POLITECNICO DI TORINO

Degree course: Management engineering

Thesis developed in the companies: Ram & Dodge (STELLANTIS)

Academic year: 2022/2023

Degree session: 12/2023

Pricing problems applied to the corporate context and environment

SUPERVISOR:

FABIO SALASSA

CANDIDATE:

MARCO VARALDA

INDEX

Introduction and organization of the document

The idea for this thesis comes from the experience I had the opportunity to have during this academic year, i.e., a curricular internship at Stellantis and in particular with the two American brands Ram and Dodge. This experience will be described in details in the next dedicated chapter, while, in this first part, the topic addressed in this document will be briefly explained with an overview of its organization, in order to facilitate the reader in reading and understanding the thesis. The topic addressed is bundle pricing, which is a type of problems that belongs to a larger category, which is algorithmic pricing. Therefore, this thesis first of all required the study of the literature related to this macro-category of problems in order to understand all its main aspects and understand where bundle pricing fits within it. For this reason, the following parts of the thesis aims to give the reader a basic understanding of algorithmic pricing, to explain some of the types of problems that can be found within it and to introduce the concept of bundling by underlining the differences with other types of problems contained within algorithmic pricing (such as combinatorial auctions). Finally, the reader will be introduced to the fulcrum of this thesis, bundle pricing, explaining all the possible problems available in the literature, their differences and we will also try to suggest some taxonomies for their classification, so that the reader can have a view as complete as possible on the subject and a clear idea of how these problems can be classified and tackled. Furthermore, as previously mentioned, an entire chapter will be dedicated to the description of the company internship experience at Ram and Dodge, the topics and issues addressed and the work I had to carry out in the corporate context in the role of product and pricing specialist. This chapter also aims to explain to the reader the connection between this thesis and the internship experience to better understand the motivations that led to the writing of this document.
Finally, after this introductory part, the real core of the thesis will be analyzed by facing a specific bundle pricing problem, named SMBPP, single minded bundle pricing problem (the concept of single minded and many other useful definitions

will be given in the following chapters). This problem will be tackled from the point of view of the corporate context in which I was placed during my internship at Stellantis, giving the reader a more concrete aspect of this type of problem. Obviously, the presentation of this type of problem will be followed by some chapters dedicated to its resolution, using 3 algorithms, one exact and two heuristics. Furthermore, statistical tests will be used to evaluate and compare the results of the algorithms and their implications.

1. Internship and role in the company

1.1 Introduction of the internship

From 13 February to 17 May 2023, I carried out a curricular internship at Stellantis in the role of product and pricing specialist. The internship took place entirely in the city of Turin at the company headquarters with full-time activities. Most of the activity was carried out in smart working, as only one day a week we went to the office in via Plava 80, while for the remaining 4 days we worked from home. At the end of this internship period, they also offered the opportunity to carry out my thesis in the company, given the interest in the topics covered.
As previously mentioned, the internship took place at Stellantis, an international holding company specialized in the production of motor vehicles. The company was born from the merger of the Italian-American group FCA (Fiat Chrysler automobiles) and the French PSA (Peugeot S.A) and has its registered office in the Netherlands. This company currently has more than 400,000 employees and revenues of approximately 180 billion. The birth of Stellantis dates back to October 2019, when FCA and PSA announced to the world with a press release their intentions to merge in one large group to become global leaders in the future of sustainable mobility. Over the next two years there were several steps which led, on 4 January 2021, to the approval of the merger by the shareholders' meetings of the two groups. The name of the new company was announced in July 2020 and comes from the latin "stello", meaning "lit by the stars". Even the new company logo takes up this concept, in fact, according to what its creators express, the logo is «the visual representation of the spirit of optimism, energy and renewal of a diversified and innovative company, determined to become one of the new leaders of the next era of sustainable mobility». The peculiar feature of this logo concerns the vowel "A" (stylized with two oblique lines), with the intention of stylistically representing shooting stars, in reference to the meaning of the name Stellantis.

The company is made up of fourteen car brands: Abarth, Alfa Romeo, Chrysler, Citroën, Dodge, DS Automobiles, FIAT, Jeep, Lancia, Maserati, Opel, Peugeot, Ram Trucks and Vauxhall. The curricular internship took place in the Dodge and RAM brands. Dodge is an American brand founded in 1914 under the name of Dodge Brothers Company, after Henry Ford refused to buy their company. RAM Trucks, instead, was born in 2009 from the separation from the Dodge brand, in particular from the Dodge Trucks division.
This curricular internship is part of the completion of the master's degree in management engineering at Politecnico di Torino and was carried out under the supervision of the academic tutor Fabio Salassa and the company tutor Andrea Paolo Maria del Panta.
The team in which I was placed is made up of about 15 people (of which about 10 are Italian and the remaining 5 from other countries) and deals with the management of the RAM and Dodge brands in Europe (and also some other Asian countries) from all points of view (pricing, finance, marketing, etc.). The only activity the team does not take care of is production, because, being RAM and Dodge American brands, production takes place exclusively overseas. For this reason, the distribution network is a little more complicated than the one of other brands. In particular, it uses the role of the so-called traders (in our case the two main traders are AEC and KW) who deal with the importation of cars from the United States, the distribution to European car dealers and their homologation. This latter activity is essential to allow the cars to be used in Europe because, being produced in America, they must be adapted to European standards and those of the individual countries in which they are sold. For example, if we want to import a car from the United States and sell it in Europe, it will need to be homologated with German standards, which could be different from the standards set by the legislation of another European country (for instance, some options or accessories could be incompatible with the safety or environmental regulations of some countries).
The role covered in the company is product and pricing specialist and had as general objective the monitoring of car prices, the analysis of market trends and competitors and finally the study of the products offered to customers in terms of models and packages available on the cars, production times and placing on the market. More in details, the activity was the following:

Pricing:

- Define the pricing competitive scenario, regulations, markets trends and constraints analysis;
- Define the pricing leverage management, during the entire product lifecycle, according to volumes and profitability targets;
- Monitor the application in Europe of Brand Pricing strategy;
- Define and monitor Pricing KPIs.

Product:

- Monitor competitor actions, industry trends and all relevant customer data with relevant stakeholders (BC operations, Pricing, BMC, Design) to identify and initiate product actions;
- Contribute to the briefing of new products initiatives, providing commercial parameters;
- Define product commercial launch plans (launch date, launch stock, opening edition...) and contribute to the launches in Europe;
- Monitor monthly price positioning.

Furthermore, during this internship I had the opportunity to participate in marketing activities, helping the team in the search for investment opportunities, in the management of social media and in the relationship with the consultancy agencies in charge of managing various aspects of marketing. All these activities will be analyzed and explained in detail in the following paragraphs.

1.2 First activity: pricing

The main activity carried out within the team is pricing, which means dealing with various aspects: monitoring the prices of your vehicles on the market in order to understand trends, for example if some vehicles are depreciating or if they are gaining value and why. Often these price changes are due to changes in the sales mix, other times they are due to an increase (or decrease) in the sale of more equipped vehicles. Secondly, you need to monitor competitors' prices and vehicles, in order to check that your products are always correctly positioned on the market in terms of value for money (features offered vs price). Other activities that concern pricing are, for example, the updating of the prices of optional and product characteristics, as they change continuously, above all if we think in terms of value perceived by the customer (the automatic windows of the car could be perceived as a great added value a few years ago, but today they are taken for granted on all new vehicles, consequently the added value perceived by the customer is almost zero).

