0	0.0	28.8	50.2	69.8	28.8	64.2	168.0	0.0	31.6	50.2	69.0	31.6	64.2	112.0	0.0
	50	32.6	52.5	73.2	32.6	42.0	72.0	0.0	27.0	52.6	71.4	27.0	84.2	214.2	0.0	28.0	52.6	71.2	28.0	84.2	143.0	0.0
	80	28.6	51.3	72.0	28.6	65.7	109.8	0.0	27.0	51.7	71.2	27.0	132.4	327.4	0.0	29.6	51.7	71.0	29.6	132.4	220.4	0.0
	20	56.6	81.0	103.4	56.6	16.2	28.0	10.0	51.2	79.6	103.8	51.2	31.9	84.0	8.0	53.0	79.6	103.4	53.0	31.9	56.0	10.8
	30	51.2	78.5	100.6	51.2	23.6	40.0	0.4	50.4	78.7	104.2	50.4	47.2	120.0	0.2	49.2	78.7	103.4	49.2	47.2	80.0	0.4
400	40	52.8	80.7	105.2	52.8	32.3	56.0	0.0	47.2	80.2	104.4	47.2	64.1	168.0	0.0	52.0	80.2	108.2	52.0	64.1	112.0	0.0
	50	54.2	83.7	108.0	54.2	41.8	71.8	0.0	52.8	83.6	107.6	52.8	83.6	216.0	0.0	52.4	83.6	106.6	52.4	83.6	143.8	0.0
	80	52.8	82.3	107.4	52.8	65.8	111.6	0.0	49.8	82.0	108.6	49.8	131.2	333.6	0.0	49.6	82.0	107.8	49.6	131.2	222.2	0.0

We performed several experiments to compare our formulations and methods with each other, and to compare these with the Benders Decomposition approach.

In the results below we denote each instance with its problem size, as either "$|\mathcal{I}|\_|\mathcal{J}|\_|\mathcal{K}|$" or "$|\mathcal{I}|\_|\mathcal{K}|$" depending on what is being compared with. For each problem we consider 5 different random instances, all results presented are the average over these 5 instances.

The headers of the tables presenting the computational results use the following nomenclature:

- $|\mathcal{I}|\_|\mathcal{J}|\_|\mathcal{K}|$ or $|\mathcal{I}|\_|\mathcal{K}|$: Name of instance.
- GAP: average gap $100\frac{UB-LB}{LB}\%$, where UB(LB) is the best upper(lower) bound.
- $RGAP$: average integrality gap $100\frac{LP-LB}{LB}\%$, with LP = linear relaxation value.
- $t(s)$: average CPU processing time in seconds (total time).

- $Rt(s)$: average CPU processing time in seconds of the linear relaxation.
- Ni : number of instances solved to full optimality.

3.4.2 Efficiency of problem formulation and benders decomposition on multi store instances

Table 3.3 shows the comparison between M2, M3 and M4 in terms of GAP, GAP_{LP} and $TIME$, using the default CPLEX's Branch & Cut. The formulations M3 and M4 use the subset of constraints described in Section 3.3. The results show that M4 dominates M3, which in turn dominates M2 in terms of GAP. Except for three instances, M4 dominates M3 in GAP_{LP}, and M3 always dominates M2, suggesting that M4 has the tightest LP relaxation. In terms of $TIME$, there is no clear winner.

Table 3.3: Comparison of GAP, $RGAP$ and $t(s)$ between the formulations M2. M3 and M4. $|\mathcal{J}| = 4$.

Instances	(M2)			(M3)			(M4)			Instances	(M2)			(M3)			(M4)		
$\lvert\mathcal{I}\rvert\_\lvert\mathcal{K}\rvert$	GAP	$RGAP$	$t(s)$	GAP	$RGAP$	$t(s)$	GAP	$RGAP$	$t(s)$	$\lvert\mathcal{I}\rvert\_\lvert\mathcal{K}\rvert$	GAP	$RGAP$	$t(s)$	GAP	$RGAP$	$t(s)$	GAP	$RGAP$	$t(s)$
100-20	0.00	4.18	25	0.00	2.31	23	0.00	2.55	23	250-20	0.00	7.48	99	0.00	3.10	48	0.00	3.11	78
100-30	0.00	3.06	62	0.00	2.22	62	0.00	2.13	41	250-30	0.00	5.64	615	0.00	3.66	417	0.00	3.25	510
100-40	0.00	3.07	244	0.00	2.67	224	0.00	2.38	110	250-40	0.00	4.40	1389	0.00	3.62	1365	0.00	3.24	1073
100-50	0.00	1.82	423	0.00	1.60	113	0.00	1.49	86	250-50	2.06	5.11	3600	1.78	4.27	3600	0.78	3.75	3050
100-80	0.00	1.41	752	0.00	1.31	346	0.00	1.16	335	250-80	2.78	4.20	3600	2.50	3.59	3600	2.23	3.62	3600
160-20	0.00	5.17	33	0.00	2.61	56	0.00	2.59	54	400-20	0.00	8.17	269	0.00	3.30	286	0.00	3.17	556
160-30	0.00	3.89	278	0.00	2.92	348	0.00	2.72	174	400-30	0.49	6.33	1882	0.00	3.65	2013	0.00	3.40	2306
160-40	0.00	3.29	1220	0.00	2.71	675	0.00	2.47	611	400-40	2.25	5.77	3600	1.83	4.28	3600	1.11	3.61	3600
160-50	0.29	2.48	2124	0.09	2.23	1513	0.00	2.06	1049	400-50	3.13	5.16	3600	2.72	4.23	3600	1.74	3.57	3600
160-80	0.67	1.94	3094	0.25	1.76	2961	0.12	1.60	2455	400-80	3.75	4.72	3600	3.22	3.87	3600	2.26	3.02	3600

In Table 3.4 we compare the results obtained by different existing solution methods on the M2 formulation. Column M2 indicates default CPLEX as before, column M2-CPLEX-BD is CPLEX with its available Benders decomposition strategy, and M2-BD uses the algorithm proposed in [14]. In the largest instances, the best results were those of M2-BD, while for medium and small instances, the best results were obtained with M2-CPLEX-BD.

Table 3.4: Computational results of model M2. $|\mathcal{J}| = 4$

Instances	M2-CPLEX-BD			M2-BD			M2			Instances	M2-CPLEX-BD			M2-BD			M2		
$\lvert\mathcal{I}\rvert\_\lvert\mathcal{K}\rvert$	GAP	Ni	$t(s)$	GAP	Ni	$t(s)$	GAP	Ni	$t(s)$	$\lvert\mathcal{I}\rvert\_\lvert\mathcal{K}\rvert$	GAP	Ni	$t(s)$	GAP	Ni	$t(s)$	GAP	Ni	$t(s)$
100-20	0.00	5	25	0.01	5	18	0.00	5	36	250-20	0.00	5	99	0.01	5	163	0.00	5	173
100-30	0.00	5	62	0.01	5	67	0.00	5	81	250-30	0.00	5	615	0.07	4	1.508	0.05	4	1.152
100-40	0.00	5	244	0.01	5	671	0.00	5	301	250-40	0.00	5	1.389	0.47	4	2.813	0.29	3	2.239
100-50	0.00	5	423	0.01	5	441	0.00	5	131	250-50	2.06	-	3.600	2.18	-	3.600	1.86	-	3.600
100-80	0.00	5	752	0.01	5	429	0.00	5	428	250-80	2.78	-	3.600	2.26	-	3.600	2.70	-	3.600
160-20	0.00	5	33	0.01	5	32	0.00	5	67	400-20	0.00	5	269	0.01	5	273	0.00	5	487
160-30	0.00	5	278	0.01	5	595	0.00	5	536	400-30	0.49	4	1.882	0.88	2	3.440	0.25	4	2.492
160-40	0.00	4	1.220	0.15	4	1.601	0.01	4	1.406	400-40	2.25	-	3.600	2.16	-	3.600	2.19	-	3.600
160-50	0.29	3	2.124	0.47	2	2.260	0.17	3	2.204	400-50	3.13	-	3.600	2.55	-	3.600	2.89	-	3.600
160-80	0.67	1	3.094	0.57	1	3.087	0.47	1	2.964	400-80	3.75	-	3.600	2.64	-	3.600	3.13	-	3.600

Comparing Tables 3.3 and 3.4, it is clear that the method that dominates in both GAP and TIME (except for instance 400-4-20), is M4. This method even outperforms the Benders decomposition approaches on multiple-store instances.

3.4.3 The effectiveness of constraints (3.7)

To assess the effect of constraints (3.7), we added all of them to problem M3 for small instances and compare the results in terms of gap, linear relaxation run time, and total run time with solving problem M3 without these constraints. We slightly modified M3 by adding all constraints (3.3), rather than a subset of them. We call this model M3*. We set the product capacity of all stores to two. Table 3.5 compares model M3* with and without constraints (3.7). Adding these constraints clearly reduces GAP_{LP} and, although $TIME_{LP}$ increases, the total solution $TIME$ is reduced in the largest instances. We remark that we did not use model M4 in this comparison, due to the high computer cost for generating all the constraints (3.4).

Table 3.5: Comparison of $RGAP$, TIME and $Rt(s)$ between formulations (M3) and (M3) plus (3.7). $|\mathcal{J}|$ and store capacity p = 5

Instance	M3*			(M3*) + (3.7)		
$\lvert\mathcal{I}\rvert$_$\lvert\mathcal{K}\rvert$	$RGAP$	$TIME$	$Rt(s)$	$RGAP$	$TIME$	$Rt(s)$
30-4-5	1.14	0.03	0.01	0.00	0.05	0.01
35-4-15	0.56	0.71	0.09	0.13	1.26	0.28
40-4-25	1.41	3.52	0.27	0.11	2.11	0.98
50-4-30	0.77	14.07	0.82	0.19	17.73	3.71
50-4-40	0.92	27.02	1.24	0.35	29.64	4.55
50-8-30	1.25	150.81	4.48	0.25	92.25	13.07
50-8-35	0.89	659.93	12.67	0.04	169.12	35.71
50-8-40	0.84	1127.90	15.61	0.14	365.75	46.05
50-8-45	0.92	1010.72	15.90	0.27	518.06	41.39
50-8-50	0.39	665.13	19.29	0.02	298.21	61.65

In Table 3.6 we explore the efficiency of valid inequalities (3.7) on large instances. In this case we considered model M4 incorporating inequalities (3.7) using *C&B* (with CPLEX default cuts turned off) and compared its gap, the number of solved instances, and solution time with that solving model M4 and model M4^o (with CPLEX default cuts off). We observe that model M4^o obtains a similar efficiency to model M4, showing that CPLEX default inequalities do not significantly influence the solution of model M4 and that *C&B*, adding cuts (3.7) at the root node, dominating both M4 and M4^o.

We finish the evaluating the impact of constraint (3.7), solving the single store instances in [14] with B&C and C&B and comparing it to the Benders' method proposed in that paper. As Table 3.7 shows, all methods arrived at the optimal solution. For the small and medium instances, the C&B obtained the best results. When the number of clients exceeds 500, and the number of products is 50 or more, the Benders' method solves most of the instances, faster.

3.4.4 Evaluation of decomposition methods on multi store instances

Considering that M2 with the Benders Decomposition approach of [14] is the best strategy for the one store problem, we compare its results with the *B&C* and *C&B* approaches applied to model M4 for multiple-store cases. Table 3.8 and Table 3.9 show the average GAP and $TIME$ over five instances for each $|\mathcal{I}|$_$|\mathcal{J}|$_$|\mathcal{K}|$ combination. In the three methods, we

Table 3.6: Comparison of valid inequality (3.7) with default CPLEX cuts and $C\&B$. Store capacity $p = 5$

Instance	$\vert\mathcal{J}\vert = 4$						$\vert\mathcal{J}\vert = 8$						$\vert\mathcal{J}\vert = 12$					
	$C\&B$		(M4^o)		(M4)		$C\&B$		(M4^o)		(M4)		$C\&B$		(M4^o)		(M4)	
$\vert\mathcal{I}\vert\_\vert\mathcal{K}\vert$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$	GAP	$TIME$
100-20	0,00	2	0,00	20	0,00	23	0,00	3	0,08	1078	0,00	753	0,00	7	0,46	1934	0,41	1783
100-30	0,00	3	0,00	34	0,00	41	0,00	10	0,10	1650	0,00	1398	0,00	13	0,53	2913	0,59	3357
100-40	0,00	8	0,00	134	0,00	110	0,00	15	0,37	1900	0,38	2132	0,00	31	1,49	2649	1,69	3025
100-50	0,00	10	0,00	91	0,00	86	0,00	21	0,04	1606	0,03	1445	0,00	17	0,14	2364	0,14	2715
100-80	0,00	21	0,00	215	0,00	335	0,00	46	0,35	3125	0,36	3172	0,00	50	0,47	3600	0,54	3600
160-20	0,00	5	0,00	43	0,00	54	0,00	10	0,16	2304	0,28	2285	0,00	16	1,21	3367	1,29	3496
160-30	0,00	13	0,00	188	0,00	174	0,00	21	0,65	3460	0,72	3318	0,00	37	1,99	3600	2,01	3600
160-40	0,00	55	0,00	391	0,00	611	0,00	52	2,21	3600	2,16	3600	0,00	172	3,15	3600	3,21	3600
160-50	0,00	33	0,00	1148	0,00	1049	0,00	71	1,06	3308	0,87	3301	0,00	239	2,67	3600	2,98	3600
160-80	0,00	247	0,18	2435	0,12	2455	0,00	474	1,51	3600	1,73	3600	0,00	733	2,18	3600	2,81	3600
250-20	0,00	14	0,00	68	0,00	78	0,00	33	0,29	1223	0,14	1676	0,00	51	2,16	3600	2,38	3600
250-30	0,00	37	0,00	364	0,00	510	0,00	331	2,17	3550	2,35	3600	0,00	297	3,50	3600	3,83	3600
250-40	0,00	45	0,00	921	0,00	1073	0,00	661	2,35	3600	2,07	3600	0,00	318	3,89	3600	4,49	3600
250-50	0,00	718	0,72	3229	0,78	3050	0,00	491	3,33	3600	3,41	3600	0,00	1166	5,32	3600	5,01	3601
250-80	0,00	944	1,68	3600	2,23	3600	0,03	1703	3,40	3600	3,83	3601	0,12	2398	4,10	3601	11,90	3600
400-20	0,00	50	0,00	386	0,00	556	0,00	257	3,17	3029	2,85	3279	0,00	260	4,77	3600	5,00	3600
400-30	0,00	123	0,00	2189	0,00	2306	0,00	607	3,89	3600	3,64	3600	0,00	1044	5,73	3600	4,94	3601
400-40	0,00	308	0,99	3600	1,11	3600	0,00	636	4,10	3600	3,76	3600	0,03	1224	4,48	3600	4,24	3601
400-50	0,00	609	1,83	3600	1,74	3600	0,03	2360	3,68	3600	3,67	3601	2,71	2480	4,28	3601	4,10	3600
400-80	0,15	2745	2,55	3600	2,26	3600	0,48	3600	18,71	3601	25,45	3600	14,11 [2]	3600	16,21	3600	17,66	3600

Table 3.7: Comparison of run TIME of instances [14]. Stores number $|\mathcal{J}| = 1$

Instance	p = 5			p = 10			p = ∞		
$\vert\mathcal{I}\vert\_\vert\mathcal{K}\vert$	(M2)-BD	$B\&C$	$C\&B$	(M2)-BD	$B\&C$	$C\&B$	(M2)-BD	$B\&C$	$C\&B$
100-20	0.67	0.17	0.10	0.62	0.17	0.08	0.64	0.17	0.07
200-20	1.25	0.51	0.29	1.19	0.55	0.33	1.13	0.51	0.19
500-20	3.00	8.97	1.27	2.90	3.94	0.71	2.69	4.22	0.60
1000-20	5.57	21.53	2.75	5.43	12.05	1.98	4.85	11.92	1.69
100-50	1.81	1.69	0.67	2.21	0.47	0.40	2.19	0.43	0.31
200-50	9.39	7.20	2.49	4.53	1.94	1.59	3.98	1.52	1.18
500-50	9.39	71.62	34.36	10.75	24.03	11.14	9.04	20.26	7.90
1000-50	20.14	323.08	328.47	23.20	128.84	112.09	19.17	71.83	101.68
100-100	5.48	6.27	3.39	7.73	2.03	2.02	8.23	1.65	1.35
200-100	14.27	30.66	20.15	45.81	20.97	11.7	13.25	9.68	6.58
100-200	19.08	21.52	18.29	24.50	18.98	15.33	30.91	10.28	7.97
200-200	33.22	94.98	77.72	50.83	65.40	54.24	55.11	35.07	23.92
500-200	73.76	1295.61	1154.98	194.86	598.24	577.19	128.45	315.86	248.01
1000-200	205.63	3600.67	7550.79	482.51	3029.24	6379.81	407.56	1571.28	2229.27
100-500	90.56	132.34	107.86	123.72	98.67	77.63	216.71	56.34	38.14
200-500	189.44	593.00	722.58	314.71	381.79	395.58	527.32	225.23	241.53
500-500	448.54	3523.83	3492.77	1106.05	3024.71	2748.98	1114.97	1634.18	954.68
500-100	25.16	344.96	301.00	32.65	104.55	122.22	30.20	57.94	73.91
1000-100	60.73	1733.49	2339.39	54.41	367.52	334.23	51.57	283.97	223.27

deactivated any cuts generated by default in CPLEX, and used as cut the valid inequality (3.7). The best results were obtained by $C\&B$ in terms of GAP and $TIME$. In fact, this approach obtained the optimal solution for most instances.