One of the activities in which I was certainly most involved was the drafting of the pricing reports, infact I was assigned to this job at the beginning of the internship and it was always my responsibility until the end of the relationship with the company. This is a monthly report which monitors the price trend of our cars for sale in some selected dealers. In our case, this report is carried out both for RAM and for Dodge in the German territory (even if the data collection also takes into account two dealers in the Netherlands and Belgium), as it represents the main market and consequently the deriving data from this area are also statistically more interesting. Going into more details, at the end of each month, and within the first week of the following month, my job was to visit the websites of the selected dealers and see the offers on them regarding our cars. Then, for each of these cars, some characteristics were recorded in a special excel file (such as price, model, version, type of fuel, whether the car is new or used, year of production, etc.). Once all the cars present on all the dealers' websites had been registered, we proceeded with the data analysis. At this point, what I had to do was to create graphs that included all the data collected up to that moment (therefore not only the data of the current month, but also all those collected in the past months) in order to be able to analyze the different trends : we always started with an analysis of the main models of RAM and Dodge, then we went into more details by analyzing the best-selling versions or those that, for some reason, deserved more attention, for example a new special model just arrived on the market, in order to understand the interest in these new vehicles. Trend analysis consisted in understanding the price trend and verifying the presence of anomalies in them. For example, sometimes it happened to see increases or decreases in the prices of certain models, even substantial ones and often these were justified by changes in the sales mix. In other words, from one month to the next the cars available on the dealers' websites changed, they switched to a more (or less) "expensive" mix, as among the models sold by RAM and, above all by Dodge, there are versions of the same model very different in terms of price. Sometimes there can even be a difference of 30,000 euros from one to the other, due to different characteristics (engines, accessories, aesthetics, aerodynamics, etc.). The consequence of this is that when you move towards a mix composed mainly of more expensive versions, the average price increases, even significantly. Another important part of this work was monitoring price variability. In particular, in the analysis of a specific model, graphs are constructed with the average, minimum and maximum price. In this way it is possible to check, for that particular model, the variability of its price and other interesting information. In this analysis, the number of units available on the dealers' websites plays an important role, in fact the most variable trends are those with fewer units available because if we have, for example, only 3 units available (one with a high price, one medium and one low), when the one with the highest price is sold, we will see significant decrease of the average price.

Another activity I carried out during my internship was the preparation of a document to control the prices of cars with specific optional (OPTs). This activity was particularly stimulating as it was developed entirely by myself and is now a tool used by some members of the pricing team. The work I did consisted in creating an excel file that would allow you to enter a particular RAM or Dodge model and a number of OPTs chosen by the user (obviously among those available for the selected model), then clicking on the "generate configuration" button, it would print on another sheet all the details entered, the price of the car model, the price of the OPTs and the total (i.e., the sum of the two prices). This file is useful for the company because, when they have to sell cars to traders, they make an offer for the models and the OPTs (for example, if a model with new particular optional is launched, it is very likely that the company will suggest traders to buy these OPTs). The traders at this point will make their requests and can decide whether or not to accept the company's offer, but some difference on the required options is very common. For this reason, the file that I created during the internship is important for monitoring the price differences between the traders' requests and the company's offers and allows for some interesting analysis. It can also be used to create a configuration and see the price of it, in case you are evaluating the potential impact on pricing of some business decision, such as removing, adding or changing an OPT's package. For example, suppose we have a package with 10 OPTs, if two of them are removed from the package and a new one is added, it is possible to understand what the price of this package should be by selecting all the individual OPTs contained in the new package.

1.3 Second activity: product

The second main activity carried out during the internship was related to RAM and Dodge products and required an initial phase of study and introduction to company issues. In fact, as previously mentioned, the car distribution system for these two brands is a little more complicated than the one of other brands, as production takes place exclusively in the United States and therefore requires the involvement of traders for the import and approval. Furthermore Dodge, but above all RAM, have a wide range of cars available and different versions of them. For this reason, a first phase of study of all the cars available was necessary in order to understand their differences.
There were various activities related to the product, for example the management of new models to be introduced on the market, and this is linked to the pricing activity as it was necessary to understand at what price to launch these cars on the European market. Then I fill in the so-called wish lists, which are used to communicate to the global brand which accessories to insert on the cars to be imported, in fact not all the accessories present on American cars were interesting for us, or they simply could not be imported because they did not comply with the rules and standards of European countries. I also dealt entirely with another activity which is the product timing. Since the production is entirely located in the United States, it is very important to keep track of the production

and distribution times of the cars. Within our team, this is done once every two weeks (approximately in the middle and at the end of the month) and this task has been assigned to me starting around the middle of the internship. In particular, a member of our team uploaded an excel file every two weeks containing all the orders for RAM and Dodge cars that had arrived up to that date. Once I received the file, my task was to check the number of orders received for each car model and check the production dates associated with them. In fact, when an order is received, it is sent to the global brand in the United States, who communicate the promised production dates. For example, if 10 orders are received for the new RAM 1500 Limited model in January, the United States informs us about the expected production date, for example in March. At this point, if there are no problems, production will start in March and the promised date will become the effective production date. But sometimes it can happen that the promised date can be postponed (or in some cases anticipated). For example, if they tell us that the production date has been postponed to April, we must take this into account. In fact, after the production phase there are some intermediate steps before these cars are available at European dealers and they require a certain fixed time, consequently if production is postponed, the arrival of the cars at the dealerships will be postponed too. My task was therefore to keep the production dates under control in order to inform the team in case of delays.

1.4 Other activities

As previously mentioned, this internship gave me the opportunity to address different issues than those stated in the internship proposal. This is because the team I've been part of is very united and everyone closely monitors all company issues through various update meetings, then obviously there are people in charge of further investigating certain aspects. In fact, the first activity carried out within the team concerned marketing for Dodge and I was also involved in the partnership with the KTM racing team (also given my personal interest in the world of motorsport), already started last year and strengthened starting since the beginning of 2023. I also helped another member of the team in analyzing the so-

called google trends and accesses to the RAM and Dodge websites in order to verify the trend of the last 3 years and analyze the effectiveness of marketing campaigns and the launches of new vehicles. This involvement in marketing activities was very challenging as it was not what I expected to do, so it was a challenge that I was happy to take on.
The first task that was assigned to me at the beginning of the internship concerned marketing and consisted in finding investment territories for the Dodge brand. In fact, the day I arrived, I immediately followed a meeting with an agency that had taken care of verifying the possibility of investing in some sport events, doing very detailed research on people's interests in some areas (sports, music, video games, etc.) in some European countries selected by our team, including Italy, Germany, Czech Republic, Spain etc. At the end of this sponsorship strategy made by the agency, the conclusion was that there were some Red Bull events with a low presence of competitors that could be interesting for our brand. My task was therefore to search and select Red Bull events in which to invest, respecting some constraints established by the team. Indeed, Dodge is a particular brand that has a small niche customer base that is very passionate about motorsport and this makes the marketing activity more complicated. If the product you have to sell is widely known and appreciated by a large number of people, the marketing choices will be easier, but if instead, as in our case, there are fewer customers and with very specific interests and characteristics, the marketing campaigns will have to be very targeted, in order to reach only that potential customer, especially if the economic availability of these campaigns is limited. The constraints imposed by the team were therefore dictated by these two needs: we had to try to reach only our potential customers and do so using the least possible amount of money. Consequently, the search for events was limited to Germany (which is by far the main market for Dodge) and was carried out within three disciplines: cycling, motorsport and extreme sports, although in reality my search is has also been extended to other sports such as music & dance, E-sports and boating and this has been appreciated by the team.
For what concern RAM, marketing activity has been focused on the partnership with KTM right from the start. In this case I was very lucky to be included in this job given my passion for motorcycles. This partnership started last year and was done with KTM racing, which is the division of KTM that deals with racing in the