As Table 3.8 and Table 3.9 show, as the store capacity increases, the time required to solve the problem tends to decrease. The problem solved to full optimality within the one-hour time limit in almost all cases using the B&C and C&B approaches, except in the few cases of the largest instances, where these two methods are able to solve one more instance than the

Table 3.8: Comparison of GAP and run $TIME$s. Store capacity $p = 5$

Instance	$\lvert\mathcal{J}\rvert$= 4									$\lvert\mathcal{J}\rvert$ = 8									$\lvert\mathcal{J}\rvert$ = 12								
	C&B			(M2)-BD			B&C			C&B			(M2)-BD			B&C			C&B			(M2)-BD			B&C		
$\lvert\mathcal{I}\rvert$_$\lvert\mathcal{K}\rvert$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$
100-20	0,00	5	2	0,01	5	17	0,00	5	17	0,00	5	3	0,01	5	1214	0,14	4	1135	0,00	5	7	1,13	1	3224	0,61	3	2571
100-30	0,00	5	3	0,01	5	91	0,00	5	36	0,00	5	10	1,17	1	3026	0,07	3	1979	0,00	5	13	1,48	0	3600	0,33	3	2968
100-40	0,00	5	8	0,05	4	1009	0,00	5	127	0,00	5	15	1,70	2	2937	0,37	3	1789	0,00	5	31	2,58	0	3600	1,60	2	2609
100-50	0,00	5	10	0,07	4	794	0,00	5	65	0,00	5	21	0,92	1	3212	0,03	4	1536	0,00	5	17	2,46	1	3440	0,13	3	2457
100-80	0,00	5	21	0,01	5	1274	0,01	5	298	0,00	5	46	0,77	1	2931	0,39	2	3099	0,00	5	50	0,87	0	3600	0,42	0	3600
160-20	0,00	5	5	0,01	5	33	0,00	5	37	0,00	5	10	2,41	0	3600	0,23	3	2282	0,00	5	16	3,57	0	3600	1,38	1	3368
160-30	0,00	5	13	0,01	5	710	0,00	5	185	0,00	5	21	3,78	0	3600	0,75	1	3407	0,00	5	37	3,86	0	3600	2,10	0	3600
160-40	0,00	5	55	0,25	3	1861	0,00	5	443	0,00	5	52	2,14	0	3600	2,25	0	3600	0,00	5	172	1,30	0	3600	2,80	0	3600
160-50	0,00	5	33	0,28	2	2273	0,00	5	1069	0,00	5	71	1,19	0	3600	1,02	1	3373	0,00	5	239	3,21	0	3600	2,50	0	3600
160-80	0,00	5	247	0,61	1	3165	0,11	3	2156	0,00	5	474	1,93	0	3600	1,43	0	3600	0,00	5	733	2,77	0	3600	2,47	0	3600
250-20	0,00	5	14	0,01	5	124	0,00	5	61	0,00	5	33	1,09	2	2974	0,21	4	1325	0,00	5	51	4,30	0	3600	1,87	0	3600
250-30	0,00	5	37	0,23	4	1486	0,00	5	301	0,00	5	331	4,49	0	3600	2,18	0	3600	0,00	5	297	5,08	0	3600	3,59	0	3600
250-40	0,00	5	45	0,70	2	3021	0,00	5	1043	0,00	5	661	4,84	1	2963	2,49	0	3600	0,00	5	318	4,85	0	3600	4,07	0	3600
250-50	0,00	5	718	2,34	0	3600	0,72	2	3449	0,00	5	491	4,05	0	3600	3,37	0	3600	0,00	5	1166	6,68	0	3600	4,92	0	3600
250-80	0,00	5	944	2,31	0	3600	1,74	0	3600	0,03	4	1703	5,78	0	3600	3,02	0	3600	0,12	3	2398	7,11	0	3600	4,32	0	3600
400-20	0,00	5	50	0,01	5	377	0,00	5	264	0,00	5	257	5,49	0	3600	3,12	1	2988	0,00	5	260	4,42	0	3600	4,82	1	3404
400-30	0,00	5	123	1,05	0	3600	0,00	5	1715	0,00	5	607	4,27	0	3600	3,91	0	3600	0,00	4	1044	5,50	0	3600	5,72	0	3600
400-40	0,00	5	308	2,48	0	3600	0,80	0	3600	0,00	5	636	4,88	0	3600	4,19	0	3600	0,03	4	1224	5,77	0	3600	4,48	0	3600
400-50	0,00	5	609	2,70	0	3600	1,66	0	3600	0,03	3	2360	3,27	0	3600	3,71	0	3600	2,71	3	2480	4,26	0	3600	4,61	0	3600
400-80	0,15	2	2745	2,56	0	3600	2,56	0	3600	0,48	0	3600	5,97	0	3600	9,23	0	3600	14,11[3]	0	3600	7,38	0	3600	17,74	0	3600

M2-BD.

Table 3.9: Comparison of GAP and run $TIME$s. Store capacity $p = 10$

Instance	$\lvert\mathcal{J}\rvert$ = 4									$\lvert\mathcal{J}\rvert$ = 8									$\lvert\mathcal{J}\rvert$ = 12								
	C&B			M2-BD			B&C			C&B			M2-BD			B&C			C&B			M2-BD			B&C		
$\lvert\mathcal{I}\rvert$_$\lvert\mathcal{K}\rvert$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$	GAP	Ni	$TIME$
100-20	0.00	5	1	0.00	5	0	0.00	5	15	0.00	5	3	0.00	5	2	0.10	4	966	0.00	5	6	0.00	5	7	0.55	3	2585
100-30	0.00	5	2	0.00	5	1	0.00	5	26	0.00	5	8	0.01	5	12	0.11	3	1892	0.00	5	11	0.00	5	16	0.51	2	2811
100-40	0.00	5	5	0.00	5	4	0.00	2	94	0.00	5	11	0.00	5	16	0.35	v3	1756	0.00	5	25	0.00	5	134	1.64	2	2616
100-50	0.00	5	4	0.00	5	2	0.00	5	37	0.00	5	13	0.00	5	29	0.02	4	1427	0.00	5	17	0.00	5	8	0.13	3	1952
100-80	0.00	5	9	0.00	5	18	0.00	5	143	0.00	5	29	0.00	5	5	0.32	2	2872	0.00	5	36	0.00	5	37	0.42	0	3600
160-20	0.00	5	3	0.00	5	2	0.00	5	27	0.00	5	9	0.01	5	10	0.14	3	2204	0.00	5	16	0.00	5	53	1.29	1	3428
160-30	0.00	5	5	0.00	5	3	0.00	5	102	0.00	5	15	0.00	5	117	0.73	2	3532	0.00	5	41	0.00	5	193	2.12	0	3600
160-40	0.00	5	18	0.00	5	34	0.00	5	401	0.00	5	47	0.00	5	41	2.06	0	3600	0.00	5	121	0.01	5	161	3.05	0	3600
160-50	0.00	5	13	0.00	5	11	0.00	5	574	0.00	5	37	0.01	5	266	0.95	1	2193	0.00	5	146	0.01	5	430	2.51	0	3600
160-80	0.00	5	46	0.01	5	125	0.09	4	1847	0.00	5	204	0.06	4	1358	1.36	0	3600	0.00	5	331	0.13	2	2840	2.42	0	3600
250-20	0.00	5	9	0.00	5	3	0.00	5	39	0.00	5	29	0.00	5	10	0.15	4	1275	0.00	5	49	0.00	5	27	2.18	0	3600
250-30	0.00	5	25	0.00	5	11	0.00	5	187	0.00	5	339	0.00	5	110	2.18	1	3533	0.00	5	316	0.01	5	324	3.57	0	3600
250-40	0.00	5	23	0.00	5	23	0.00	5	612	0.00	5	257	0.00	5	122	2.23	0	3600	0.00	5	222	0.00	5	676	4.44	0	3600
250-50	0.00	5	213	0.00	5	420	0.56	3	2924	0.00	5	313	0.00	5	382	3.15	0	3600	0.00	5	651	0.00	5	210	5.12	0	3600
250-80	0.00	5	212	0.01	5	532	0.93	0	3600	0.00	5	843	0.79	5	1062	2.52	0	3600	0.03	4	1431	0.01	4	1909	3.34	0	3600
400-20	0.00	5	48	0.00	5	17	0.00	5	191	0.00	5	256	0.00	5	335	3.05	1	2978	0.00	5	251	0.11	3	1790	4.81	1	3395
400-30	0.00	5	87	0.00	5	49	0.01	4	1750	0.00	5	536	0.00	5	888	3.95	0	3600	0.00	5	981	0.08	3	2130	5.69	0	3600
400-40	0.00	5	156	0.00	5	91	0.47	1	3503	0.00	5	333	0.19	2	2423	3.94	0	3600	0.03	4	1115	0.32	1	3257	4.54	0	3600
400-50	0.00	5	424	0.01	5	394	1.16	1	3556	0.02	4	1359	0.32	2	2539	3.77	0	3600	0.12	3	2320	1.23	1	3197	4.41	0	3600
400-80	0.00	5	733	0.00	5	932	1.34	0	3600	0.30	2	2715	6.44	0	4012	10.61	0	3600	7.42	1	3576				12.28	0	3600

3.5 Conclusions

We adapted valid inequalities of the FLPCP to our problem, to improve the LP relaxation. As the number of valid inequalities is large, we use *B&C* and *C&B* strategies to solve the problem. The numerical experiments were done using published data, including that in [14]. These experiments show that our approach not only solves a previously unsolved problem, but it also obtains better results in synthetic data than the best approach known so far, proposed in [14]. For one store, using synthetic data in [14], the Benders decomposition algorithm solved the instances with 500 and 1000 clients, faster.

There are several pathways to expand the research in this chapter. For products in the category of appliances, it is reasonable to consider that customers purchase at most one unit. For other categories of products, however, e.g., clothing, consumers can buy a bundle of products [47], which is an extension worth exploring. For this extension, there are several possibilities: the customer can purchase either all the products in the bundle in one store or she can visit several stores, purchasing part of the bundle in each of them. The travel cost will be included in the customer's decision. Adding temporal decisions of both of the consumer and the firm, to include inventory, product obsolescence, and markdown pricing, can be considered in future research.

Chapter 4

Mixed-Integer Programming for Combinatorial Coalition Formation Problem

4.1 Introduction[1]

Digital marketplaces or e-marketplaces have contributed to greater interaction between purchasers and merchants since they facilitate global deals and enable the later to provide goods, services to a higher number of buyers without geographical or timing restrictions. The benefits of the e-marketplaces allow the sellers, on the one hand, to enlarge the demand and increase their sales, and on the other hand, the buyers obtain a greater supply variety that lets them select better prices, quality, features, or faster delivery of their products. In this context, an opportunity from e-commerce is to offer wholesale products, i.e., to sell a large number of products in an exclusive exchange with a discount on the unit price [74]. Wholesale pricing has several benefits to the merchant or seller, such as less marketing costs than selling through retailers, a focus on coordinating the logistics, and increasing the market share. Although there are several benefits for the buyers, these are frequently unable or unwilling to purchase a large volume of products, reducing their bargaining power. To skip this limitation, the buyers can attempt to form groups to take advantage of the wholesale price, where the price per unit to be paid depends on the number of buyers in a cluster.

Technology has simplified the buyers' coordination through platforms that make it possible for groups to purchase specific products. Normally in such platforms, a coordinator collects certain buyer information, such as reservation price, their preferences on products and quantity. With this data, it is possible to calculate the purchasing group and to allocate the profit to every member. Buyers can then accept the coalition or not; should they decline, the coordinator will recalculate again until an agreement is reached [113, 91].

[1]This work has been partially funded by grants ANID-PFCHA/Doctorado Nacional/2016 # 21161328; INRIA Associated Team BIPLOS; CONICYT PIA AFB180003; FONDECYT 1190064; Fonds de la Recherche Scientifique -FNRS under Grant PDR T0098.18, and the the supercomputing infrastructure of the NLHPC (ECM-02) of the Universidad de Chile. M.L.C. and H.X.

Furthermore, buyers may wish to purchase more than one item. Because of this there exist bundles with more than one item or service at a more attractive price than if it were sold separately [47]. Hence, retail can extract the surplus from different customers by simply incorporating substitutes and complementary products. In addition, retail can offer bundles with low-demand products together with high-demand items, reducing the corresponding stockpiles and inventory costs. Under these conditions, [88] considers a Combinatorial Coalition Formation (CCF) problem where buyers can buy more than one unit of a product and declare a reservation price for bundles. A virtual reservation price for each item is calculated, and then it is divided into sub coalitions per item. If the set of subcoalitions are compatible, i.e., each buyer belongs to all subcoalitions or none, they complete a coalition. Afterwards, a calculation of the reservation price must be made, again, and the sub coalition set is to be found until they are compatible. All of this proves that in the case of a decreasing linear price in the number of products, once this algorithm finds a set of compatible subcoalitions, the coalitions to be found are optimal. Moreover, there is a calculation of the core with the corresponding payoff for each buyer. In addition, checking if a core for the optimal coalition structure exists or not is NP-complete unless its value is explicitly provided. Hence, [76] presents an integer programming (IP) method to check core-non-emptiness. They defined the algorithm CoreP that finds the value of the coalition $V(CS*)$ by solving an IP and later checks if the value $V(CS*)$ can be divided without coalition members deviating by solving a linear programming (LP) problem. They also propose an exact algorithm called CoreD, which utilizes the dual problem of the linear relaxation of the above IP problem, and, they introduce a weak $\varepsilon - core$ to find a second-best payoff vector when the core is empty. Computational experimental showed that CoreD is superior to CoreP when the core is empty. In [89] there is a CCF in which the reservation price is for a bundle confirming if the items are complementary or substitute, considered a heterogeneous preference of buyers. In contrast to the previously mentioned work, there is a proposal of a coalition for bundles instead of products. Through this, it is proven that the CCF problem reduces to the Complete Set Partitioning, which is NP-hard. A greedy algorithm is therefore proposed for the CCF problem. Finally, they proposed a cost-sharing rule to allocate the payment for each member of coalitions. This rule must satisfy the budget balance (the total payment at least cover the total cost), individual rationality (all individual payments are at most reserve prices), and ensure the stability of coalitions, in this case, avoiding deviations within each bundle coalition.

Wholesale prices are widely used in retail to represent discounts applied to large volume sales. Such discounts are applied when the volume sold reaches a certain threshold. It is therefore natural to represent the wholesale price of each unit of product as a decreasing step function of the volume purchased. In this case, the price is a constant within ranges of purchase volume. In general, a step function is a piece-wise linear function, for which there exist three types formulations in the literature: namely the multiple-choice model, the incremental cost formulation, and the convex combination model [40]. We note that it is possible to approximate any nonlinear function with a piece-wise linear function, where the accuracy depends on the size of its linear segments.

In order to solve generic problems with piece-wise linear functions [41] proposes using a special ordered set of type 2 (SOS2). A set of binary variables are said to be SOS2 when at most two adjacent variables are nonzero. In [120] the branch-and-cut algorithm for LPs with piece-wise linear continuous costs is extended to the lower semicontinuous case. They also extend the SOS2 formulation for LPs with piece-wise linear continuous costs to the lower semicontinuous case. The extension of such formulations to the multivariate nonseparable case is studied in [121]. The paper [81] studies the polyhedral structure of piecewise linear optimization problems and derives strong valid inequalities.

The piece-wise linear function has been applied used in many real-life applications. For instance, the multi-choice model has been used in multi-commodity network flow problem [109], transportation costs [36], facility location problem [68], capacitated location problem with freight cost discount [53], and network design [52], to name a few. The incremental model, in turn, has been applied in facility location problem [69], capacitated local access network problem [104], multi-commodity network design with piecewise cost [13]. A step cost function is used as edge weights to represent economies of scale in [104]. Two formulations are considered to represent this cost functions, based on the incremental and the multi-choice models.

We introduce two different MIP formulations to solve the CCF. The first formulation involves a generic non-increasing pricing function. With this purpose, the function is discretised, according to the number of customers –or buyers. Therefore, the size of such formulation quickly increases in the number of variables, which deepnds on the number of participants. The second formulation applies a step function, which is widely implemented in wholesalers and embraces what is shown in [121]. Moreover, strengthened procedures are included for each instance. Finally, we demonstrate a Benders Decomposition approach to solve the *step prices* model. Computational results and its performance show that this is a bona fide method to solve the problem.

The chapter is organised as follows: Section 4.2 introduces the problem, the bi-level formulation, three different single-level formulations, and some valid inequalities. In Section 4.3, the suggested solution methods are described. Computer testing that compares the different formulations and methods is presented in Section 4.4. Finally, we give our conclusions and expansion lines of eventual work in Section 4.5.

4.2 Problem definition and formulations

This Section presents a description of the Combinatorial Coalition Formation Problem (CCF) we propose. We provide two different formulations for CCF: the first can be used for any price function, while the second is for the step price function only. Moreover, we introduce valid inequalities that strengthen the formulations.

4.2.1 Problem description

Suppose that a market offers multiple products with non-additive values (i.e., items may be complementary or substitute). A set $\mathcal{K}$ of products k is available as well as a set $\mathcal{B}$ of different bundles. Each bundle $b_i \in \mathcal{B}$ is defined as a nonempty set of products such that $b_i \subseteq \mathcal{K}$. Let $p_k(n)$ be a non-increasing function of the price where n is the number of buyers of the product k. In addition, there exists a coordinator who groups together a set $\mathcal{J}$ of consumers or potential buyers j wishing to take advantage of the wholesale price. Each buyer j wishes to buy at most one bundle $b_i \in \mathcal{B}_j$ where $\mathcal{B}_j \subset \mathcal{B}$. They have heterogeneous preferences over the items k in the bundles. Each consumer $j \in \mathcal{J}$ has a reservation price $r_{i,j}$ for bundle $b_i \in \mathcal{B}_j$.

A product coalition C_k is defined as the set of buyers who purchase product k and the price of item K is consequently $p_k(|C_k|)$. Homologously, a bundle coalition C_{b_i} is the group of buyers who buy the bundle b_i. Hence we can defined $C_k = \cup_{b_i \ni k} C_{b_i}$. Therefore, a coalition configuration denoted by C is a set of bundle coalitions C_{b_i} $b_i \in \mathcal{B}$ such that each consumer j belongs to at most one bundle coalition. The surplus $v(C)$ of coalition configuration C is defined as:

$$v(C) = \sum_{b_i \in \mathcal{B}} \left(\sum_{j \in C_{b_i}} r_{i,j} - \sum_{k \in b_i} p_k(|C_k|)|C_k| \right)$$

The coordinator must decide which bundle to propose to each consumer to maximize the surplus obtained by subtracting the bundles' total cost from the sum of the reservation prices. In this case, the utility transfer is essential because the coalition configuration obtained has a non-negative utility. Therefore, although some coalition members could have negative individual utility, their participation helps the prices to reduce enough for receiving greater social welfare, not possible to obtain without their participation. After coalitions formation, the coordinator must determine the payoff of each member of each coalition, according to some criterion (typically of fairness or stability). For instance, [89] used the cost-share as a stability criterion. The members with negative individual utility are assigned their reservation prices -getting a zero utility- and the rest of the participants receive the payoff with the remained surplus.

Table 4.1 and 4.2 provide set and parameter notations, respectively.

Table 4.1: Notation (sets)

$\mathcal{J}$	set of buyers
$\mathcal{K}$	set of products
$\mathcal{B}$	set of bundles
$\mathcal{B}_j$	set of the bundles of customer j

Table 4.2: Notation (parameters)

$r_{i,j}$	reservation price of client $j \in \mathcal{J}$ for bundle $b_i \in \mathcal{B}_j$
$p_k(n)$	unit price for product $k \in \mathcal{K}$ with demand n

Example

We present a small illustrative instance of Combinatorial Coalition Formation in [89]. Assume four buyers a, b, c and d are offered three items g_1, g_2 and g_3, which are camera A, camera B and a flash memory, respectively. The cameras are considered substitutes (though not necessarily perfect substitutes), while any camera and the memory card are complementary goods. Table 4.3 shows the reservation prices of the buyers, while Table 4.4 provides the price-quantity curve for each item, as it specifies the unit price per item as a non-increasing discrete function of the quantity. Now, assume that the bundle coalitions are configured as follows: $C_{2,3} = \{a, b\}$ for $g_{2,3}$, and $C_2 = \{c, d\}$ for g_2. Notice that this is equivalent to the set of item coalitions: $\tilde{C}_2 = \{a, b, c, d\}$ for g_2, and $\tilde{C}_3 = \{a, b\}$ for g_3. Given this configuration, no buying group is formed for g_1, the resulting price for g_2 is 350 (four buyers), and that for g_3 is 40 (two buyers). Thus, the unit price of the bundle $g_{2,3}$ is $350 + 40 = 390$. Then, the total cost of $C_{2,3}$ is $2 \cdot 390 = 780$, and the cost of the C_2 is $2 \cdot 350 = 750$. Therefore, the surplus of the bundle coalition $C_{2,3}$ is $385 + 405 - 780 = 10$, and the surplus of C_2 is $355 + 360 - 700 = 15$. Finally, the total surplus of the coalition configuration is $10 + 15 = 25$.

Table 4.3: Buyers' reservation prices

	g_1	g_2	g_3	$g_{1,3}$	$g_{2,3}$	$g_{1,2}$	$g_{1,2,3}$
a	150	350	20	380	385	500	545
b	380	365	25	400	405	600	650
c	0	355	0	0	360	355	360
d	360	360	0	365	370	510	525

Table 4.4: Price function

	Volume (units)				
	1	2	3	4	≥ 5
g_1	340	340	320	320	320
g_2	365	365	365	350	340
g_3	40	40	38	38	35

Computational complexity

We set first define the decision version of problem CCF as follows:

Instance: A set $\mathcal{J}$ of buyers, a set $\mathcal{K}$ of products, and a set $\mathcal{B} = \{b_1, \ldots, b_{|\mathcal{B}|}\}$ of bundles where $b_i \subseteq \mathcal{K}$. Each buyer $j \in \mathcal{J}$ has a reservation price $r_{i,j}$ for the bundle b_i. The unit price of product k is $p_k(|S|)$ where $|S|$ is number of buyers that purchase product k. An integer number M.

Question: Does there exist a coalition structure $C = \{C_{b_1}, \ldots, C_{b_{|\mathcal{B}|}}\}$ with $C_{b_i} \subset \mathcal{J}$ for all $b_i \in \mathcal{B}$ and $C_{b_i} \cap C_{b_{i'}} = \emptyset$, for all $i, i' \in \mathcal{I}$ and

whose value

$$v(C) = \sum_{b_i \in \mathcal{B}} \left(\sum_{j \in C_{b_i}} r_{i,j} - \sum_{k \in b_i} p_k(|C_k|)|C_k| \right)$$

is at least M?

Theorem 4.2.1 Problem CCF is NP-complete even if all bundles are singletons.

Proof. First, it is easy to see that the decision version of Problem CCF belongs to NP. Indeed, given a coalition structure C, its value $V(C)$ can be computed in polynomial time.

Then, to prove that Problem CCF is NP-hard we show that *Set Packing Problem* reduces polynomially to CCF. First, we define Problem Set Packing (SP) as follows :

- *Instance* : A collection S of finite sets $S_i, i = 1, \ldots, m$ and an integer M.
- *Question* : Does S contains M mutually disjoint sets?

Given an instance of SP, we construct a instance of CCF as follows. The set of buyers $\mathcal{J} = \cup_{i=1}^{m} S_i$ and the set of bundles $\mathcal{B}$ contains m bundles b_j that are all distinct singletons. We define the reservation price $r_{i,j}$ of buyer j for bundle b_i and the price function $p_i(k)$ as follows :

Remark that as $\mathcal{J}$ is the set of customers, $\mathcal{J} \neq \emptyset$ and $|\mathcal{J}| \geq |S_i|$ then $2 \cdot |I| > (|S_i|+1)/|S_i|$. Hence, we can define our problem as $\forall i$ determine a coalition C_i such that each customer

$$r_{i,j} = \begin{cases} \frac{|S_i|+2}{|S_i|} & \text{if } j \in \mathcal{S}_i \\ 0 & \text{if } j \notin \mathcal{S}_i \end{cases} \ ; \quad p_i(k) = \begin{cases} 2 \cdot |\mathcal{I}| & \text{if } k < |S_i| \\ \frac{|S_i|+1}{|S_i|} & \text{if } k \geq |S_i| \end{cases}$$

belongs to at most one coalition and the value of coalition structure is:

$$v(C) = \sum_{i=1}^{m} \left(\sum_{i \in C_i} r_{i,j} - p_i(|C_i|)|C_i| \right) \geq B$$

We now show that there exists a coalition structure C with value $V(C) \geq B$ iff there exist M mutually disjoint sets in $\{S_i\}$. Let define the contribution $v(C_i)$ of product i to the value $V(C)$ of a coalition structure as

$$v(C_i) = \sum_{j \in C_i} r_{i,j} - p_i(|C_i|)|C_i|$$

We prove that $v(C_i)$ is equal to 1 if $C_i = S_i$ and is negative otherwise.

We consider three different possible cases as follows.

(i) if $|C_i| \geq |S_i| + 1$ then

$$\begin{aligned} v(C_i) = \sum_{j \in C_i \cap S_i} r_{i,j} - p_i(|C_i|)|C_i| &= |C_i \cap S_i| \frac{|S_i|+2}{|S_i|} - \frac{|S_i|+1}{|S_i|}|C_i| \\ &\leq |S_i| \frac{|S_i|+2}{|S_i|} - \frac{(|S_i|+1)^2}{|S_i|} = \frac{-1}{|S_i|} < 0. \end{aligned}$$

where the inequality is obtained since $|C_i \cap S_i| \leq |S_i|$ and $|C_i| \geq |S_i| + 1$.

(ii) if $|C_i| \leq |S_i| - 1$ then

$$\begin{aligned} &= |S_i| + 2 - \left(1 + \frac{2}{|S_i|}\right) - 2|\mathcal{I}| = \\ &= |S_i| + 1 - \frac{2}{|S_i|} - 2|\mathcal{I}| < |S_i| + 1 - 2|\mathcal{I}| \leq 1 - |\mathcal{I}| < 0 \end{aligned}$$

where the first inequality is obtained since $|C_i \cap S_i| \leq |S_i| - 1$. The last inequality is obtained due to $|C_i| \leq |\mathcal{I}|$.

(iii) if $|C_i| = |S_i|$ and $C_i \neq S_i$ then $v(C_i) = |C_i \cap S_i| \frac{|S_i|+2}{|S_i|} - \frac{|S_i|+1}{|S_i|}|S_i|$ as $C_i \neq S_i$ implies $|C_i \cap S_i| \leq |S_i| - 1$. Hence,

$$v(C_i) \leq (|S_i| - 1)\frac{(|S_i| + 2)}{|S_i|} - |S_i| - 1 = |S_i| + 2 - 1 - \frac{2}{|S_i|} - |S_i| - 1 = \frac{-2}{|S_i|} < 0$$

Finally, if $C_i = S_i$ then

$$v(C_i) = |S_i| \frac{|S_i + 2|}{|S_i|} - \frac{|S_i + 1|}{|S_i|}|S_i| = |S_i| + 2 - |S_i| - 1 = 1.$$

Therefore, there exists a Set Packing with B sets if and only if there exists a set of coalition with total value B. □

4.2.2 Model formulation

The combinatorial coalition formation problem can be formulated as a mathematical programming problem. We propose two different formulations. The first can be used for general non-increasing price functions while the second can be used for price step functions only. We begin introducing a nonlinear optimization formulation for the coalition formation problem which will be our starting point.

A nonlinear optimization formulation

We define binary variables $y_{i,j}$ that take the value one when the customer j purchases bundle b_i and zero otherwise. We also consider a continuous variable z_k that indicates the number of customers that purchase product k, which corresponds to the size of the coalition C_k. The coalition formation problem can be stated as

$$
\begin{aligned}
(NLP) \quad & \max \sum_{j\in\mathcal{J}} \sum_{b_i\in\mathcal{B}_j} r_{i,j} y_{i,j} - \sum_{k\in\mathcal{K}} p_k(z_k) z_k && && \text{(4.1a)}\\
& s.t. \sum_{b_i\in\mathcal{B}_j} y_{i,j} \leq 1 && (j\in\mathcal{J}), && \text{(4.1b)}\\
& \sum_{j\in\mathcal{J}} \sum_{\substack{b_i\in\mathcal{B}_j\\ b_i\ni k}} y_{i,j} = z_k && (k\in\mathcal{K}), && \text{(4.1c)}\\
& y_{i,j}\in\{0,1\} && (b_i\in\mathcal{B}_j),(j\in\mathcal{J}), && \text{(4.1d)}\\
& z_k \geq 0 && (k\in\mathcal{K}). && \text{(4.1e)}
\end{aligned}
$$

Here the objective function maximizes the social utility (4.1a). Constraints (4.1b) indicate that every customer can buy at most one bundle, and constraints (4.1c) determine the number of customers who buy each product k. The model above is a nonlinear mixed-integer optimization problem, where the nonlinearity is due to the total purchase cost in the objective function.

Given that customer purchasing decisions are integer, the total amount of each product k that is sold, z_k is also an integer, and therefore the values of product prices that are observed turn out to be a discrete group of prices. Let us denote by $p_{k,l}$ the unit price of product k if l units of that product are sold. This means that these price values can be determined in $O(|\mathcal{J}||\mathcal{K}|)$ time. We remark that when the price function of each product is constant, i.e. $p_{k,l} = p_k$, for all possible l, then (NLP) is a mixed integer linear optimization problem and the optimal coalition structure is obtained by assigning each customer to the bundle that maximizes the difference between her reservation price for this bundle and the sum of the prices of the products constituting the bundle.

Considering these observations, we now present two mixed integer linear formulations of the coalition formation problem. The first formulation, *Generic Price Function* (GPF) formulation, can be used for any price function $p_k(z_k)$. In many cases, given its simplicity, the retailer uses wholesale prices that follow a non-increasing step function, where the price remains constant in a purchase range. Once it exceeds the upper limit of the range, the price is reduced and remains constant until the next limit. Although the (GPF) can model this type of function, the number of binary variables it uses may be greater than necessary. Therefore, given the particular structure of the piecewise constant decreasing price function, we introduce the *Step Price Function* (SPF) formulation based on the work [40] who proposed the *multiple-choice model.*

Generic Price Function formulation

Given that number of purchases of every product is integer, we define a binary variable $x_{k,l}$ that is equal to 1 if l customers purchase product k and 0 otherwise, for $k \in \mathcal{K}$ and $l \in \mathcal{L} = \{1, 2, \cdots, |\mathcal{J}|\}$. We use these binary variables to reformulate the problem (NLP) as an linear mixed-integer model, called *Generic Price Function* (GPF), adding the following set of constraints for each product $k \in \mathcal{K}$:

$$\sum_{l \in \mathcal{L}} l \cdot x_{k,l} = z_k, \tag{4.2a}$$

$$\sum_{l \in \mathcal{L}} x_{k,l} \leq 1, \tag{4.2b}$$

$$x_{k,l} \geq 0 \; . \tag{4.2c}$$

Then the nonlinear terms $p_k(z_k)z_k$ in the objective function of (NLP) can be replaced by the following linear expression $p(z_k)z_k = \sum\limits_{l \in \mathcal{L}} l p_{k,l} x_{k,l}$.

Step Price Function formulation

For this case, given a product $k \in \mathcal{K}$ we establish a collection of purchase size thresholds $a_{k,1}, \cdots, a_{k,L_k}$ and corresponding prices $d_{k,1} > d_{k,2} > \cdots > d_{L_k}$. The set of indices of the different segments of the price function of product k is denoted by $\mathcal{L}_k = \{1, \cdots, L_k\}$. By definition, we set $a_{k,0} = 0$ for each product $k \in \mathcal{K}$. Therefore, given that z_k buyers purchase product k, where z_k belongs to the interval $[a_{k,l}, a_{k,l+1})$, the price of product k that these buyers pay is defined as $p_k(z_k) = d_{k,l}$. Figure 4.2.2 illustrates a step price function with this notation.