ambits of motocross, enduro and extreme enduro. Seeing my particular interest, my company tutor decided to involve me in this activity right from the start. As already mentioned before, the partnership with KTM has been renewed and strengthened in 2023 with a greater presence of RAM in racing (particularly motocross). This involves the presence of cars on the motocross track and in the paddock, the presence of the RAM logo behind the starting grids, on the riders' jerseys and bikes and behind the podium during the prize giving. Furthermore, this year's partnership also provides for greater involvement and greater interaction between RAM and KTM dealers.
Another activity that I did for the company is the analysis of the so-called google trends. One of the methods used by the team to verify the interest shown by people in their brands is to analyze the number of accesses to their websites and the google trends. In fact, Google offers a service that allows you to view the search history of a specific word, also inserting filters on it, for example in our case by searching for the word "RAM" we risked confusing our car searches with those for Random Access Memory, but google allows you to limit searches to the automotive sector, allowing us to do our analyses. To do this, I helped another trainee of our team who was in charge of collecting the data and building the graphs, while I had to analyze and comment the trends. Going into more details, we used this google tool, to see the number of people who had typed the words "RAM" and "Dodge" in the google search bar and we compared them with those of other competitors or other brands, such as Alfa Romeo, Cupra, Ford, BMW etc. Google also allows you to download this data in Excel and then to build graphs that allow you to analyze trends. My analysis was not limited to understanding the trend of these numbers, but also looking for any seasonal trend and verifying the effectiveness of some company decisions. For example, if a new model was announced in September 2022, or a marketing campaign on social media had started, its effectiveness could be verified by looking at the increase in searches for the brand in the following months, if successful, the numbers increased otherwise they remained stable. Furthermore, the analysis of the other brands also made it possible to compare the performance of RAM and Dodge with those of the other brands and see the impacts of some particular periods such as the pandemic in 2020.

In addition to Google trends, access to our brand websites by the users was also analyzed. In particular, we collected data on access to US and European websites by users from European countries selected by the team, which were also the most developed markets in this case (Germany, Czech Republic, Netherlands, Sweden, etc.). Through these numbers it was possible to verify the interest and effectiveness of websites during the last 3 years, the percentage of accesses of European users on the total and, by comparing these data with those of google trends, it was possible to go even further. In fact, in some cases we saw that, even if the accesses to the website were decreasing, the google trends remained stable, or even grew. From this we could deduce that the problem of the loss of interest was limited to websites because people was still searching for information on RAM and Dodge but they did so in places other than websites (articles, social pages, forums, etc.). This was therefore a negative figure for the website, but not for the brand in general which, by contrast, was able to maintain stable interest.

1.5 Final consideration about the internship

I think this internship experience was positive, above all for the fact that it was my first real work experience in a large company and it allowed me to see how all the internal dynamics and procedures of a multinational company work. The people I have been worked with have always been available to explain concepts to me that was not clear at first glance. The fact that I was able to work in a context in which everyone helps each other was also very positive, because, as I explained in detail in the previous paragraphs, I had the opportunity to see different areas of work, from pricing to marketing and I think that this is extremely positive for a curricular internship, the objective of which is precisely to acquire as many skills as possible and better understand the world of work and in particular the corporate context. This experience therefore allowed me, on one hand, to learn notions that I lacked, such as the entire marketing part, which was not addressed during the studies at Politecnico, and on the other hand, to apply concepts that I already knew and that I had learned during university, allowing me to see a more practical application of these concepts. One of the university courses that I found useful in this internship was strategy and organization held by Professor Neirotti during the first year of the master's degree, in particular the group work proposed during the course which allowed us to apply some theoretical concepts precisely to the automotive sector (i.e., the same sector in which I did my internship), including pricing and strategies to decide the best price for cars. Other courses that helped me in this internship are economics and industrial economics, held by professors Benfratello and D'Ambrosio, during which some issues concerning pricing and bundles were addressed. For example, given a certain model of car and given a bundle of accessories on it, what is the best price to fix knowing that customers have a predetermined budget and will buy the car and accessories only if the price does not exceed the budget? These problems, addressed in a more simplistic way during the courses, were real problems faced by our team, as I have already explained earlier talking about OPTs pricing analysis. The interest in the latter topic was precisely what convinced me to write this thesis, as during the internship I had the opportunity to apply the theoretical concepts of this topic to a real case.

2. Algorithmic pricing

2.1 Introduction

Every company nowadays is facing various problems during the daily management of its business, from marketing to communication, from production to logistics. One of the fundamental aspects that a company which offers products and services to its customers must manage is pricing, i.e., deciding what is the best price for its products/services in order to achieve a specific objective. This objective depends on what the company wants to achieve in the short and long term, for example the maximization of sales volumes, but much more often the maximization of its profits. If pricing decisions are not made accurately and without the necessary analysis and research, this could lead to huge loss of profits for the company. For example, a low price for your products could lead to a sharp increase in demand, but a sharp reduction in the profit margin. Furthermore, too high demand may be difficult, if not impossible, to meet due to production capacity constraints. On the other hand, however, too high price could on one side increase the margin on the single product sold, but on the other side decrease the number of customers willing to pay that price to buy the product, with all the consequences (for example, a customer could turn to a competitor to buy a similar product). From these two simple examples it is easy to understand the importance of pricing decisions and in particular the possible consequences of wrong strategies.
These considerations that have just been done lead to the definition of the concept of pricing problem, a two-level game between the company and its customers in which the company fixes the prices trying to maximize its profits, while the customers. Based on the price set by the company, they decide whether to buy or not the products offered. This definition of a two-level game recalls the concept of leader and follower: In these types of problems there is a seller (in our case the company) who sets prices trying to maximize his objective function, while in a second stage of the game the buyers (the company's

customers) decide whether and which product to buy in order to maximize their objective function, which will obviously be different from that of the company and will take into account the price that the individual customer is willing to pay for the products offered by the company. Being the leader, the company must act first. Obviously, the possibility of knowing information related to customer preferences ex ante (that is, before establishing the prices of its products) represents a great advantage. With the advent of the internet and new technologies, obtaining this type of information is increasingly simpler than in the past. In fact, companies can acquire information about the market, industries, trends and future forecasts in each sector. Furthermore, it is also possible to obtain information on specific customers: their preferences, what they buy and how often, how much they spend, etc. This obviously facilitates the decision-making process of those within the company who have to decide on pricing strategies.