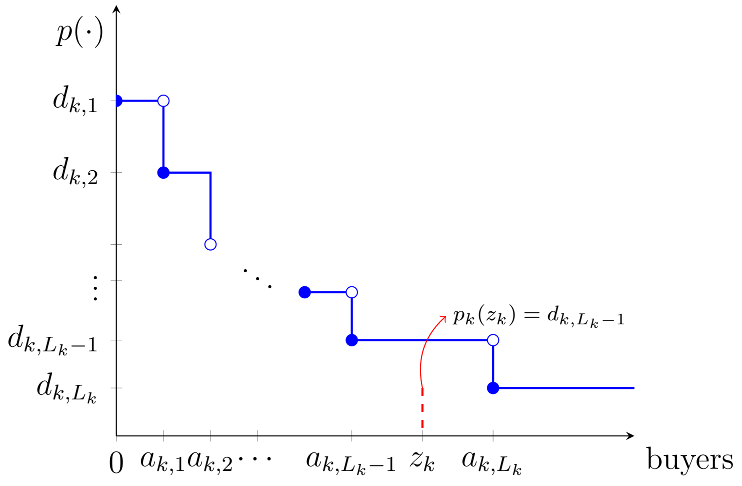

Figure 4.1: Step price function representation

The multi-choice-model [6] uses binary variable $s_{k,l}$ to identify the interval in the price function that is active. That is $s_{k,l} = 1$ if $z_k \in [a_{k,l-1}, a_{k,l})$ for $k \in \mathcal{K}$ and $l \in \mathcal{L}_k$. The model also uses variable $\tilde{z}_{k,l}$ to decompose z_k the total purchases of product k into the intervals of the step function, and finally φ_k a nonnegative variable that represents the total cost paid for product k. With these variables, we can express the coalition formation problem given a step price function as a mixed integer linear optimization problem. We do so by replacing $p(z_k)z_k = \varphi_k$ in the objective function and adding constraints (4.3a)-(4.3e) below for every product $k \in \mathcal{K}$.

$$\sum_{l \in \mathcal{L}_k} \tilde{z}_{k,l} = z_k \tag{4.3a}$$

$$\sum_{l \in \mathcal{L}_k} s_{k,l} = 1 \qquad k \in \mathcal{K}, \tag{4.3b}$$

$$a_{k,l-1} s_{k,l} \leq \tilde{z}_{k,l} \qquad k \in \mathcal{K}, l \in \mathcal{L}_k, \tag{4.3c}$$

$$a_{k,l} s_{k,l} \geq \tilde{z}_{k,l} \qquad k \in \mathcal{K}, l \in \mathcal{L}_k, \tag{4.3d}$$

$$d_{k,l} z_k - (1 - s_{k,l})M \leq \varphi_k \qquad k \in \mathcal{K}, l \in \mathcal{L}_k \ . \tag{4.3e}$$

Constraint (4.3a) establishes the relation between z_k and $\tilde{z}_{k,l}$. Constraint (4.3b) forces that just one interval $l \in \mathcal{L}_k$ must be active; constraints (4.3c) and (4.3d) limit the value $\tilde{z}_{k,l}$ to the bounds of the interval $l \in \mathcal{L}_k$, while the constraint (4.3e) rescues the total cost φ_k of the product $k \in \mathcal{K}$. Notice that the big-M value can be set to d_{k,L_k}. It is well known that big-M constraints lead to poor LP relaxations and long computational times, therefore we consider reformulations of this constraint.

We present computational results for two variants of multi-choice version of the coalition formation problem, (SPF1) and (SPF2), depending on how they modify the big-M constraint (4.3e).

- We refer to (SPF1) (for Step Price Function formulation 1) when we replace constraint (4.3e) with

$$\sum_{l \in \mathcal{L}_k} d_{k,l} \tilde{z}_{k,l} = \varphi_k \quad k \in \mathcal{K} \ . \tag{4.3e-1}$$

  We note that constraint (4.3b) means that only one variable $s_{k,l}$ with $l \in \mathcal{L}_k$ can be active for each product k, therefore $\tilde{z}_{k,l}$ is equal zero for all intervals, except the active one (the one with $s_{k,l} = 1$). Which shows that (4.3e-1) is equivalent to (4.3e).

- We refer to (SPF2) (for Step Price Function formulation 2) when we replace constraint (4.3e) with

$$d_{k,l} z_k + \sum_{l'=l+1}^{L_k} (d_{k,l'} - d_{k,l}) a_{k,l'} s_{k,l'} \leq \varphi_k \quad k \in \mathcal{K}, l \in \mathcal{L}_k \ . \tag{4.3e-2}$$

  This equivalent formulation is obtained by strengthening constraint (4.3e) by considering the difference between the price of the interval l and the sum of variation of price respect the set of intervals less to l.

Our computational experiments show that both (SPF1) and (SPF2) perform better than the formulation with the original big-M constraint (4.3e).

There is an alternative formulation for the case with piecewise constant step function, known as the incremental model [93]. We present this version of the coalition formation problem here for completeness. In the *incremental model* variable $\tilde{z}_{k,l} \in [0, a_{k,l} - a_{k,l-1}]$ and binary variable $s_{k,j} = 1$ for all $1 \leq j \leq l$ with l such that $z_k \in (a_{k,l-1}, a_{k,l}]$. Hence constraint (4.3b) is no longer valid. In the incremental model formulation we replace constraints (4.3b) - (4.3e) for each product by the following set of constraints:

$$\sum_{l \in \mathcal{L}_k} \tilde{z}_{k,l} = z_k \tag{4.4a}$$

$$(a_{k,l} - a_{k,l-1}) \cdot s_{k,l+1} \leq \tilde{z}_{k,l} \qquad l \in \mathcal{L}_k \tag{4.4b}$$

$$(a_{k,l} - a_{k,l-1}) \cdot s_{k,l} \geq \tilde{z}_{k,l} \qquad l \in \mathcal{L}_k \tag{4.4c}$$

$$d_{k,l} z_k - (1 + s_{k,l} - s_{k,l+1}) \cdot M \leq \varphi_k \qquad l \in \mathcal{L}_k \tag{4.4d}$$

Again, the big M can be tightened by using d_{k,L_k} for every $l \in \mathcal{L}_k$ and a similar reformulations to (4.3e-2) of this constraint can be used. We note that both, the *multiple-choice model* and the *incremental model*, consider the same number of total variables, while the *multiple-choice model* adds $\sum_{k \in \mathcal{K}} 4 \cdot L_k + 2$ constraints and the *incremental model* adds $\sum_{k \in \mathcal{K}} 3 \cdot L_k + 1$ additional constraints to (NLP). Recall that $L_k = |\mathcal{L}_k|$. Our preliminary computational results show that both the *incremental model* and *multiple-choice model* have a similar performance. We therefore work in the rest of this thesis with the *multiple-choice model* as it is easier to develop Benders decomposition methods for this model.

4.3 Solution methods

The formulations (GPF) and (SPF) for the CCF presented above are mixed-integer optimization problems that can solve small instances in a reasonable time by a commercial solver. However, real world applications can involve instances with many buyers, products, and bundles that must be solved quickly. Hence, it is necessary to develop algorithms to efficiently find optimal solutions. We focus on an exact decomposition based solution method, in contrast to previous works that have proposed heuristic methods [88, 89, 76]. In particular, we now present a Bender's decomposition strategy for formulation (SPF1).

First, we rewrite the model (SPF1) in order to remove the auxiliary variables $\boldsymbol{\varphi}$ and $\boldsymbol{z}$. We replace variable $\boldsymbol{\varphi}$ in the objective according to equation (4.3e-1) and substitute the variable $\boldsymbol{z}$ using the equation (4.3a) obtaining the following formulation:

$$
\begin{aligned}
\text{(RSPF1)}\ \max & \sum_{j\in\mathcal{J}}\sum_{b_i\in\mathcal{B}_j} r_{i,j}y_{i,j} - \sum_{k\in\mathcal{K}}\sum_{l\in\mathcal{L}_k} d_{kl}\tilde{z}_{k,l} && && \text{(4.5a)}\\
s.t. & \sum_{j\in\mathcal{J}}\sum_{\substack{b_i\in\mathcal{B}j\\ b_i\in k}} y_{i,j} = \sum_{l\in\mathcal{L}_k}\tilde{z}_{k,l} && k\in\mathcal{K}, && \text{(4.5b)}\\
& \sum_{b_i\in\mathcal{B}_j} y_{i,j}\le 1 && j\in\mathcal{J}, && \text{(4.5c)}\\
& \sum_{l\in\mathcal{L}_k} s_{k,l} = 1 && k\in\mathcal{K}, && \text{(4.5d)}\\
& a_{k,l-1}s_{k,l}\le \tilde{z}_{k,l} && k\in\mathcal{K}, l\in\mathcal{L}_k, && \text{(4.5e)}\\
& a_{k,l}s_{k,l}\ge \tilde{z}_{k,l} && k\in\mathcal{K}, l\in\mathcal{L}_k, && \text{(4.5f)}\\
& s_{k,l}\in\{0,1\} && k\in\mathcal{K},\ l\in\mathcal{L}_k, && \text{(4.5g)}\\
& y_{i,j}\in\{0,1\} && b_i\in\mathcal{B}_j,\ j\in\mathcal{J}. && \text{(4.5h)}\\
& \tilde{z}_{k,l}\ge 0 && k\in\mathcal{K}, l\in\mathcal{L}_k. && \text{(4.5i)}
\end{aligned}
$$

Note that due to constraints (4.5b), (4.5c), (4.5e), and (4.5f) the following constraints for $k\in\mathcal{K}$ always holds in the master problem.

$$
\sum_{l\in\mathcal{L}_k} s_{k,l}a_{k,l-1} \le \sum_{j\in\mathcal{J}}\sum_{\substack{b_i\in\mathcal{B}j\\ b_i\in k}} y_{i,j} \le \sum_{l\in\mathcal{L}_k} s_{k,l}a_{k,l}\ . \tag{4.6}
$$

Notice that Benders decomposition applies naturally in (RSPF1) because the integer variables $(\boldsymbol{s},\boldsymbol{y})$ and continuous variables $\tilde{\boldsymbol{z}}$ can be separated leaving the integer variables in the first stage and delaying the continuous variables. Therefore, the master problem decides what bundle each customer purchases and to what price interval the total purchases of each product fall into. The second level problem determines the units sold per product; it can split into a subproblem for each product. To the best of our knowledge, benders decomposition has not been applied previously in CCF. Hence, the master problem becomes:

$$
\begin{aligned}
\max & \sum_{j\in\mathcal{J}}\sum_{b_i\in\mathcal{B}_j} r_{i,j}y_{i,j} - \sum_{k\in\mathcal{K}}\xi_k, && && \text{(4.7a)}\\
s.t. & \sum_{b_i\in\mathcal{B}_j} y_{i,j}\le 1 && j\in\mathcal{J}, && \text{(4.7b)}\\
& \sum_{l\in\mathcal{L}_k} s_{k,l}=1 && k\in\mathcal{K}, && \text{(4.7c)}\\
& \sum_{l\in\mathcal{L}_k} s_{k,l}a_{k,l-1}\le \sum_{j\in\mathcal{J}}\sum_{\substack{b_i\in\mathcal{B}j\\ b_i\in k}} y_{i,j} && k\in\mathcal{K}, && \text{(4.7d)}\\
& \sum_{j\in\mathcal{J}}\sum_{\substack{b_i\in\mathcal{B}j\\ b_i\ni k}} y_{i,j}\le \sum_{l\in\mathcal{L}_k} s_{k,l}a_{k,l} && k\in\mathcal{K}, && \text{(4.7e)}\\
& \alpha_k^e\sum_{\substack{b_i\in\mathcal{B}j\\ b_i\ni k}} y_{i,j} + \sum_{l\in\mathcal{L}_k}\beta_{k,l}^e a_{k,l}s_{k,l} + \sum_{l\in\mathcal{L}_k}\gamma_{k,l}^e a_{k,l-1}s_{k,l}\le \xi_k && k\in\mathcal{K},\ e\in\mathcal{E}_k, && \text{(4.7f)}\\
& \alpha_k^r\sum_{\substack{b_i\in\mathcal{B}j\\ b_i\ni k}} y_{i,j} + \sum_{l\in\mathcal{L}_k}\beta_{k,l}^r a_{k,l}s_{k,l} + \sum_{l\in\mathcal{L}_k}\gamma_{k,l}^r a_{k,l-1}s_{k,l}\ge 0 && k\in\mathcal{K},\ r\in\mathcal{R}_k, && \text{(4.7g)}\\
& y_{i,j}\in\{0,1\} && b_i\in\mathcal{B}_j,\ j\in\mathcal{J}, && \text{(4.7h)}\\
& s_{k,l}\in\{0,1\} && k\in\mathcal{K},\ l\in\mathcal{L}_k. && \text{(4.7i)}
\end{aligned}
$$

Here, for each $k \in \mathcal{K}$, the sets $\mathcal{E}_k$ and $\mathcal{R}_k$ denote the extreme points and extreme rays, respectively, of the polyhedron that defines the feasible region of the second stage problem described below. Due to the sets of extreme points and extreme rays being large, a cut generation algorithm is used to gradually add these constraints.

On the other side, given $(\hat{\boldsymbol{s}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}})$ a feasible solution to master problem, the second stage problem is separable in $|\mathcal{K}|$ sub-problems which we present below:

$$G_k(y,s) = \min_{\tilde{z}_k} \sum_{l \in \mathcal{L}_k} d_{k,l} \tilde{z}_{k,l}, \tag{4.8a}$$

$$s.t. \sum_{j \in \mathcal{J}} \sum_{\substack{b_i \in \mathcal{B}j \\ b_i \in k}} \hat{y}_{i,j} = \sum_{l \in \mathcal{L}_k} \tilde{z}_{k,l} \tag{4.8b}$$

$$a_{k,l-1}\hat{s}_{k,l} \leq \tilde{z}_{k,l} \qquad l \in \mathcal{L}_k, \tag{4.8c}$$

$$a_{k,l}\hat{s}_{k,l} \geq \tilde{z}_{k,l} \qquad l \in \mathcal{L}_k, \tag{4.8d}$$

$$\tilde{z}_{k,l} \geq 0 \qquad l \in \mathcal{L}_k. \tag{4.8e}$$

The cuts for the master problem are obtained as the objective function of the dual of this second level problem evaluated in the optimal solution. The dual of this second level problem is:

$$DG_k(\alpha,\beta,\gamma) = \max_{\alpha_k,\beta_k,\gamma_k} \alpha_k Y_k + \sum_{l \in \mathcal{L}_k} \beta_{k,l} a_{k,l} \hat{s}_{k,l} + \sum_{l \in \mathcal{L}_k} \gamma_{k,l} a_{k,l-1} \hat{s}_{k,l} \tag{4.9a}$$

$$s.t.\ \alpha_k + \beta_{k,l} + \gamma_{k,l} \leq d_{k,l} \qquad (l \in \mathcal{L}_k), \tag{4.9b}$$

$$\beta_{k,l} \leq 0 \qquad (l \in \mathcal{L}_k), \tag{4.9c}$$

$$\gamma_{k,l} \geq 0 \qquad (l \in \mathcal{L}_k). \tag{4.9d}$$

Here, $(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\beta})$ are the dual variables of constraints (4.8b), (4.8c) and (4.8d) respectively and let Y_k denote the units sold of product k, given by

$$Y_k = \sum_{j \in \mathcal{J}} \sum_{\substack{b_i \in \mathcal{B}j \\ b_i \in k}} \hat{y}_{i,j} \qquad \text{for } k \in \mathcal{K}\ . \tag{4.10}$$

Note that constraints (4.7d) and (4.7e) ensure that there exists a feasible solution for problem (4.8) and therefore the master problem will not use *feasibility cuts* (4.7g). Since the second stage problem and its dual are always feasible, then for every master problem solution $(\hat{\boldsymbol{s}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}})$, there is an optimal extreme point for the dual second stage problem. If $\hat{\xi}_k \ngeq G_k(\hat{s}, \hat{y}))$ then this optimal extreme point gives an *optimality cut* (4.7f) that is added to master problem.

Proposition 4.3.1 below gives an analytical expression for the primal solution of the second stage problem, while Proposition 4.3.2 does the same for the optimal solution for the second stage problem. These results help reduce computational time of a Benders decomposition implementation.

Proposition 4.3.1 Given $(\hat{\boldsymbol{s}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}})$ feasible for Problem (4.7). Let $k \in \mathcal{K}$, Y_k from (4.10), and define $l^* = \arg\max_{l \in \mathcal{L}_k} \{\hat{s}_{k,l}\}$. Then the optimal solution to Problem (4.8) is

$$\tilde{z}^*_{k,l} = \begin{cases} Y_k & l = l^* \\ 0 & \text{o/w} \end{cases}$$

with an optimal solution value equal to $G_k(s, y) = Y_k d_{k,l^*}$.

The proof of Proposition 4.3.1 is straightforward, and amounts to verifying that the proposed solution is the only feasible solution.

Proposition 4.3.2 Given $(\hat{\boldsymbol{s}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}})$ feasible for Problem (4.7) and Y_k from (4.10), then the optimal solution to problem (4.9) is given by

$$\alpha_k = d_{k,l^*}, \tag{4.11a}$$

$$\beta_{k,l} = \begin{cases} d_{k,l} - d_{k,l^*} & \text{if } l > l^*, \\ 0 & \text{otherwise.} \end{cases} \quad \forall l \in \mathcal{L}_k, \tag{4.11b}$$

$$\gamma_{k,l} = 0 \ \forall l \in \mathcal{L}_k. \tag{4.11c}$$

with $l^* = \arg\max_{l \in \mathcal{L}_k} \{\hat{s}_{k,l}\}$ and the optimal solution value equals

$$DG_k(\alpha, \beta, \gamma) = Y_k \alpha_k = Y_k d_{k,l^*}$$

Proof. The proof is divided into two steps: first, we show that the proposed solution is feasible and then we show that the objective value of the proposed solution is exactly $d_{k,l^*} Y_k$, which is the optimal objective of the primal problem.

- *Feasible*: Observe that $d_{k,1} > d_{k,2} > \cdots > d_{L_k}$. Thus, we need verify constraint Constraint (4.9b). For $l \leq l^*$, we have that

$$\begin{aligned} \alpha_k + \beta_{k,l} + \gamma_{k,l} &= d_{k,l^*} + 0 + 0 \\ &\leq d_{k,l} \end{aligned}$$

  and for $l > l^*$, then we have that

$$\begin{aligned} \alpha_k + \beta_{k,l} + \gamma_{k,l} &= d_{k,l^*} + (d_{k,l} - d_{k,l^*}) + 0 \\ &= d_{k,l} \end{aligned}$$

  Moreover, how $\gamma_{k,l} = 0 \ \forall l \in \mathcal{L}_k$ the before, also satisfy the constraint (**??**). This concludes our proof of the feasibility $(\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \boldsymbol{\gamma}_k)$.
- *Objective function*: Observe that $\beta_{k,l^*} = 0$. Thus, we have that

$$\begin{aligned} \alpha_k \left(\sum_{\substack{b_i \in \mathcal{B}j \\ k \in b_i}} y_{i,j} \right) + \sum_{l \in \mathcal{L}_k} \beta_{k,l}(b_{k,l} s_{k,l}) + \sum_{l \in \mathcal{L}_k \setminus \{1\}} \gamma_{k,l}(b_{k,l-1} s_{k,l}) &= d_{k,l^*} Y_k + 0 + 0 \\ &= d_{k,l^*} Y_k \end{aligned}$$

  which is exactly the optimal value of the primal problem. Thus, it follows that $(\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \boldsymbol{\gamma}_k)$.

□

For the case of the linear relaxation of the master problem (4.7), it is also possible to obtain the exact solution by way of an efficient use of the Algorithm 5. In this context, the procedure is as follows. Given the purchase size Y_k of product k, we establish as base the interval lower bound for all intervals with a value $\hat{s}_{k,l} > 0$, i.e. $a_{k,l-1}\hat{s}_{k,l} = \tilde{z}_{k,l}$. Then sequentially, decreasing in l from index L_k to 1, we increase each $\tilde{z}_{k,l}$ by $\min\{Y_k - Z_k, a_{k,l}\hat{s}_{k,l} - \tilde{z}_{k,l}\}$ where Z_k accumulates the updated value of $\tilde{z}_{k,l}$ up to this point. We repeat this procedure until $Z_k = Y_k$. When this procedure stops then the last step l^* is identified and completed.

Algorithm 5 Greedy Algorithm for Solving Problems (4.8)

```
 1: Require: $\hat{\boldsymbol{s}}^k$, $Y_k$ from (4.10), set $\mathcal{L}_k$ ordered s.t. $d_{k,L_k} < \cdots < d_{k,1}$
 2: $\tilde{z}_{k,l} = a_{k,l-1}\hat{s}_{k,l}$ for all $l \in \mathcal{L}_k$ and $Z_k = \sum_{l\in\mathcal{L}_k} \tilde{z}_{k,l}$
 3: for $l \in \{L_k, \ldots, 1\}$ do
 4:     $\tilde{z}_{k,l} = \tilde{z}_{k,l} + \min\{Y_k - Z_k, a_{k,l}\hat{s}_{k,l} - \tilde{z}_{k,l}\}$
 5:     $Z_k = Z_k + \min\{Y_k - Z_k, a_{k,l}\hat{s}_{k,l} - \tilde{z}_{k,l}\}$
 6:     if $Z_k = Y_k$ then
 7:         $G_k = \sum\limits_{l\in\mathcal{L}_k} \tilde{z}_{k,l} d_{k,l}$
 8:         $l_k^* \leftarrow l$
 9:         break
10:     end if
11: end for
```

After obtaining the l_k^* from Algorithm 5 we apply the Algorithm 6 to get the dual variables. Similar to the MIP of problem (4.7) we establish nonzero dual variables different to l_k^*. The idea is to add cuts that incorporate both variables $(\boldsymbol{s}, \boldsymbol{y})$ in the master problem. Therefore, we give the value $\alpha_k = d_{k,l_k^*}$. Note that l_k^* can make constraint (4.8c) active or inactive, so for simplicity, we establish the associated dual variables are equal to zero. Additionally, the constraints (4.8d) with $l_k^* > l$ and $s_{k,l} > 0$ are inactive then the dual variable $\beta_{k,l} = 0$ for $l_k^* > l$. On the other hand, the constraints (4.8c) are active then $\gamma_{k,l} > 0$ $l_k^* > l$. Similarly, the constraints (4.8c) with $l_k^* < l$ and $s_{k,l} > 0$ are inactive then the dual variable $\beta_{k,l} < 0$ $l_k^* > l$ and the constraints (4.8d) are inactive then $\gamma_{k,l} = 0$ for $l_k^* < l$.

Algorithm 6 Greedy Algorithm for Solving Problems (4.9)

```
 1: Require: $\boldsymbol{s}^k, l_k^*$ obtained since the Algorithm , $Y_k = \sum_{j\in\mathcal{J}} \sum_{\substack{b_i\in\mathcal{B}j \\ b_i\ni k}} y_{i,j}$
 2: $\alpha_k = d_{k,l_k^*}$, $\beta_k = \gamma_k = 0$ for all $l \in \mathcal{L}_k$
 3: for $l \in \{1, \ldots, L_k\}$ do
 4:     if $l > l_k^*$ then
 5:         $\beta_{k,l} = d_{k,l} - d_{k,l_k^*}$
 6:     else if $l < l_k^*$ then
 7:         $\gamma_{k,l} = d_{k,l_k^*} - d_{k,l}$
 8:     end if
 9: end for
```

Theorem 4.3.1 Let $\tilde{\boldsymbol{z}}$ be the solution produced by Algorithm 5, and let $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ be the solution produced by Algorithm 6. Then,

- y is feasible for the primal problem (4.8);
- $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is feasible for the dual problem (4.9); and
- the two solutions are optimal for their respective problems.

Proof. We proceed in three stages. First, we show that $\boldsymbol{z}$ is primal feasible. Second, we show that $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ is dual feasible. Finally, we show that two solutions satisfy complementary slackness, which establishes that are optimal.

- **Primal feasibility**
  Note that constraints (4.8c) and (4.8d) are never violated in the algorithm. Since these constraints are never violated, the slackness associated with the inequality constraint are never negative. We only need to verify that constraint (4.8b) is satisfied, i.e, that the sum of the $\tilde{z}_{k,l}$ variables equals Y_k.

  Observe that we begin with $S_k = \sum_{l \in \mathcal{L}_k} s_{k,l} b_{k,l-1}$, then if $S_k = Y_k$ the algorithm terminates. If $S_k < Y_k$ we complete the rest. In decreasing order in $\mathcal{L}_k$ (so that $d_{k,L_k} > d_{k,L_k-1} > \cdots d_{k,1}$), we increase $\tilde{z}_{k,l}$ by $\min\{Y_k - S_k, b_{k,l}s_{k,l} - b_{k,l-1}s_{k,l}\}$ i.e., we increase $\tilde{z}_{k,l}$ so that either Y_k is reached or the maximum quantity available in interval l is reached. In case of the first option the algorithm terminates; in case of the second option, we continue with the following interval. Observe that when $s_{k,l} = 0$ this interval does not participate in the sum and can be omitted.
- **Dual feasibility**
  By definition to Algorithm 6, we have that

$$\alpha_k = d_{k,l^*} \text{ and } \beta_{k,l^*} = \gamma_{k,l^*} = 0$$

  The dual constraints are :

$$\alpha_k + \beta_{k,l} + \gamma_{k,l} \leq d_{k,l} \quad l \in \mathcal{L}_k, \tag{4.12a}$$
$$\beta_{k,l} \leq 0 \quad l \in \mathcal{L}_k, \tag{4.12b}$$
$$\gamma_{k,l} \geq 0 \quad l \in \mathcal{L}_k. \tag{4.12c}$$

  Following the initial definition, when Algorithm (6) terminates, we obtain $\gamma_{k,l} = 0$ and $\beta_{k,l} = d_{k,l} - d_{k,l^*} \leq 0$ for all $l \in \{1, \dots, l^*\}$. Analogously, we get $\beta_{k,l} = 0$ and $\gamma_{k,l} = d_{k,l} - d_{k,l^*} \geq 0$ for all $l \in \{l^*, \dots, L_k\}$. Notice that this definition ensures all constrains are active. Consequently, the dual is feasible.
- **Complementary slackness** The complementary slackness conditions are:

$$\alpha_k \cdot \left(\sum_{j \in \mathcal{J}} \sum_{\substack{b_i \in \mathcal{B}j \\ k \in b_i}} y_{i,j} - \sum_{l \in \mathcal{L}_k} \tilde{z}_{k,l} \right) = 0, \tag{4.13a}$$
$$\beta_{k,l} \cdot (b_{l-1}s_{k,l} - \tilde{z}_{k,l}) = 0, \tag{4.13b}$$
$$\gamma_{k,l} \cdot (b_l s_{k,} - \tilde{z}_{k,l}) = 0, \tag{4.13c}$$
$$\hat{z}_{k,l} \cdot (\alpha_k + \beta_{k,l} + \gamma_{k,l} - d_{k,l}) = 0 \tag{4.13d}$$

Equation (4.13a) is satisfied due to (4.8b). Similarly, the solution (α, β, γ) obtained from Algorithm (6) satisfies by construction constraints (4.9b) with equality, implying that equation (4.13d) is satisfied. In addition, this solution satisfies equations (4.13b) and (4.13c) since $\gamma_{k,l} = 0$ for all $l \in \{1, \ldots, l^* - 1\}$ and $b_{k,l}s_{k,l} = \tilde{z}_{k,l}$ otherwise, and $\beta_{k,l} = 0$ if $l \in \{l^*, \ldots, L_k\}$ with $b_{k,l-1}s_{k,l} = \tilde{z}_{k,l}$ otherwise. Then, since complementary slackness holds, the primal and dual solutions are optimal.

□

We note that the solution method works for a combinatorial coalition formation problem with capacity constraints on products. We can add the capacity constraints to the models described above in two ways: indirectly and directly. First, we can modify the price function, adding a value large enough for quantities greater than the product's capacity. Another alternative is to add the a capacity constraint to the master problem as the following inequality:

$$\sum_{j \in \mathcal{J}} \sum_{\substack{b_i \in \mathcal{B}j \\ b_i \in k}} y_{i,j} \leq cap_k \;\; k \in \mathcal{K}, \tag{cap-SPF}$$

where cap_k is the capacity available of product k, $k \in \mathcal{K}$.

4.4 Computational experiments

4.4. COMPUTATIONAL EXPERIMENTS

In this Section, we present the computational tests performed. All procedures and algorithms have been written using the language Python and, for MIP problems, we used GUROBI version 9.1. In Section 4.4.1 we describe the synthetic data that was used in our experiments, and Section 4.4.2, we present the results of our experiments.

4.4.1 Instances: description

We now describe the generation of instances used for our computational experiment. Due to the fact that SPF supports only a step price function, we compare the performance of both GPF as SPF for this type of price function. Therefore, we describe below the reservation price, the function price, and finally, the bundle formation considered. The instances considered are based on the experiment set used in [89] for Combinatorial Coalition Formation.

Reservation prices: the reservation price depends on the size of the bundle, rather than on which products are contained. We establish the willingness to pay for the bundle b_i of purchaser j as $r_{i,j} \in [\underline{r} \cdot k^{\alpha}, \overline{r} \cdot k^{\alpha}]$ where $\underline{r}$ and $\overline{r}$ are the lower and upper bounds of the reservation price for a single item, k is the number of items in the bundle, and $\alpha > 0$ indicate if the products in a bundle are substitutes or complements. In this manner, $\alpha = 1$ represents that the customer is indifferent to buying a bundle or buying the items in the bundle separately. On the other hand, $\alpha > 1$ represents when the buyers prefer to buy the items as a bundle to a subset of them, i.e., the products are complementary. In contrast, when $\alpha < 1$ the reservation price of purchasing the items as a bundle is less than buying a subset of them, i.e., the items are substitutes.

Price-quantity curve: describes the price drop between p_h to p_l as a function of increase in quantity. We assume the price drops from p_h to p_l regularly, i.e., each decreasing step is coming after the same increase in the order quantity and resulting in equal price reductions. We denote the *Number of steps* by $R_k \in [\underline{R}, \overline{R}]$ where $\underline{R}$ and $\overline{R}$ are the minimum and maximum number of steps respectively. The Price Decrease Rate (PDR) is defined as the ratio of price reduction to the quantitative increase over each step. We defined $PDR \in [\underline{PDR}, \overline{PDR}]$ are the minimum and maximum steps respectively. The PDR determines how steep is the quantity discount, i.e., with a higher PDR, more buyers can purchase in the buying group. We note that with more steps on the price curve, the buying group's formation is easier as even a small increase in the sold quantity can drive down the price.

Bundle formation: we randomly choose $|\mathcal{B}|$ sets from the full list of possible permutations with a maximum length Q of the $|\mathcal{K}|$ products with $Q < |\mathcal{K}|$. In previous experiments, we observed that instances with overlaps between bundles are harder to solve. Hence, in order the exploit possible overlap between bundles, we set up the bundle size to six and assume that there exist two essential products that are mutual substitutes. Specifically, given these two products, one of them is necessarily selected randomly in each bundle. For instance, we suppose a set $\mathcal{K} = \{k_1, k_2, k_3, k_4, k_5\}$ where k_1 and k_2 are essential. If the bundle size is three, then the possible bundles of the set $\mathcal{B}$ are $\{k_1, k_3, k_4\}$, $\{k_1, k_3, k_5\}$, $\{k_1, k_4, k_5\}$, $\{k_2, k_3, k_4\}$, $\{k_2, k_3, k_5\}$ and $\{k_2, k_4, k_5\}$.

For our experiments we used the following values $\underline{r} = 80, \overline{r} = 130$, $|\mathcal{J}| = \{100, 150, 200, 300, 500, 1000\}$; $|\mathcal{B}| = \{15, 31, 62, 93, 124, 155, 207\}$; $p_l = 70$, $p_h = 100$; $|\mathcal{K}| = 15$; $Q = 5$ as maximum number of products for bundles with $\alpha = \{0.5, 0.55, \ldots 1.25, 1.30\}$ and $\underline{R} = 4$ and $\overline{R} = 10$. We generated 100 random instances for each set of values of the parameters described above. The headers of the Tables have the following meanings:

- Instance: Name of instance;
- GAP: average gap $100\frac{UB-LB}{LB}\%$, where UB(LB) is the best upper(lower) bound;
- *RGAP*: average integrality gap $100\frac{LP-LB}{LB}\%$, with LP = linear relaxation value;
- Sol: number of instances that reached optimal solution during the maximum time of experiment;
- $t(s)$: average CPU time in seconds (Total time).

4.4.2 Comparison between the different formulations

Table 4.5 shows the computational performance of the different formulations. As expected, the computational performance of formulation (GPF) is worse than (SPF1) and (SPF2). Although (SPF1) and (SPF2) reformulation has the same linear relaxation, the (SPF2) obtained a better GAP than (SPF1), and additionally, no higher computational cost is perceived. At the same time, (SPF2) presented the best value in the majority of all indexes studied of the three.

On the other side, note that the number of constraints and variables generated for (GPF) is higher than in (SPF1) and (SPF2), which may impact the algorithms' processing time. However (GPF) can be used for any function price even if it is nonincreasing; hence it is useful when buyers are few. Despite (SPF) being a model to represent step price functions, since purchase quantities are integer, (SPF) can represent any non-increasing function by considering unit length intervals.