After these premises we can finally give a definition of algorithmic pricing, even if, as previously specified, this subject is really very broad and includes a great variety of problems that are also significantly different from each other. Algorithmic pricing has the objective to study and solve the computational problems faced by a company when it has to determine the prices for its products and/or services with the aim of maximizing profits, using all available information about customer preferences. From this point of view, it is possible to define the concept of pricing as a menu of prices offered by the company. Given pricing, consumer preferences represent the desired allocation of buyers. Knowing the pricing and consumer preferences, the algorithmic pricing problem consists in cross-referencing these two data to maximize the seller's profits. What has not yet been explained is the term "algorithmic" in the definition of algorithmic pricing. In fact, the ever-growing trend in recent years is to entrust price decisions to algorithms that allow, given the constraints of the problem, the objective functions and all the other necessary data, to establish the best price for the products. Even in some cases, particularly in dynamic pricing, price decisions are entrusted completely to algorithms that automatically and dynamically adjust the prices of products/services on the basis of the information available at that moment (for example market conditions or fluctuations in supply and demand) and are able to process a large amount of data in a short time, allowing for the rapid adoption of

effective strategies. We will not go into detail on dynamic pricing now as this topic will be discussed more extensively in the following pages.
Although the literature on algorithmic pricing dates back to several years ago, it has become particularly rich in recent years (especially the last decade), as these algorithms to find the best pricing strategies have been used by the biggest brands in the world such as Amazon, Google, Microsoft, etc. This has led to questions about the ethics of these algorithms because, having as their objective the maximization of corporate profits, they can lead to incorrect behavior towards customers (think, for example, about pricing strategies which aim to maximize the price in order to extract all the consumer surplus). Furthermore, the practice of entrusting the pricing decisions to an algorithm also has legal implications, as it is necessary to ensure transparency in the management of customer data and to establish responsibility in case this data is not used in a proper way. However, the ethical and legal aspects of algorithmic pricing require a very deep study and will therefore not be analyzed in this thesis.

2.2 Some definitions

After this first introduction, some examples of the most common pricing problems will be presented in order to give the reader a clearer idea of the topic. But first, however, it is worth spending a few lines to give some important definitions that are often found in pricing problems and whose knowledge makes it easier to understand the various types of problems. The suggestion for the reader is therefore to use this short paragraph as a legend to understand the terms that will be used in the following pages.
Bundle: it is a set of products or services selected from all the seller's products and sold together as a single offer. In other words, it's a combination of items that can be purchased together, usually at a better price than buying the components individually.
Reservation price (or budget): it is the maximum price that a customer is willing to pay for a product or service offered by the seller. It is the price level above which the customer believes that the purchase is not convenient. The reservation

price is an individual threshold and can vary from person to person based on their preferences, needs, budget and perception of the value of the product or service.
Limited supply: It means that there is a limited quantity or availability of the product or service offered by the seller, which may not be sufficient to fully satisfy consumer demand. This is usually the case of tangible products (e.g., newspapers).
Unlimited supply: is a term used to describe a situation where the supply of a product or service is abundant and is not subject to restrictions on availability. In other words, the product or service is available in sufficient quantities to fully satisfy consumer demand. This is usually the case of digital products (e.g., online newspapers).
Unit-demand customer: is a type of customer who is interested in different bundles of products or services and has a valuation for each bundle offered by the company. Given the price of each bundle, this type of customer will choose the one that gives them the highest utility (calculated as the difference between their valuation and the price of a given bundle).
Single-minded customer: Unlike the unit-demand customer, the single-minded customer is interested in only one bundle offered by the seller. By definition, this type of customer will only have a positive evaluation for the bundle they are interested in, while they will attribute a null value to all other bundles. Consequently, the customer will buy the bundle if it is cheap enough, otherwise they will buy nothing.
Envy-free: is a term that describes an allocation of products between the buyers in which one customer does not envy the price or value of another customer's offer. In other words, a customer is satisfied with the price he pays for a product or service and would not want to exchange his price for that of another customer.

2.3 Examples of pricing problems

Algorithmic pricing is a set of techniques and methodologies for solving pricing problems. For this reason, it is worth listing some of the most common of these problems in order to familiarize not only with pricing problems, but also with the terminology used in them. The problems that will be used in this paragraph are not part of the typology that will be extensively discussed in this thesis, but they are still a useful tool for understanding the topic. The types of problems that will be briefly described are the rank pricing problem and the network pricing problem, for which a rich literature can be found.

Rank pricing problem:

The Rank Pricing Problem (RPP) is a multi-product pricing problem in which customers have to choose between different products offered by the seller and they are of the unit-demand type, i.e., they attribute different evaluations to the products and in this specific case they are interested in buying only one product (thus excluding the case of multiple purchases by the same buyer). Customers also have their own ranking of product candidates which then leads to the creation of a customer preference list. It is assumed that the customer's budget is uniform (does not change from product to product), so once prices are set by the company, customers act as followers and solves their own optimization problem, buying the highest rated product they can afford (if any). In other words, the customer has a predetermined budget and must decide which product to buy, obviously his choice will be the highest-ranking product (with the highest valuation) among those he can afford (which means with a price below the budget). All products within a customer's budget are assumed to be available, i.e., we consider an unlimited supply of products. In more mathematical terms it can be said that the seller offers a set of $K = \{1, \dots, k, \dots, K\}$ products to a group of $I = \{1, \dots, i, \dots I\}$ customers, the latter have a preference value S_i^k for each product and $S_i^k > S_i^{k+1}$ means that customer I prefers product k over product k+1.

There are obviously several variants of this problem which can be obtained by changing some characteristics of the problem itself or of the actors present in it (buyers and sellers). The most famous variant is that of the Capacitated rank pricing problem in which the assumption of unlimited availability of the seller's products is deleted. In this case the problem can be further complicated by looking for an envy-free solution. In fact, in the case of unlimited supply, the solution is by definition envy-free, but this does not apply in the case of limited supply.

Given that the rank pricing problem implies unit-demand customers, it is particularly interesting for the purposes of this thesis, because this type of customer is, for example, very common in the automotive sector, i.e., the field in which I did my internship. In fact, when a person has to buy a car, he usually has a budget and is interested in buying only one of the products offered by the seller and has a positive evaluation for each of them.

Network pricing problem:

With network pricing problem (NPP) we mean a bi-level pricing problem on a network, in which there are two types of arcs: a first subset of arcs is owned by an authority (for example a municipality) or a company that imposes a cost (toll) on the users who use them and another subset of remaining arcs on which instead there is no toll (think, for example, about highway and alternative free roads). The authority/company aims to maximize toll revenue, while the users want to minimize their total travel cost (total travel cost = toll + travel cost. Infact, arcs not managed by the authority/company does not have zero cost, they are simply not subject to tolls, but still have a travel cost) and therefore will always travel on their least cost route between the origin and the destination. Furthermore, users choose their best path once authorities have set tolls, thus having complete knowledge of all network costs. The authority therefore has to take into account the reaction of users: too high tolls may convince users not to use toll arches, while if the prices are too low, the authority may lose potential revenue.

Also in this case there are different variants of the same problem, for example there is the case of the Network pricing problem with unit toll, which means that all the arcs managed by the company have an equal cost, while in the more general case the cost of each arc depends on some parameters such as the length of the arc itself.

It should be noted that in all these cases of NPP just mentioned, the capacity of the roads is considered as unlimited, which means there is not a maximum number of users in the roads in the same time (unlimited supply).

3. Main elements of algorithmic pricing

After a first introductory and theoretical part on algorithmic pricing and after having seen some examples of pricing problems, it is necessary to try to highlight some of the types of problems that are contained in this vast macro-topic. As has already been said, we will not focus on all algorithmic pricing contains because it would require a specific and very in-depth study, but we will try to list and briefly explain some elements that are closest to the object of this thesis. 4 topics have been selected and they will be analyzed in the following pages. In order to present them, we will use the image below (Figure 1).

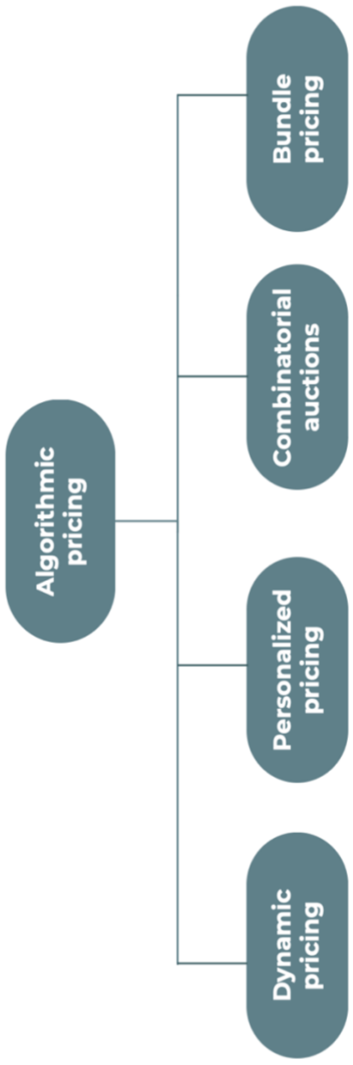


Figure 1: Main elements of algorithmic pricing

The categories of problems that we will address in this chapter are therefore: dynamic pricing, personalized pricing, combinatorial auctions and bundle pricing. In the next paragraphs they will be explained in details, in particular the first 3. Instead, the next chapters will be dedicated to bundle pricing, as it is the central topic of the thesis.

3.1 Dynamic pricing

Nowadays, especially with the advent of e-commerce and digitization, pricing strategies are undergoing a radical transformation. Among all the strategies used today, dynamic pricing represents one of the most promising and advanced techniques for companies and for maximizing their profits. it is possible to define dynamic pricing as a pricing strategy based on the real-time adjustment of the prices of the products/services of a seller (in our case a company) on the basis of various information related not only to the consumers themselves, for example their demand in a given instant of time, but also to external conditions of the market and the sector in which the company operates. The fields in which dynamic pricing is used today are many and in continuous expansion, such as retail trade (especially online), public transport, all areas related to tourism, therefore travel, the hotel industry, etc. In fact, thanks to the analysis of data and information available, it is possible to adapt prices to market supply and demand, but also to more complex factors such as seasonality and market trends. Regarding this, Douglas Ivester, CEO of Coca-Cola from 1997 to 2000, pronounced a famous sentence that may seem trivial, but can help to understand this topic:

“Would you pay more for a coke on a hot day?”

With this sentence he discussed the possibility of introducing an automated system that regulates the price of Coca-Cola in the vending machines based on the outside temperature. Ivester had in fact understood that the utility of Coca-Cola varied from one moment to the others. During a hot summer day, in front of

an important sport final, its utility for people attending the match is at the maximum, for this reason it is advisable to raise the price. This demonstrates how the price of a product can be adjusted according to the season to maximize the company's profits.
This last aspect, however, makes us understand what the potential risks of dynamic pricing may be. In fact, these tools must be used with caution due to the ethical aspects behind it, such as fairness towards consumers, but also corporate reputation. In fact, using again the example of Coca-Cola, it is easy to imagine that consumers might not be satisfied with a pricing strategy in which the heat is "exploited" to raise prices, because it is true that the utility of a drink increases with the outside temperature, but it is also true that the product offered to the customer is the same, consequently the latter will not be happy to pay a higher price. This can therefore represent unfair behavior towards consumers and consequently a decline in the company's reputation. The ethical aspects behind dynamic pricing enjoy a rich literature in this regard, both from the company side and from the consumer side. Even the possible strategic responses of the buyers have been analyzed, such as for example controlling prices and information available ex ante or even planning purchases in order to avoid price increases. However, as already mentioned for algorithmic pricing, the ethical aspects behind these strategies and methods are not the subject of this thesis and will not be further explored.
To better explain dynamic pricing, examples in which this strategy is adopted in real cases will be presented. The first example is the scientific paper "Matching of everyday power supply and demand with dynamic pricing: Problem formalization and conceptual analysis" (Theate, Sutera, Ernst - 2023). This paper is interesting because it is a dynamic pricing problem applied to one of the most interesting and discussed sectors of the moment, renewable energies. In fact, climate change is posing very complicated challenges that must be faced from different points of view. The problem highlighted in this paper is that of the intermittence of renewable sources, in fact they are an important alternative tool to traditional energy sources, but their availability depends on uncontrollable factors (for example, wind energy depends on the wind, while photovoltaic energy depends on sunlight). This translates into an intermittent supply of energy to the customer. The paper therefore proposes a dynamic pricing model that allows for a more

"overlapped" demand curve on the supply curve. In other words, the prices of the energy produced by renewable sources are adapted to their availability at a given instant of time. When the offer is very high, thanks to certain climatic factors, prices decrease in order to allow access to this type of clean energy to the greatest number of people. On the other hand, when supply is scarce, prices rise in order to reduce demand. A graphical result of this paper and the effectiveness of dynamic pricing in this type of problem is shown in figure 2 (taken directly from the paper).

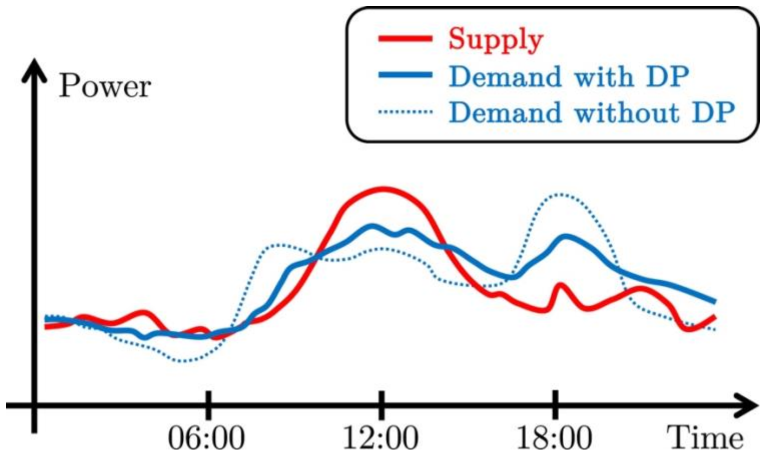


Figure 2: Overlap of supply and demand with and without dynamic pricing

The case just studied is actually a very simple example of dynamic pricing, but much more complex problems can be found in the literature. For example, the next problem that will be analyzed is covered in the paper "Optimizing E-tailer Profits and Customer Savings: Pricing Multistage Customized Online Bundles", published in 2011 in the journal Marketing Science by Yuanchun Jiang, Jennifer Shang, Chris F. Kemerer and Yezheng Liu. This paper was selected because it combines dynamic pricing with the central topic of this thesis, bundle pricing. The topic deals with internet retailing, i.e., a seller who owns an online store offers his products to users who visit the site. An example can be a site for a clothing shop, sports goods or technology store within which it is possible to create a cart of products to buy. The paper concerns in particular the so-called ODR (online recommendation system), in which, when a user places a product in the cart, the site automatically suggests other items that may be of interest to the customer. For example, if we imagine a sport goods website, if the user puts a soccer ball in