Table 4.5: Comparison of GAP, *RGAP* and t(s) between the formulations GPF and SPF. $|\mathcal{K}| = 12$

	SP1				SPF2				GPF					SP1				SPF2				GPF			
Instance	GAP	*RGAP*	t(s)	Sol	GAP	*RGAP*	t(s)	Sol	GAP	*RGAP*	t(s)	Sol	Instance	GAP	*RGAP*	t(s)	Sol	GAP	*RGAP*	t(s)	Sol	GAP	*RGAP*	t(s)	Sol
100-62	0.337	13.524	1,949	82	0.016	13.516	1,412	98	7.345	14.435	3,600	-	300-62	0.052	0.998	553	91	0.020	0.995	291	98	0.565	1.023	1,962	53
100-93	0.261	9.627	1,966	78	0.019	9.621	1,151	99	9.224	11.157	3,600	-	300-93	9.346	12.191	3,563	4	8.790	12.051	3,556	4	11.527	12.169	3,600	-
100-124	0.254	8.768	2,294	79	0.066	8.756	1,690	95	8.598	10.109	3,600	-	300-124	8.345	10.928	3,588	2	10.171	12.256	3,536	7	11.066	11.422	3,600	-
100-155	0.214	7.842	2,476	85	0.020	7.835	1,867	99	7.757	9.005	3,600	-	300-155	10.002	11.782	3,604	-	11.247	12.918	3,580	4	11.036	11.318	3,600	-
100-207	0.693	6.769	3,150	63	0.021	6.740	2,331	99	6.665	7.701	3,600	-	300-207	10.874	11.950	3,585	2	9.523	11.828	3,419	21	10.377	10.568	3,601	-
150-62	0.579	8.977	2,444	56	0.116	8.949	1,539	91	6.211	9.363	3,600	-	500-62	0.000	0.002	1	100	0.000	0.002	1	100	0.000	0.002	3	100
150-93	1.130	9.586	2,907	53	0.119	9.538	2,059	95	10.779	11.929	3,600	-	500-93	13.037	15.074	3,582	2	13.178	14.606	3,585	1	12.471	12.838	3,601	-
150-124	2.829	8.917	3,262	32	1.076	8.732	2,817	64	9.022	10.162	3,600	-	500-124	15.037	15.686	3,603	-	13.570	14.437	3,602	-	11.252	11.514	3,602	-
150-155	3.447	7.816	3,349	20	1.516	7.704	3,002	56	8.177	9.038	3,600	-	500-155	14.436	15.136	3,603	-	14.389	15.261	3,585	2	10.964	11.151	3,655	-
150-207	4.177	7.382	3,374	14	3.743	7.419	3,064	31	8.607	9.184	3,600	-	500-207	14.512	15.162	3,595	1	14.085	14.977	3,571	3	10.209	10.343	3,691	-
200-62	0.210	5.466	1,887	67	0.064	5.463	920	93	3.361	5.713	3,486	4	1000-62	0.000	0.000	1	100	0.000	0.000	1	100	0.000	0.000	3	100
200-93	5.230	10.336	3,387	14	2.815	9.934	3,130	39	10.370	11.415	3,600	-	1000-93	13.346	14.907	3,581	2	12.495	14.258	3,582	2	13.336	13.633	3,602	-
200-124	5.234	9.228	3,499	10	5.347	9.732	3,232	27	9.588	10.277	3,600	-	1000-124	16.404	16.706	3,608	-	16.206	16.506	3,603	-	12.969	13.190	3,602	-
200-155	5.480	8.722	3,516	9	5.507	8.872	3,444	14	9.182	9.652	3,600	-	1000-155	16.287	16.528	3,607	-	17.078	17.321	3,603	-	13.449	13.626	3,600	-
200-207	5.485	7.866	3,562	5	5.191	8.282	3,310	23	9.094	9.391	3,600	-	1000-207	16.296	16.461	3,604	-	19.507	19.668	3,602	-	11.300	11.423	3,600	-

4.4.3 Evaluation of Bender's decomposition

We studied three different ways to add the Benders cut into the *Branch and Bound* (B&B) tree: the traditional version of *Branch and Cut* (B&C) where it is possible to add the cuts in all nodes; the reduced version of *Branch and Cut* (B&C2) where cuts are added only in the incumbent nodes (nodes with integer solution), and *Cut and Branch* (C&B) where the cuts are applied only in the root node, i.e., Benders decomposition is used only in the linear relaxation. Cuts are added as lazy constraints to the subproblems generated in each tree node of the B&B method, using the callback environment provided for GUROBI. All GUROBI default cuts were turned off to prove the efficiency of the Benders cuts. The results of the computational experiment are shown in Table 4.6. It is important to note that the methods we study incorporate the stabilization scheme described in section **??**. To construct the feasible solution of the stabilization scheme, we consider, given a fractional solution, that each customer selects the bundle with a fractional value closest to one. We note that the *C&B* obtained the best performance in most cases reducing the GAP, the t(s), and solved a greater number of instances optimally than the alternative methods.

Table 4.6: Comparison of GAP and t(s) between of Benders Decomposition approach for SPF. $|\mathcal{K}| = 12$

	RSPF-CB			RSPF-BD2			RSPF-BD				RSPF-CB			RSPF-BD2			RSPF-BD		
Instance	GAP	t(s)	Sol	GAP	t(s)	Sol	GAP	t(s)	Sol	Instance	GAP	t(s)	Sol	GAP	t(s)	Sol	GAP	t(s)	Sol
100-62	0.005	177	100	0.010	455	99	0.008	446	99	300-62	0.002	17	100	0.000	34	100	0.000	33	100
100-93	0.006	280	100	0.000	565	100	0.000	550	100	300-93	0.077	1,210	96	0.451	2,079	78	0.431	2,021	80
100-124	0.007	439	100	0.028	946	98	0.022	900	98	300-124	0.136	1,674	94	0.910	2,685	59	0.828	2,665	60
100-155	0.006	500	100	0.032	1,046	98	0.022	959	99	300-155	0.274	1,867	86	0.960	2,801	57	0.937	2,754	57
100-207	0.006	669	100	0.065	1,349	96	0.063	1,329	96	300-207	0.671	2,335	70	1.608	2,991	36	1.614	2,991	35
150-62	0.006	248	100	0.053	558	96	0.052	551	96	500-62	0.001	6	100	0.000	5	100	0.000	4	100
150-93	0.006	545	100	0.075	1,069	96	0.077	1,022	96	500-93	0.553	2,092	80	1.424	3,013	46	1.341	2,972	49
150-124	0.006	715	100	0.112	1,449	92	0.103	1,410	94	500-124	0.994	2,616	61	1.910	3,177	33	1.763	3,139	37
150-155	0.006	770	100	0.081	1,486	95	0.089	1,449	95	500-155	1.532	3,050	46	2.626	3,442	14	2.514	3,423	15
150-207	0.021	1,140	99	0.332	1,955	81	0.306	1,903	83	500-207	2.100	3,184	29	2.920	3,499	11	2.935	3,488	10
200-62	0.004	54	100	0.000	118	100	0.000	115	100	1000-62	0.000	17	100	0.000	7	100	0.000	7	100
200-93	0.007	865	100	0.158	1,621	90	0.164	1,573	91	1000-93	2.852	3,298	23	4.127	3,547	7	4.079	3,540	7
200-124	0.056	1,088	97	0.364	1,917	83	0.351	1,899	84	1000-124	3.138	3,432	16	4.177	3,565	4	4.017	3,569	3
200-155	0.081	1,332	95	0.543	2,281	70	0.511	2,239	73	1000-155	3.684	3,478	9	4.213	3,556	4	4.134	3,550	4
200-207	0.206	1,611	91	0.704	2,435	66	0.700	2,437	66	1000-207	3.381	3,492	9	4.339	3,555	5	4.281	3,560	5

The analysis shows that there does not exist a clear difference of yield between B&C and B&C2. We can explain the results considering the number of possible cuts that can generate each method. Indeed, we remark that the benders cuts generated are equivalent to strengthened constraint (4.3e-1). Hence, the possible cuts that we can add are $|\mathcal{K}| \times L_k$. Therefore, once the cuts are added in the root node, the number of additional cuts that can be generated is minimal.

Figure **??** shows the mean gap and the percentage of optimally resolved instances by instance size ($|\mathcal{J}|, |\mathcal{B}|$) obtained for B&C and C&B. Note that instances up to 1000 clients have been resolved. We can observe that the higher the number of clients and bundles, the more complex the instances are, with the increase in bundles complicating the problem more than the increase in buyers. Still, both the B&C and C&B methods optimally solve instances within1000 customers and 62 bundles. Moreover, at most instance sizes, C&B can solve some percentage (except with 1000 customers and more than 155 bundles). Furthermore, C&B obtain the best gap in all instances, with a mean gap less to 6% for instances within 1000 purchases.

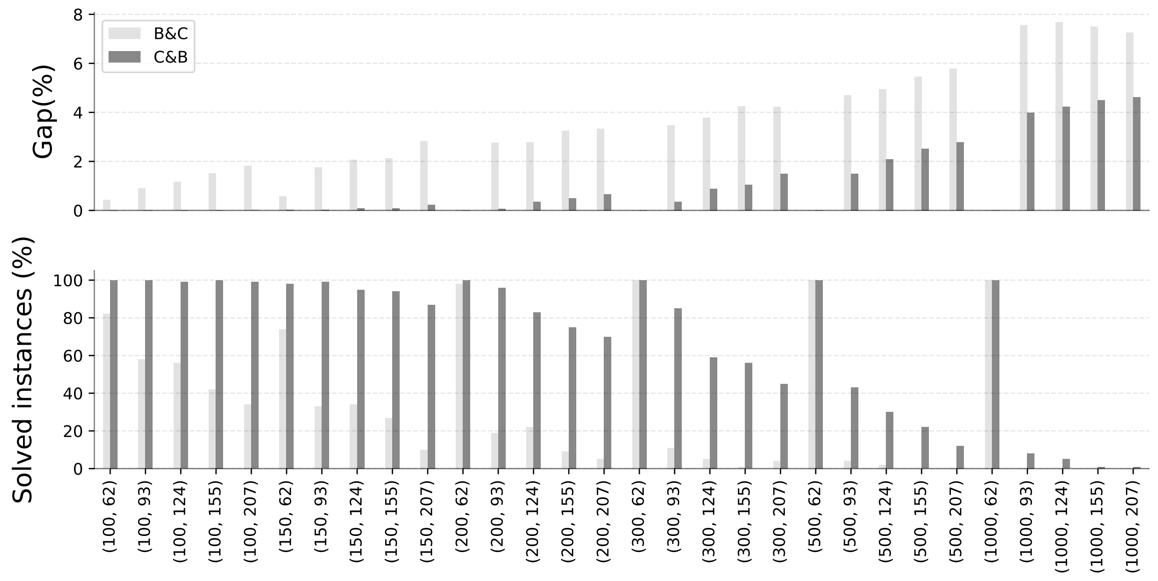

Figure 4.2: Barplot upper represents the mean gap of each method by size, and lower barplot represents the percentage of solved instances by size.

In addition, we study the effect of incorporating a stabilization scheme and initial solution on the performance of the *C&B* and *B&C* methods. We propose a simple warm start where each customer chooses the bundle with the best reserve price. Subsequently, we prove a local change by removing customers that can improve the solution. In the first part, let us see how including stabilization affects these methods. The results obtained in Figure 4.3 show that the use of stabilization in both methods does not reduce the number of nodes explored. However, Figure 4.4 shows that more solutions are solved optimally in the set study time (one hour) when stabilization is included. At the same time, Figure 4.5 shows that when one adds a stabilization scheme, the methods obtained a higher percentage of instances solved close of optimal and for the remaining instances are reduced the gap. Therefore, for this case, although including stabilization does not apparently decrease the number of nodes explored, it allows us to go deeper into more efficient nodes that help us to improve better solutions in a shorter time.

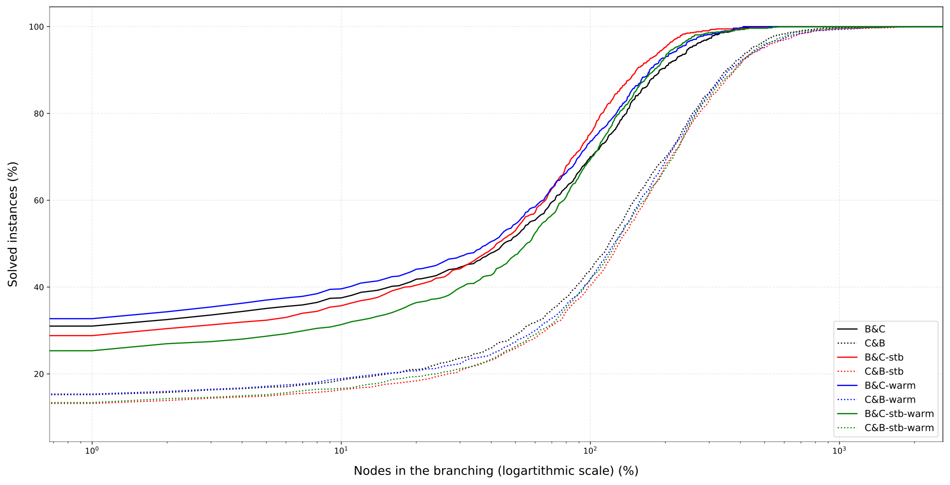

Figure 4.3: The percentage of solved instances with a number nodes less or equal to that corresponding abscissas is represented for each methods.

On the other hand, we observe that for both cases, if one just gives it an initial solution without doing stabilization, the methods worsen its overall performance. Meanwhile, when we include a warm start to the stabilization scheme improves the performance of $B\&C$. In contrast, for $C\&B$, the opposite is true, i.e., it is not useful to combine with a warm start.

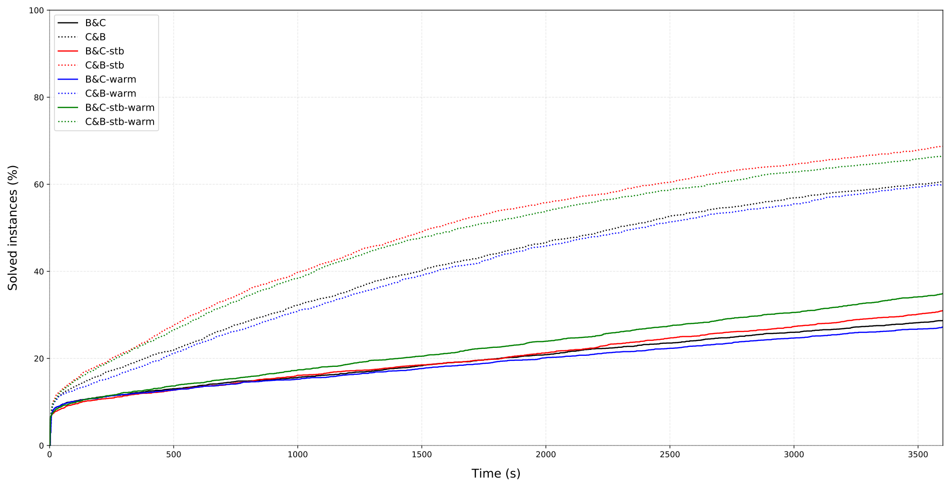

Figure 4.4: The percentage of solved instances in an time less or equal to that corresponding abscissas is represented for each methods.

Finally, we can deduce that results described a clear advantage in the yield obtained of $C\&B$ over $B\&C$. In Figure 4.3 it is shown that $C\&B$ must explore a higher number of nodes for solving the MILP. Furthermore, in Figure 4.4 show that the percentage of instances solved before one hour is double for $C\&B$ than $B\&B$ (over 60% for the case of $C\&B$ and less than 30% for $B\&C$). In the same sense, Figure 4.5 determines that 80% of the instances solved with $C\&B$ obtain a gap of less than 30% within one hour. While that for $B\&C$ only a 40% of instances reach the same percentage of the gap.

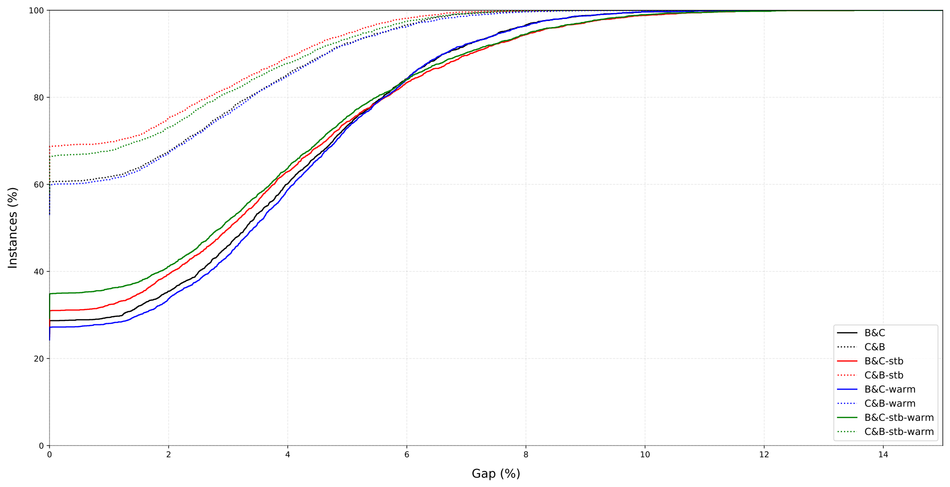

Figure 4.5: The percentage of instances with a gap less or equal to that corresponding abscissas is represented for each methods.

4.5 Conclusions

4.5. CONCLUSIONS

Nowadays, the constant expansion of the internet and the number of users worldwide have boosted the digital marketplace's growth, which invites new forms of interaction between purchasers and all types of merchants. For example, sellers can offer wholesale products at a lower price, but not necessarily the buyers wish to or can purchase a large number of items. Thus, the purchasers with a similar preference can be grouped to buy together to take advantage of a wholesale price. In these circumstances, a coordinator can divide a set of buyers with heterogeneous preferences interested in obtaining a better price, according to each member's reservation price. This problem is known as the coalition formation problem in related literature, which seeks to divide an set of agents into a disjoint subset to maximize the total reward.

In this scenario, we used a *Combinatorial Coalition Formation* problem, where the products are offered in bundles, and each customer is willing to buy at most one bundle. Thus, every purchaser has a reservation price for any of them, which –at the same time- depends on whether they are complementary or substitute. This decision problem may be simplified to a Set Packing Problem which is a NP-hard problem. As an additional complication, the pricing function depends on the size of the purchase. However, the coalition applies to each bundle because the buyers purchase bundles, causing difficulties in establishing a clear definition of the value in the coalition structure.

From of point of view of sellers, wholesale pricing helps reduce the inventory cost, involves savings in marketing cost, and preserves or increases their market power. Additionally, in the bundle, the sellers can reduce the inventory of items difficult to sell by selling them in conjunction with higher turnover products.

In this chapter, we presented mixed-integer optimization models for the Coalition Formation Problem. We showed a general price function (GPF) formulation and a step price function (SPF). The first formulation uses a binary variable for every purchase size, discretizing the cost function, obtaining a value for the linear coalition. Due to the step price function widely used to model price wholesale, we introduced a multi-choice model for SPF. Given its structure, it is possible to apply a Benders decomposition approach.