the cart, the site might suggest soccer shoes. As the user places products in the cart, the system collects more and more information and will be able to suggest more and more specific items. All this is also associated with an advantage for the customer, in fact, by buying the products suggested by the system, the customer earns a discount on the total purchases (ex. 5% discount on the final cart). This type of problem is called ODBP, online dynamic bundle pricing, and it is a useful tool for maximizing company profits, but at the same time guaranteeing savings for customers, in this sense it can be considered a win-win strategy.
This last problem just described is also interesting for another reason, i.e., that from certain points of view it recalls the second category of algorithmic pricing problems, personalized pricing. In fact, these problems are not completely disconnected, and sometimes it is possible to find "borderline" problems that seem to be a bridge between two different categories. In fact, in the last paper, the choice of the product to suggest was based on the products inserted by the consumer in the cart. This then takes us to the next category, personalized pricing.

3.2 Personalized pricing

Personalized pricing develops in a context similar to that of dynamic pricing, but has had a rapid growth in recent years with the advent of e-commerce, in fact many companies seek to make the customer experience more and more personalized in order to increase loyalty for their brand. There is no better way to present and explain what personalized pricing is than to explain the traditional approach to which it opposes, the so-called "one size fits all". This method is a uniform and standardized approach which consists in applying the same rules to all individuals, without taking into account the diversity of clients. Each situation is treated in the same way, without adapting the solution to the specific characteristics of that situation. It might seem that this approach does not make sense and that customization is always preferable to the one size fits all approach, but the reason why it has been and still is used lies in a very simple

reason. It simplifies, standardizes and unifies company processes and policies, reducing their complexity and ensuring good efficiency in different situations. However, this method has several limitations, as ignoring the needs of individual customers may result in a suboptimal solution and may not meet customer expectations. In a real case, the one size fits all approach could consist in offering exactly the same product to all customers, but often personalization based on the characteristics of the single customer can improve company results. We can therefore define personalized pricing as a pricing strategy that differs from traditional methods as it is based on the segmentation of the company's customers and the use of personal and historical data to establish individual prices for each potential buyer. If the objective is only to maximize the seller's profits, it can be seen as a sort of first-degree price discrimination based on the customer's willingness to pay, but the customization of the offer can also take place to increase customer satisfaction, for example in the case of personalized prices based on the area in which customers live.
So, it is now easy to understand the difference between dynamic pricing and personalized pricing: The first concerns the factors that determine a price variation, in fact in dynamic pricing these factors concern the market, supply and demand, seasonality, etc. while in personalized pricing these factors concern the customer himself. The second difference concerns the objective of the two strategies: dynamic pricing has the sole objective of maximizing the company's profit, while personalized pricing also includes among the objectives that of improving customer satisfaction. Despite these clear theoretical differences, in real cases these two pricing strategies overlap in many cases and are applied in combination, as for example in the last paper described in the paragraph on dynamic pricing (2.1). In fact, it is very common to find papers in the literature dealing with the so-called personalized dynamic pricing problems, which is precisely a combination of the two strategies.
As was said at the beginning, personalized pricing experienced its boom after the advent of e-commerce, this is because on websites it is possible to track data, preferences and user behavior thanks to the use of cookies. What customers often don't know is the enormous amount of data that a simple web browsing can bring with them, such as likes, purchasing history, location data, etc. For this

reason, one of the most discussed topics in the literature regarding personalized pricing is the ethics and legal issues behind this data tracking system.
There are many examples in the literature dealing with this topic, some interesting examples of personalized pricing can be taken from the paper "Personalized Pricing and Quality Differentiation" (Choudhary, Ghose, Mukhopadhyay, Rajan), published in 2005 on Management Science. Without going into the details of the paper, it explains how personalized pricing can be found in different sectors, for example where there are companies that need to have large enterprise-level software. In this case, the seller and the buyer collaborate to carry out the so-called return on investment (ROI) analysis, in which they try to understand the benefit for the customer deriving from the sale of the product. This analysis is used to determine the price of the software, calculated as a percentage of this mutually agreed ROI. The higher the ROI for a certain customer, the higher will be the price for him. This practice of setting the price based on the customer is not only common in the software industry, but also in the healthcare, chemical industry, sale of enterprise telephone cost auditing software, etc.
Another interesting case reported in the same paper cited above is that of e-commerce, in particular Amazon and its online retailing business. Initially Amazon had already tried to apply personalized pricing strategies, trying to offer different prices to different consumers for their DVDs. Obviously, however, the customers who managed to get information about the prices of other users and who saw a lower price for the others were certainly not satisfied, this in fact caused a backlash of the customers, which is why this Amazon experiment was short-lived. However, in the following years, the company continued to try to apply personalized pricing, without annoying customers. An example is the so-called Amazon Gold Box. In this system, each customer was associated with a gold box with his name, which contained discounted products. However, these discounts were only available for a specific customer and only within the gold box, while outside of it the products did not have the same discount. This allowed Amazon to charge different prices to different consumers and is an example of the continuing evolution of personalized pricing.

3.3 Combinatorial auctions

Since bundle pricing will be explained in details in the next chapter, combinatorial auctions are the last category covered in algorithmic pricing. Also in this case the topic is very large, and would require a dedicated study, but since this is not the topic we want to discuss in this thesis, we will try to treat it in a more superficial way, to allow the reader to understand it without going into the technicalities behind it. The literature concerning combinatorial auctions is far from recent, in fact it dates back to the 80s, when this type of auction was used for the allocation of airport time slots. In fact, combinatorial auctions have gained relevance in the context of strategies for allocating resources, goods and services that are assigned or exchanged in customized combinations. Before going into the details of this topic and seeing some examples where they are used, let's look at a definition of these combinatorial auctions. They can be defined as auctions in which individual items are offered in bundles and auction participants offer prices for the entire bundle. In case of victory, all the items contained in it are assigned to the winner, while in case of loss, nothing is received. This type of auction is therefore much more complicated than traditional ones in which only one item is offered for auction, as the value of the bundle may not simply be the sum of its elements. Think about this simple example. If the auction concerns the assignment of advertising slots on a website (for example 5 slots), obtaining 5 adjacent slots could have a higher value in terms of visibility than having the same number of slots but scattered throughout the site. This illustrates how selling a package of 5 adjacent slots together as a bundle can be of greater value to the buyer than buying 5 slots separately. In this sense, combinatorial auctions lead us towards the topic of this thesis, bundle pricing. In fact, combinatorial auctions are nothing more than traditional auctions that exploit the concept of bundles to sell multiple products at the same time.
This type of strategy in auctions allows participants to submit bids for sets of objects, allowing them to express preferences and interdependencies between the objects themselves. This feature makes combinatorial auctions perfect for allocation/assignment problems where items have a complementary value or have synergies with each other. The areas in which combinatorial auctions are

mostly used are therefore the airport and naval sector, the assignment of communication frequencies, the allocation of advertising slots, the transport sector and in general in the economic and industrial fields. In fact, some logistics consulting companies offer software that allows trucking companies to bid on lane bundles. Billions of dollars are paid by large companies like Ford, Wal-Mart and K-Mart in contracts like this.
Despite the great opportunities that this topic has and its numerous practices, combinatorial auctions also have critical issues from a computational point of view. In fact, one of the most complicated challenges is to determine the best combination of items to insert within the same bundle. This problem is of fundamental importance for what has been said before, i.e., the existence of complementarities and synergies between the items. An incorrect combination could significantly reduce the perceived value of the bundle and cause serious losses to the company offering the products for auction.
In the literature it is possible to find many types of combinatorial auctions, which vary from each other even for marginal aspects related to the rules of the auction or the type of customer. For example, we may have single-minded bidders, i.e., interested in only one bundle offered by the company, or there may be auctions in which participants are interested in multiple bundles. There are also multi-units combinatorial auctions (MUCA), in which customers are interested in buying more units of the resources offered. In this case a further complication is added, i.e., customers must also indicate the desired quantity of each combination offered. Some examples in which the type of auction varies according to the rules of the auction itself are the simultaneous multiple round auction (SMRA), in which the participants present their bids simultaneously in different parts of the auction called rounds, and the auction closes only when nobody makes a new bid. Another example is generalized vickrey auctions (GVAs), which are designed to convince participants to reveal their true valuation of auction items. This typology is used in the sale of travel packages, where participants bid based on their actual evaluation of individual travel packages.
To better understand combinatorial auctions, a real case will now be illustrated from the paper "Combinatorial Auctions in the Procurement of Transportation Services" published by Yossi Sheffi in 2004, in which the problem of transport services is analyzed and a solution that uses combinatorial auctions is proposed.

In the transport sector there is usually a carrier who takes care of putting the shipper (i.e., the manufacturer/distributors) and the customer in contact. The carrier is usually a trucking company (transport by road, ship, plane or rail). Transportation services are purchased using an RFP (request for proposal) as is often the case for traditional goods. However, the process for transport differs from that for traditional goods because the cost is more influenced by economies of scale and scope. Let's see how this process works to better understand the concept. When the shipper has to transport something between an origin and a destination, he asks the carrier to use a transport line (for example 10 loads from Turin to Milan) and pays a cost for transportation. The problem arises in certain cases when the shipper intends to increase the number of loads, for example going from 10 to 20 (again from Turin to Milan). In fact, the shipper will not expect to pay exactly double, but to receive a discount due to the greater number of loads. However, in some cases it could not only get a negative response from the carrier, but also a higher price. This happens because a greater number of loads in the same direction could complicate the logistics of the carrier, for example it could find itself with too many trucks and drivers in Milan, which would force it to send these empty trucks to pick up a new load elsewhere, instead of moving from Milan already with a new load. From the point of view of the carrier, the ideal case would be to get those 10 additional loads in the opposite direction (from Milan to Turin), this would allow him to balance vehicles and drivers and reduce the cost. So, in this example, the cost of transport on a route depends not only on the number of loads on that route (economy of scale), but also on those on other routes (economy of scope).
One of the possible solutions for the shipper is to use the combinatorial auction model. This means, in the real case, asking the carrier to make an offer, not only for individual routes, but also for packages of different routes. The carrier will make offers on the routes most advantageous to him, based on his economy, his customers or the domicile of the drivers. The idea is to convince the carrier to create packages that allow him to reduce costs, so that this cost reduction can also be partially passed to the shipper. While in a traditional auction the number of bids is always equal to the number of voyages, combinatorial auctions also offer the possibility of pricing packages of routes, increasing the potential bid for shippers.

Before moving on to the next chapter on bundle pricing, it is very important to understand the difference between combinatorial auctions and bundle pricing, as in some cases they can look very similar. This difference can be explained from two points of view: first of all, the nature of the offer, in fact in bundle pricing packages are offered to the customer at a fixed price decided by the seller, instead in combinatorial auctions the packages are sold in customized combinations through an auction process. The second difference, which is directly connected to the first one, is the pricing decision process. In bundle pricing, as just explained, the problem lies precisely in deciding the price of the bundles to offer to the customer, while in combinatorial auctions it is deciding which packages to sell and the buyers to whom the packages should be sold, then, the price will be determined by the auction mechanism. The latter concept leads to the definition of the so-called winner determination problem (WDP) which plays a fundamental role in combinatorial auctions. In fact, in this type of auction, the process of determining the winner is made more complicated by the nature of these auctions, in which bundles of products are purchased (and not single items). The term winner determination problem refers to the process of determining the best combinations of items to sell and assigning them to the various participants. This allocation must be made with the aim of maximizing the total value of the allocation itself, while respecting the constraints of the auction. The WDP represents a though challenge for the seller from a computational point of view, as the number of possible combinations grows exponentially with the number of products offered and bidders. This requires the application of algorithms to solve linear programming models, mixed-integer programming and, in some cases, also the use of heuristics.

4. Bundle pricing

4.1 Introduction to bundle pricing

To explain what bundle pricing is, the best thing is to start with a very simple case, the solution of which is obvious, but it helps to understand this type of problem. Suppose we are a seller who has a single product to sell (for simplicity we assume that he has an infinite quantity available, sufficient to satisfy any demand) and that there are 3 customers interested in buying the product. These 3 potential buyers have different valuations and reservation prices for the same product: customers 1,2 and 3 are willing to pay 5, 10 and 40 euros respectively to purchase the seller's product. What is the price that maximizes the seller's profit? If the seller decides to sell the product for €5, this would allow him to sell to all customers, with a total profit of €15. If, on the other hand, the seller decided to sell only to customer 3, setting a price of 40, he would sell only one product, but the profit would still be higher. This is therefore the strategy that should be used to maximize the seller's profit. In this case, as mentioned above, the solution was obvious, but in a more general case where there are M products for sale and N customers interested in buying them, with different interests and reservation prices, things become more complicated. In this case it becomes convenient to use more complex strategies. In fact, when there are more products, customers may be interested and have an evaluation only for some subsets of products (bundles), so to maximize profits the seller will have to understand what is the best price for that bundle. We can therefore define bundle pricing as a pricing strategy in which customers request bundles of product, i.e., subsets of products, instead of individual products and the seller must decide the price of them. This approach of creating bundles can convince customers to buy more products, which they would not have purchased individually, providing buyers with an incentive which can be a discount on the bundle price (therefore the bundle price will be lower than the sum of its components) or, for example, offer