The computer experience shows that the (SPF) has a better performance than the (GPF) for the step formulation. This occurs since the number of variables generated in an (SPF) is lower than in (GPF). An interesting theoretical outcome suggests that the cut produced by the Benders Decomposition approach becomes equal to a valid inequality proposed. Additionally, the experiments revealed a higher performance from the Cut and Branch than from the Branch and Cut procedures. This phenomenon arises from the fact that the cuts generated at the root nodes are not tight enough, producing growth in the size of the problem solved in each node.

Chapter 5

Conclusion

The marketplace is a space for bargaining among sellers and buyers. On one side, sellers must decide pricing, assortment, markdown, location, store, and distribution. On the other side, the buyers must decide what, when, how much, and where to purchase. Depending on the time and gain or loss required, these can be tactical, operative, or strategic decisions. Given the complexity of many of these decisions, it is desirable to develop tools to catch the main attributes and help make better choices.

Consequently, mathematical programming has developed diverse models that simulate agents' behavior, allowing decision makers to study and interpret outcomes, which are then used to inform decisions. The more agents and decisions are included in these mathematical programming models, the more complex and challenging to solve these models become. Hence, it is crucial to explore more sophisticated models and efficient algorithms to solve them in a reasonable time. Along this line, decomposition methods attempt to split the problem's formulation into simpler models that are embedded inside an efficient solution scheme. For instance, Lagrangian relaxation, cut generation, and bender decomposition have shown advantages in solving problems, particularly in mixed-integer programming.

This thesis aims to address two essential sub-fields in the marketplace and e-marketplace through assortment problem and coalition formation problem, respectively, from the point of view of mathematical programming models and algorithmic developments for their efficient solution. The first problem focuses on traditional marketplaces where companies face limited physical space in the stores to exhibit line products, especially in slower-moving consumer goods, e.g., TVs, washing machines, refrigerators. Therefore, knowing what items to exhibit would reduce the risk of a slow turnover. Futhermore, considering buyers' purchase decisions is essential for an efficient location and pricing decision. Nowadays, with market studies, it is possible to quantify variables that influence the buyers' decision, such as willingness to pay and travel costs to acquire the products. With that, it is possible to simulate the buyers' behavior when facing different assortments in each store, establishing the firm's best decision.

On the other hand, in many cases, companies offer discounts for volume purchases; however, customers may not necessarily be able to access them, either because they do not have the economic capacity or need fewer units. Hence, one way to access this benefit is taking e-marketplaces to buy together in groups or coalitions of customers with similar preferences. Consequently, the second problem aims to establish coalitions of buyers to take advantage of wholesale prices. In this case, it is considered that customers want to buy a bundle of products, where the price of each product depends on the number of people who buy it. Therefore, the problem consists of selecting the best structure of customer coalitions so as to maximize social welfare.

5.1 Contributions

Contributions for PLOMS

- Model for PLOMS: the traditional models for the Line Product Design problem has considered the multiple store and geographical dimension. For the first case, earlier works suggest solving a model in every store or repeat the same assortment in all of them. For the second case, notice that the customer could have different preferences among areas. Hence, they can travel further, assuming a travel cost, to acquire a lower-price product or when a product is unavailable at the closest store. This work proposed a novel model to Line Product Design with multiple stores and the travel cost of buyers. We proposed a bi-level model to represent this behavior, where the leader is a firm and followers are the customers. Given that a customer purchases at most one product and knows its utility, it is possible to define an order of the preferences in the choice pair product/location. This determines the set of better (and worse) alternatives to a given product/location; consequently, the second level problem can be transformed into a set of constraints that allow it to collapse in a single-level optimization problem. Besides, the PLOMS can consider pricing decisions in each store by repeating the items with discrete prices. Notice that given that customers are rational, the products available would only have one price in each store. At the same time, the formulation can consider that a bundle of products are purchased in only one store. We can consider bundles like a product, hence obtaining a set of preferences and rescuing the single-level formulation.
- Algorithm for PLOMS: this work provided three different decomposition methods to solve the PLOMS: (i) Lagrangian relaxation, (ii) the Bender decomposition, and (iii) cuts generation. For (i) the Lagrangian methods we considered three different relaxation approaches, the first relaxes the worst preference constraint, the second the worst preference constraint and the linking constraint, and the third method the worst preference constraint and the limit purchase constraint. The last two relaxations split the problem into efficient sub-problems as they satisfy the integrality property, while the sub-problem of the first relaxation is a Single Location Problem that is hard to solve. The (ii) Bender decomposition, which has been used to solve the version single-store, is studied to extend this efficient solution method to multiple stores. The (iii) cuts generation, since PLOMS is similar to FCPL, then we adapt its family of valid inequalities from FCPL to PLOMS. For the latter case, we tested adding valid inequalities using B&C and C&B. To validate the results, we used the bender decomposition as a benchmark. Computational experiments showed that C&B has better results, improving the current results.

Therefore, there are three main contribution for the PLOMS model: extend the model to multiple stores including the travel cost in customer decision, a family of valid inequalities for this problem, and improving solution methods with a C&B approach that improves on current practice.

Contributions for CCF

- Model for CCF: (i) general price function; the model assumes a non-increasing function that can model different discount strategies depending on the number of buyers. The formulation is a compact integer programming in which the number of variables increases with the number of buyers and are difficult to solve. We provide a strengthened linear relaxation; (ii) function ladder price, the model considers a function ladder price, a common profile price that uses the wholesale. The formulation is mixed-integer programming that uses a discrete variable to active the echelon belongs to the number of buyers. The model admits a decomposition method that split the continuous and discrete variables. Besides, we include strengthening to improve linear relaxation.
- Algorithm for CCF: the mixed-integer programming to ladder price function allows a Benders decomposition approach to split the integer and continuous variables. By adding a simple constraint, only optimality cuts are necessary in Bender's decomposition. The subproblem can be solved analytically. The computational experiments show that the approach can obtain optimal solutions for instances with up to two hundred buyers.

5.2 Future work

For the Multiple Location Product Problem, a strong assumption is that the buyers acquire at most one product. Although for the slower moving consumer goods, e.g., TV or washing machine, it is reasonable; often the customer wishes to purchase a bundle of items either in a single store or various stores. Furthermore, the products can be complementary or substitute depending on the items, where business decisions will probably be different. New models and solution methods should be investigated to incorporate these realistic aspects to the problem. Being able to include bundle purchases is of interest both from the mathematical point of view and from practice.

This study incorporated only one firm that sells; hence, adding competition is another relevant problem. In a first approach, it is possible to consider a leader firm and another follower. In this case, the models are a three-level formulation; leader, follower, and consumer. Assuming the model provided in this thesis, it is possible to collapse the consumer choice model into the follower optimization problem leading to a bilevel modelof the two agents. In order to solve this, it is necessary to apply methods in bilevel programming, as presented in Chapter 2.

Besides, including temporal issues can approximate further the model to the real world. The sellers must decide where and when to allocate the products, pricing considering the inventory cost and obsolescence in a time window. In contrast, the buyers can decide to wait to find a better price or to wait until the product becomes available in the closest store, given his willingness to pay and patience. Additionally, the buyers could buy under a certain probability; therefore, each agent will maximize their excepted utility.

For the Combinatorial Coalition Formation Problem, this thesis covered the buyers' side. Thus, it would be interesting to investigate this problem from the seller's perspective when faced group of buyers that can form coalitions. This problem can be modeled as a Stackelberg game in which the firm is the leader, and the customers are the followers; therefore, the firm proposes a price function depending on the size of the purchase; subsequently, with this information, the buyer groups purchase a particular bundle. Furthermore, the products that are bundled together in a bundle could also be part of the decision as another extension of the model.

Besides, it is possible to incorporate decisions depending on time, such as the buyers' arrival and the deadline for coalition formation where the fairness and stability solutions are different. The outcomes will change whether consider the choice online, which depended on how to arrive at the buyers, or offline, where all the information is revealed beforehand. Comparing both approaches is an attractive investigation for the future.

Bibliography

[1] E. Aiyoshi and K. Shimizu. Hierarchical decentralized system and its new solution by a barrier method. In *Proceedings-International Conference on Cybernetics and Society*, pages 667–671, 1980.

[2] E. Aiyoshi and K. Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29(12):1111–1114, 1984.

[3] F. A. Al-Khayyal, R. Horst, and P. M. Pardalos. Global optimization of concave functions subject to quadratic constraints: an application in nonlinear bilevel programming. *Annals of Operations Research*, 34(1):125–147, 1992.

[4] H. Alibeiki, S. Li, and R. Vaidyanathan. Market dominance or product cost advantage: Retail power impacts on assortment decisions. *International Journal of Production Economics*, 222:107505, 2020.

[5] N. Aras and H. Küçükaydın. Bilevel models on the competitive facility location problem. In *Spatial Interaction Models*, pages 1–19. Springer, 2017.

[6] A. Balakrishnan and S. C. Graves. A composite algorithm for a concave-cost network flow problem. *Networks*, 19(2):175–202, 1989.

[7] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000.

[8] F. Barahona and F. A. Chudak. Solving large scale uncapacitated facility location problems. In *Approximation and complexity in numerical optimization*, pages 48–62. Springer, 2000.

[9] J. F. Bard and J. T. Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292, 1990.

[10] J. F. Bard, J. Plummer, and J. C. Sourie. A bilevel programming approach to determining tax credits for biofuel production. *European Journal of Operational Research*, 120(1): 30–46, 2000.

[11] J. E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

[12] A. Belloni, R. Freund, M. Selove, and D. Simester. Optimizing product line designs: Efficient methods and comparisons. *Management Science*, 54(9):1544–1552, 2008.

[13] P. Belotti, F. Malucelli, and L. Brunetta. Multicommodity network design with discrete node costs. *Networks: An International Journal*, 49(1):90–99, 2007.

[14] D. Bertsimas and V. V. Mišić. Exact first-choice product line optimization. *Operations Research*, 2019.

[15] O. Besbes and D. Sauré. Product assortment and price competition under multinomial logit demand. *Production and Operations Management*, 25(1):114–127, 2016.

[16] A. Bhatnagar and S. S. Syam. Allocating a hybrid retailer's assortment across retail stores: Bricks-and-mortar vs online. *Journal of Business Research*, 67(6):1293–1302, 2014.

[17] W. F. Bialas and M. H. Karwan. Two-level linear programming. *Management science*, 30(8):1004–1020, 1984.

[18] F. Bistaffa, A. Farinelli, and S. Ramchurn. Sharing rides with friends: A coalition formation algorithm for ridesharing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[19] F. Bistaffa, A. Farinelli, G. Chalkiadakis, and S. D. Ramchurn. A cooperative game-theoretic approach to the social ridesharing problem. *Artificial Intelligence*, 246:86–117, 2017. doi: 10.1016/j.artint.2017.02.004.

[20] E. Y. Bitar, E. Baeyens, P. P. Khargonekar, K. Poolla, and P. Varaiya. Optimal sharing of quantity risk for a coalition of wind power producers facing nodal prices. In *2012 American Control Conference (ACC)*, pages 4438–4445. IEEE, 2012.

[21] L. Boongasame, H.-f. Leung, V. Boonjing, and D. K. Chiu. Forming buyer coalitions with bundles of items. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 714–723. Springer, 2009.

[22] L. Boongasame, P. Temdee, and F. Daneshgar. Forming buyer coalition scheme with connection of a coalition leader. *Journal of theoretical and applied electronic commerce research*, 7(1):111–122, 2012. doi: 10.4067/S0718-18762012000100009.

[23] D. Braess. Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung*, 12 (1):258–268, 1968.

[24] P. Briest and P. Krysta. Buying cheap is expensive: Approximability of combinatorial pricing problems. *SIAM Journal on Computing*, 40(6):1554–1586, 2011.

[25] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. Joint design and pricing on a network. *Operations research*, 56(5):1104–1115, 2008.

[26] L. Brotcorne, P. Marcotte, and G. Savard. Bilevel programming: The montreal school.

INFOR: Information Systems and Operational Research, 46(4):231–246, 2008.

[27] L. Brotcorne, B. Fortz, and M. Labbé. Special issue on bilevel optimization, 2020.

[28] H. I. Calvete, C. Domínguez, C. Galé, M. Labbé, and A. Marin. The rank pricing problem: models and branch-and-cut algorithms. *Computers & Operations Research*, 105:12–31, 2019.

[29] W. Candler and R. Townsley. A linear two-level programming problem. *Computers & Operations Research*, 9(1):59–76, 1982.

[30] L. CáNovas, S. GarcíA, M. Labbé, and A. MaríN. A strengthened formulation for the simple plant location problem with order. *Operations Research Letters*, 35(2):141–150, 2007.

[31] M. Caramia and R. Mari. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters*, 10(5):997–1019, 2016.

[32] M. Chen, Z.-L. Chen, G. Pundoor, S. Acharya, and J. Yi. Markdown optimization at multiple stores. *IIE Transactions*, 47(1):84–108, 2015.

[33] M.-C. Chen and C.-P. Lin. A data mining approach to product assortment and shelf space allocation. *Expert Systems with Applications*, 32(4):976–986, 2007.

[34] Y. Chen, M. Florian, and S. Wu. A descent dual approach for linear bilevel program. *CENTRE DE RECHERCHE SUR LES TRANSPORTS PUBLICATION*, (866), 1992.

[35] R. Chicoisne and F. Ordóñez. Risk averse stackelberg security games with quantal response. In *International Conference on Decision and Game Theory for Security*, pages 83–100. Springer, 2016.

[36] T. R. Christensen and M. Labbé. A branch-cut-and-price algorithm for the piecewise linear transportation problem. *European journal of operational research*, 245(3):645–655, 2015.

[37] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[38] V. Conitzer and T. Sandholm. Complexity of determining nonemptiness of the core. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 230–231, 2003.

[39] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90, 2006.

[40] K. L. Croxton, B. Gendron, and T. L. Magnanti. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science*, 49(9):1268–1273, 2003.

[41] I. R. de Farias Jr, M. Zhao, and H. Zhao. A special ordered set approach for optimizing a discontinuous separable piecewise linear function. *Operations Research Letters*, 36(2): 234–238, 2008.

[42] G. Dobson and S. Kalish. Heuristics for pricing and positioning a product-line using conjoint and cost data. *Management Science*, 39(2):160–175, 1993.

[43] T. A. Edmunds and J. F. Bard. Algorithms for nonlinear bilevel mathematical programs. *IEEE transactions on Systems, Man, and Cybernetics*, 21(1):83–89, 1991.

[44] H. A. Eiselt and V. Marianov. *Foundations of location analysis*, volume 155. Springer Science & Business Media, 2011.

[45] H. A. Eiselt, V. Marianov, and T. Drezner. Competitive location models. In *Location science*, pages 365–398. Springer, 2015.

[46] H. A. Eiselt, V. Marianov, and T. Drezner. Competitive location models. In *Location science*, pages 391–429. Springer, 2019.

[47] Y. Fang, L. Sun, and Y. Gao. Bundle-pricing decision model for multiple products. *Procedia computer science*, 112:2147–2154, 2017.

[48] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2017.

[49] T. Flamand, A. Ghoniem, M. Haouari, and B. Maddah. Integrated assortment planning and store-wide shelf space allocation: An optimization-based approach. *Omega*, 2017.

[50] P. Fontaine and S. Minner. Benders decomposition for discrete–continuous linear bilevel problems with application to traffic network design. *Transportation Research Part B: Methodological*, 70:163–172, 2014.

[51] J. Fortuny-Amat and B. McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the operational Research Society*, 32(9): 783–792, 1981.

[52] A. Frangioni and B. Gendron. Piecewise linear cost network design. In *Network Design with Applications to Transportation and Logistics*, pages 167–185. Springer, 2021.

[53] D. Gao, W. Jiao, and J. Zhang. Capacitated facility location problem with freight cost discount. In *2010 7th International Conference on Service Systems and Service Management*, pages 1–5. IEEE, 2010.

[54] J. Gao. On dynamic pricing and assortment personalization in strategic settings. 2017.

[55] B. Gendron, P.-V. Khuong, and F. Semet. A lagrangian-based branch-and-bound algorithm for the two-level uncapacitated facility location problem with single-assignment constraints. *Transportation Science*, 50(4):1286–1299, 2016.

[56] A. M. Geoffrion. Lagrangean relaxation for integer programming. In *Approaches to integer programming*, pages 82–114. Springer, 1974.

[57] A. Ghoniem and B. Maddah. Integrated retail decisions with multiple selling periods and customer segments: optimization and insights. *Omega*, 55:38–52, 2015.

[58] A. Ghoniem, B. Maddah, and A. Ibrahim. Optimizing assortment and pricing of multiple retail categories with cross-selling. *Journal of Global Optimization*, 66(2):291–309, 2016.

[59] D. B. Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, 4:47–85, 1959.

[60] G. Greco and A. Guzzo. Constrained coalition formation on valuation structures: Formal framework, applications, and islands of tractability. *Artificial Intelligence*, 249:19–46, 2017. doi: 10.1016/j.artint.2017.04.005.

[61] P. E. Green and A. M. Krieger. Models and heuristics for product line selection. *Marketing Science*, 4(1):1–19, 1985.

[62] P. E. Green and A. M. Krieger. Recent contributions to optimal product positioning and buyer segmentation. *European Journal of Operational Research*, 41(2):127–141, 1989.

[63] B. Guta. Subgradient optimization methods in integer programming with an application to a radiation therapy problem. 2003.

[64] P. Hanjoul and D. Peeters. A facility location problem with clients' preference orderings. *Regional Science and Urban Economics*, 17(3):451–473, 1987.

[65] P. Hansen, Y. Kochetov, and N. Mladenovi. *Lower bounds for the uncapacitated facility location problem with user preferences.* Groupe d'études et de recherche en analyse des décisions, HEC Montréal, 2004.

[66] A. Haurie, R. Loulou, and G. Savard. A two-player game model of power cogeneration in new england. *IEEE Transactions on Automatic Control*, 37(9):1451–1456, 1992.

[67] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical programming*, 6(1):62–88, 1974.

[68] K. Holmberg. Solving the staircase cost facility location problem with decomposition and piecewise linearization. *European Journal of Operational Research*, 75(1):41–61, 1994.

[69] K. Holmberg and J. Ling. A lagrangean heuristic for the facility location problem with staircase costs. *European journal of operational research*, 97(1):63–74, 1997.

[70] H. Hotelling. Stability in competition. In *The Collected Economics Articles of Harold Hotelling*, pages 50–63. Springer, 1990.

[71] F.-S. Hsieh and J.-B. Lin. Assessing the benefits of group-buying-based combinatorial reverse auctions. *Electronic Commerce Research and Applications*, 11(4):407–419, 2012. doi: 10.1016/j.elerap.2012.02.003.

[72] A. Hübner and K. Schaal. An integrated assortment and shelf-space optimization model with demand substitution and space-elasticity effects. *European Journal of Operational Research*, 261(1):302–316, 2017.

[73] A. H. Hübner and H. Kuhn. Retail category management: State-of-the-art review of quantitative research and software applications in assortment and shelf space management. *Omega*, 40(2):199–209, 2012.

[74] M. Hyodo, T. Matsuo, and T. Ito. An optimal coalition formation among buyer agents based on a genetic algorithm. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 759–767. Springer, 2003. doi: 10.1007/3-540-45034-3_77.

[75] Y. Ishizuka and E. Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1):73–88, 1992.

[76] A. Iwasaki, S. Ueda, N. Hashimoto, and M. Yokoo. Finding core for coalition structure utilizing dual solution. *Artificial Intelligence*, 222:49–66, 2015.

[77] K. Jakob and P. M. Pruzan. The simple plant location problem: survey and synthesis. *European journal of operational research*, 12:36–81, 1983.

[78] R. G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2):146–164, 1985.

[79] V. V. Kalashnikov, S. Dempe, G. A. Pérez-Valdés, N. I. Kalashnykova, and J.-F. Camacho-Vallejo. Bilevel programming and applications. *Mathematical Problems in Engineering*, 2015, 2015.

[80] L. Kaufman, M. V. Eede, and P. Hansen. A plant and warehouse location problem. *Journal of the Operational Research Society*, 28(3):547–554, 1977.

[81] A. B. Keha. *A polyhedral study of nonconvex piecewise linear optimization.* Georgia Institute of Technology, 2003.

[82] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[83] Y. Kochetov and D. Alexandrov. Simple plant location problem with partial external finance: lower bound, heuristic and exact solution. In *Operations Research Proceedings 1996*, pages 90–94. Springer, 1997.

[84] 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.

[85] A. G. Kök, M. L. Fisher, and R. Vaidyanathan. Assortment planning: Review of literature and industry practice. In *Retail supply chain management*, pages 175–236. Springer, 2015.

[86] C. D. Kolstad and L. S. Lasdon. Derivative evaluation and computational experience with large bilevel mathematical programs. *Journal of optimization theory and applications*, 65(3):485–499, 1990.

[87] H. Küçükaydin, N. Aras, and I. K. Altınel. Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution. *European Journal of Operational Research*, 208(3):206–220, 2011.

[88] C. Li and K. Sycara. Algorithm for combinatorial coalition formation and payoff division in an electronic marketplace. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 120–127, 2002.

[89] C. Li, K. Sycara, and A. Scheller-Wolf. Combinatorial coalition formation for multi-item group-buying with heterogeneous customers. *Decision Support Systems*, 49(1):1–13, 2010.

[90] H. Li and J. Jiang. An optimal combinational coalition formation based on genetic algorithm. In *Proceedings of the 1st International Conference on E-Business Intelligence (ICEBI2010),*, pages 159–165. Atlantis Press, 2010.

[91] X. Liang, L. Ma, L. Xie, and H. Yan. The informational aspect of the group-buying mechanism. *European Journal of Operational Research*, 234(1):331–340, 2014. doi: 10.1016/j.ejor.2013.10.031.

[92] V. Marianov, H. A. Eiselt, and A. Lüer-Villagra. Effects of multipurpose shopping trips on retail store location in a duopoly. *European Journal of Operational Research*, 269(2): 782–792, 2018.

[93] H. M. Markowitz and A. S. Manne. On the solution of discrete programming problems. *Econometrica: journal of the Econometric Society*, pages 84–110, 1957.

[94] R. D. McBride and F. S. Zufryden. An integer programming approach to the optimal product line selection problem. *Marketing Science*, 7(2):126–140, 1988.

[95] I. Moon, K. S. Park, J. Hao, and D. Kim. Joint decisions on product line selection, purchasing, and pricing. *European Journal of Operational Research*, 262(1):207–216, 2017.

[96] O. Morgenstern and J. Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

[97] S. Mou, D. J. Robb, and N. DeHoratius. Retail store operations: Literature review and research directions. *European Journal of Operational Research*, 2017.

[98] S. Narabin and V. Boonjing. A location-price-based buyer coalition. In *2012 IEEE*

12th International Conference on Computer and Information Technology, pages 416–420. IEEE, 2012. doi: 10.1109/CIT.2012.97.

[99] G. P. Papavassilopoulos. Algorithms for static stackelberg games with linear costs and polyhedra constraints. In *1982 21st IEEE Conference on Decision and Control*, pages 647–652. IEEE, 1982.

[100] A. Rahmani and S. MirHassani. Lagrangean relaxation-based algorithm for bi-level problems. *Optimization Methods and Software*, 30(1):1–14, 2015.

[101] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3): 801–817, 2017.

[102] T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings. Coalition structure generation: A survey. *Artificial Intelligence*, 229:139–174, 2015.

[103] G. K. Saharidis and M. G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44 (1):29–51, 2009.

[104] F. S. Salman, R. Ravi, and J. N. Hooker. Solving the capacitated local access network design problem. *INFORMS Journal on Computing*, 20(2):243–254, 2008.

[105] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial intelligence*, 111(1-2):209–238, 1999.

[106] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15(5):265–272, 1994.

[107] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2 (28):307–317, 1953.

[108] H. Shin, S. Park, E. Lee, and W. Benton. A classification of the literature on the planning of substitutable products. *European Journal of Operational Research*, 246(3): 686–699, 2015.

[109] J. Shu, Z. Li, and W. Zhong. A production–transportation problem with piecewise linear cost structures. *IMA Journal of Management Mathematics*, 23(1):67–79, 2012.

[110] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[111] D. Srinivasan, L. T. Trung, and C. Singh. Bidding and cooperation strategies for electricity buyers in power markets. *IEEE Systems Journal*, 10(2):422–433, 2014. doi: 10.1109/JSYST.2014.2329314.

[112] A. Sukstrienwong. Forming buyer coalition with bundles of items by ant colony

optimization. In *Proceedings of the 2011 international conference on Computers and computing*, pages 111–118, 2011.

[113] A. Sukstrienwong. A price-based mechanism for online buyer coalition by genetic algorithms. *International Journal of Innovative Computing, Information and Control (IJICIC)*, 14(05):1653–1679, 2018. doi: 10.24507/ijicic.14.05.1653.

[114] H. Sun, Z. Gao, and J. Wu. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied mathematical modelling*, 32(4): 610–616, 2008.

[115] L. Tran-Thanh, T.-D. Nguyen, T. Rahwan, A. Rogers, and N. R. Jennings. An efficient vector-based representation for coalitional games. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[116] H. Tuy, A. Migdalas, and P. Värbrand. A global optimization approach for the linear two-level program. *Journal of Global Optimization*, 3(1):1–23, 1993.

[117] I. Vasilyev, X. Klimentova, and M. Boccia. Polyhedral study of simple plant location problem with order. *Operations Research Letters*, 41(2):153–158, 2013.

[118] I. L. Vasilyev and K. B. Klimentova. The branch and cut method for the facility location problem with client's preferences. *Journal of Applied and Industrial Mathematics*, 4(3): 441–454, 2010.

[119] L. Vicente, G. Savard, and J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications*, 81(2):379–399, 1994.

[120] J. P. Vielma, A. B. Keha, and G. L. Nemhauser. Nonconvex, lower semicontinuous piecewise linear optimization. *Discrete Optimization*, 5(2):467–488, 2008.

[121] J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations research*, 58(2):303–315, 2010.

[122] T. Voice, M. Polukarov, and N. R. Jennings. Coalition structure generation over graphs. *Journal of Artificial Intelligence Research*, 45:165–196, 2012. doi: 10.1613/jair.3715.

[123] H. Von Stackelberg. *Marktform und gleichgewicht*. J. springer, 1934.

[124] J. Wilson. Cooperative games with transferable utility. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[125] E. Yücel, F. Karaesmen, F. S. Salman, and M. Türkay. Optimizing product assortment under customer-driven demand substitution. *European Journal of Operational Research*, 199(3):759–768, 2009.

[126] Y. Zhang. Designing a retail store network with strategic pricing in a competitive environment. *International Journal of Production Economics*, 159:265–273, 2015.

[127] F. S. Zufryden. A dynamic programming approach for product selection and supermarket shelf-space allocation. *Journal of the operational research society*, 37(4):413–422, 1986.

Appendix

5.2.1 Stabilization root node

The purpose of a stabilization procedure is to reduce the number of times we have to solve the separation problems before we conclude solving each node, i.e., the stabilization procedure ensures a quicker closing of the gap between the upper bound and the dual lower bound.

The classical Kelley scheme, [Kelley, 1960] for cut generating, which is known to have a very bad performance. The reason for this bad performance is that the dual lower bound calculated at each iteration can be very erratic, and as a result, the process has a somewhat slow convergence. Authors as [Fischetti et al., 2016b] have noted that a point out that the convergence behavior of the overall cut strategy greatly depends on the point chosen to be separated at each iteration and, as such, the performance of this cut loop can be easily improved by implementing a simple in-out stabilization procedure like the one shown in [Ben-Ameur and Neto, 2007].

In general, at a given root node iteration, suppose one has a (fractional) solution, aout, obtained from solving the master problem, and a feasible solution for the original MILP, ain. Then, at that iteration, rather than attempting to separate aout, one can separate the following 'intermediate' point instead:

$$a_{sep} = \lambda a_{in} + (1 - \lambda) a_o ut$$

where lambda]0; 1]. Note that when = 1, this amounts to not performing any stabilization, and the lower the value of, the more aggressive the stabilization performed becomes, in the sense that the point that one feeds into the separation problems is closer to a feasible solution. Algorithm 3 shows pseudocode for this stabilization procedure.

Note that it is important to adequately tune the value for a good performance of the stabilization procedure. As in Kelley's scheme, no stabilization can lead to slow convergence, but an over-aggressive stabilization can lead to too much time spent on solving separation subproblems that do not generate violated cuts.

Algorithm 7 Cut generation

```
1: $stop = True$
2: **for** $i \in \mathcal{I}$ **and** $stop = True$ **do**
3:     $S = \{(j,k) | x_{ijk} > 0 \ (j,k) \in \mathcal{T}_i\}$
4:     $jk^i_{\min} = \arg\min_{(j,k)\in S}\{\sigma_i(j,k)\}$
5:     $jk^i_{\max} = \arg\max_{(j,k)\in S}\{\sigma_i(j,k)\}$
6:     **if** $y_{(j,k)^i_{\max}} < 1$ **then**
7:         **for** $i' \in \mathcal{I} \setminus \{i\}$ **do**
8:             $(j,k)^{i'}_{\max} = \arg\max_{(j,k)\in S\cap\mathcal{T}_{i'}}\{\sigma_i(j,k)\}$
9:             **for** $(\hat{j},\hat{k}) \in \mathcal{T}_i \cap [(j,k)^i_{\min}, (j,k)^{i'}_{\max}]$ **do**
10:                **if** $\sum_{(j',k')\in\mathcal{W}_{i\hat{j}\hat{k}}} x_{ij'k'} = y_{(j,k)^i_{max}}$ **then**
11:                    break
12:                **end if**
13:                **if** $\sum_{(j',k')\in\mathcal{W}_{i\hat{j}\hat{k}}} x_{ij'k'} + \sum_{(j',k')\in\mathcal{B}_{ii'\hat{j}\hat{k}}} x_{i'j'k'} \leq 1$ **then**
14:                    add cut$(i, i', \hat{j}, \hat{k})$
15:                    $stop = False$
16:                    break
17:                **end if**
18:            **end for**
19:        **end for**
20:    **end if**
21: **end for**
```

Algorithm 8 Greedy Algorithm for Solving Problems (4.8) without constraint (4.6)

```
1: **Require:** $\boldsymbol{s}^k, Y_k = \sum_{\substack{b_i\in\mathcal{B}j \\ k\in b_i}} y_{i,j}$, set $\mathcal{L}_k$ is ordering s.t. $d_{k,|\mathcal{L}_k|} < \cdots < d_{k,1}$
2: $\tilde{z}_{k,l} = b_{k,l-1}s_{k,l}$ for all $l \in \{2,\ldots,|\mathcal{L}_k|\}$ and $\tilde{z}_{k,1} = 0$
3: $\underline{Z}_k = \sum_{l\in\mathcal{L}_k} \tilde{z}_{k,l}$ , $\overline{Z}_k = \sum_{l\in\mathcal{L}_k} b_{k,l}s_{k,l}$
4: **if** $\underline{Z}_k > Y_k$ or $\overline{Z}_k < Y_k$ **then**
5:     $G_k = \infty$
6:     **break**
7: **else**
8:     **for** $l \in \{|\mathcal{L}_k|,\ldots,1\}$ **do**
9:         $\tilde{z}_{k,l} = \tilde{z}_{k,l} + min\{Y_k - Z_k, b_{k,l}s_{k,l} - \tilde{z}_{k,l}\}$
10:        $Z_k = Z_k + min\{Y_k - Z_k, b_{k,l}s_{k,l} - \tilde{z}_{k,l}\}$
11:        **if** $G_k = Y_k$ **then**
12:            $G_k = \sum_{l\in\mathcal{L}_k} \tilde{z}_{k,l}d_{k,l}$
13:            $l^*_k \leftarrow l$
14:            **break**
15:        **end if**
16:    **end for**
17: **end if**
```

Algorithm 9 Greedy Algorithm for Solving Problems (4.9) without constraint (4.6)

```
1: **Require:** $\boldsymbol{s}^k, l_k^*$ obtained since the Algorithm , $Y_k = \sum_{\substack{b_i \in \mathcal{B}j \\ k \in b_i}} y_{i,j}$
2: $\overline{S}_k = \sum_{l \in \mathcal{L}_k} s_{k,l} b_{k,l}$, $\underline{S}_k = \sum_{l \in \mathcal{L}_k \setminus \{1\}} s_{k,l} b_{k,l-1}$
3: if $\overline{S}_k < Y_k$ then
4:     $\overline{\alpha}_k = -1$
5:     $\overline{\gamma}_{k,l} = 1$ for all $l \in \{2, \ldots, |\mathcal{L}_k|\}$
6: else if $\underline{S}_k > Y_k$ then
7:     $\overline{\alpha}_k = 1$
8:     $\overline{\beta}_{k,l} = -1$ for all $l \in \{1, \ldots, |\mathcal{L}_k|\}$
9: else
10:     $\overline{S}_k = \sum_{l \in \mathcal{L}_k} s_{k,l} b_{k,l}$, $\underline{S}_k = \sum_{l \in \mathcal{L}_k \setminus \{1\}} s_{k,l} b_{k,l-1}$
11:     $\alpha_k = d_{k,l_k^*}$
12:     $\beta_k = 0$ for all $l \in \{1, \ldots, |\mathcal{L}_k|\}$
13:     $\gamma_k = 0$ for all $l \in \{2, \ldots, |\mathcal{L}_k|\}$
14:     for $l \in \{1, \ldots, |\mathcal{L}_k|\}$ do
15:         if $l > l_k^*$ and $s_{k,l} > 0$ then
16:             $\beta_{k,l} = d_{k,l} - d_{k,l_k^*}$
17:         else if $l < l_k^*$ and $s_{k,l} > 0$ and $l > 1$ then
18:             $\gamma_{k,l} = d_{k,l_k^*} - d_{k,l}$
19:         end if
20:     end for
21: end if
```

5.2.2 Primal feasible heuristic