complementary products, i.e., products that increase customer satisfaction when purchased together rather than separately.
Bundle pricing is not only a theoretical topic but also has several important practical applications, and in fact it is used by companies in many sectors: in the tourism sector, travel agencies often offer packages that include, travel, hotel, breakfast, guided tours, various experiences, etc. In the IT sector, companies tend to offer product packages to give customers a 360-degree experience (think, for example, of the Microsoft Office package with Word, Excel, PowerPoint etc.). In the catering industry, menus proposed by the chef that include several courses, food or wine tastings, etc. are now common. Or again, in the telecommunications and media sector, where packages are offered that include subscriptions to newspapers and magazines, or streaming platforms can offer packages that include various services. We could go on to list many other examples of companies that use bundle pricing in their sectors, first of all one of the most important companies in the world, Amazon, which with its Amazon Prime subscription offers the customer various services such as fast deliveries, discounts on products, the Prime Video platform for movies, e-books, etc.
Many companies have started using this pricing strategy because there are many benefits: increase in overall sales and revenues, because selling in bundles can convince customers to spend more and buy more products than if they buy the products separately, reduce of complexity, because instead of selling each product individually, the company can offer a single package with multiple services included, increasing the value perceived by the customer, especially if the packages contain complementary products.
When companies deal with pricing bundles instead of single products, the problem is also complicated by another important factor, the choice of which products to bundle. In fact, we can distinguish 3 types of products: complementary, substitutes and independent. By placing these products in the same bundle, it can lead to very different results in terms of sales and profits. For example, if a company creates a bundle with complementary products (e.g., hot dog + ketchup and mayonnaise, or a wine with two glasses included), the customer will have an evaluation for the bundle higher than the sum of the evaluations of the single products contained in it. If the company bundles substitute products, the bundle valuation will be lower than the sum of the

valuations for the individual items, while it will be the same for independent products. To summarize this concept, we can use a simple formula contained in the paper "A simulation-based approach to price optimization of the mixed bundling problem with capacity constraints" (Mayer, Klein, Seiermann - 2013). In this article the so-called degree of contingency θ is defined as follows:

$$\theta = (V_{i0} - \sum_{j=1}^{J} V_{ij}) / \sum_{j=1}^{J} Vij$$

Where V_{i0} is the valuation of customer I for the bundle and V_{ij} is the valuation of customer I for product j. If $\theta > 0$, the products in the bundle are complementary, if $\theta < 0$, products are substitutes and if $\theta = 0$, products are independents.
Talking about of bundle pricing, it is interesting to define the so-called component pricing (or pure components), a pricing strategy that represents a useful tool to understand how to set the bundle price. With component pricing we mean the strategy of determining the optimal prices for individual products, without grouping them in a bundle, but selling it separately. So, the company sets a price for each of the products it sells, trying to maximize its profits. A variant of this typology is the so-called uniform pricing, in which a uniform price is set for all products (this can be advantageous for certain types of companies as it greatly simplifies the price structure). In reality, there is no pricing strategy that is always better than the others, in fact this depends on various factors, constraints and assumptions that are made when considering individual problems. As explained earlier, component pricing can also be useful for determining the price of a bundle. Let's imagine to have 3 products to be included in a bundle (we consider three independent products for simplicity), and we need to establish the price of the bundle. If we know that the price of the 3 components is respectively 40,50,60 euros, we can say the price of this bundle will have to be less than 150 euros (40+50+60). In fact, it is a common practice to offer the customer the so-called bundle discount, i.e., the difference between the cost of the bundle and the sum of the prices of the products that make it up. In the simple example just described, we have assumed that we already know the prices of the 3 products, but in a real case the optimal prices would have to be determined and one method for doing so is component pricing. For this reason, this pricing strategy can also be used as a support tool for determining the price of bundles.

After this first introduction on bundle pricing, it is important to say that this type of problem is made up of other subcategories. In fact, if we think about a multi-product company that has to decide how to sell its products, the possible strategies are a lot: it could sell the products separately, it could group all the products in a certain number of bundles, it could allow the customer to buy both individual products and bundles. But that's not all, once you decide to sell through bundles, you could determine the prices of it based on the content, or offer a fixed price for a certain number of products, or even establish a price for each possible size of the bundle. All of these variants of the same problem are actually different bundling problems. To clarify this concept, all these types of problems and some possible taxonomies to classify them will be illustrated in the following pages.

4.2 Taxonomies

In this part we will list some taxonomies to classify the bundle pricing problems available in the literature. This work is important to have a clear view of the possible types of problems and understand the differences. Note that the taxonomies that will be presented are not to be considered exhaustive, as they are only some of the possible classifications for these problems. We therefore begin to list the types of problems that have been identified and that will be reported in the following figures, explaining their main characteristics very briefly (a detailed explanation can be found in the next chapter):
Pure bundling: it is a pricing strategy in which the company offers the customer the possibility of buying only product bundles, not individual products separately;
Mixed bundling: unlike pure bundling, the company allows the customer to purchase both single products and bundles;
Bundle size pricing: the company sets a price for every possible bundle size, regardless the content;
Customized pricing: the customer has the possibility to buy a certain number of products at a fixed price;

Personalized bundle pricing: the customer composes his own bundle and asks the company to offer him a price. the customer decides whether or not to accept the offer.
Once the types of problems have been described, we can move on to the taxonomies that have been chosen. Note that when describing some taxonomies, some type of problem could be "border-line", i.e., be somewhere between two categories based on the constraints and characteristics that are taken into consideration (in fact, for example, not all pure bundling problems are identical, but each of them can have its own particularity).

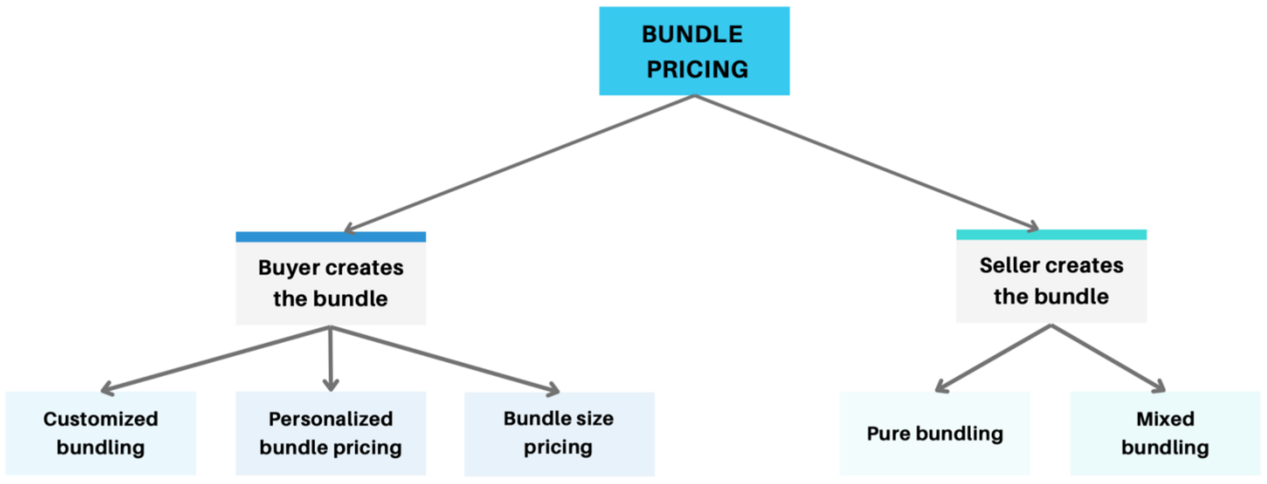


Figure 3: Taxonomy 1

The first taxonomy is based on the question "is bundle created by the buyer or by the seller?". This means that in some types of problems (customized bundling, personalized bundle pricing and bundle size pricing) it is the buyer who composes his bundle, deciding which products to include in it, while in pure bundling and mixed bundling it is the company to offer packages that the customer can accept or refuse (although in mixed bundling all possible combinations of bundles are offered, so the customer's choice is as wide as possible).

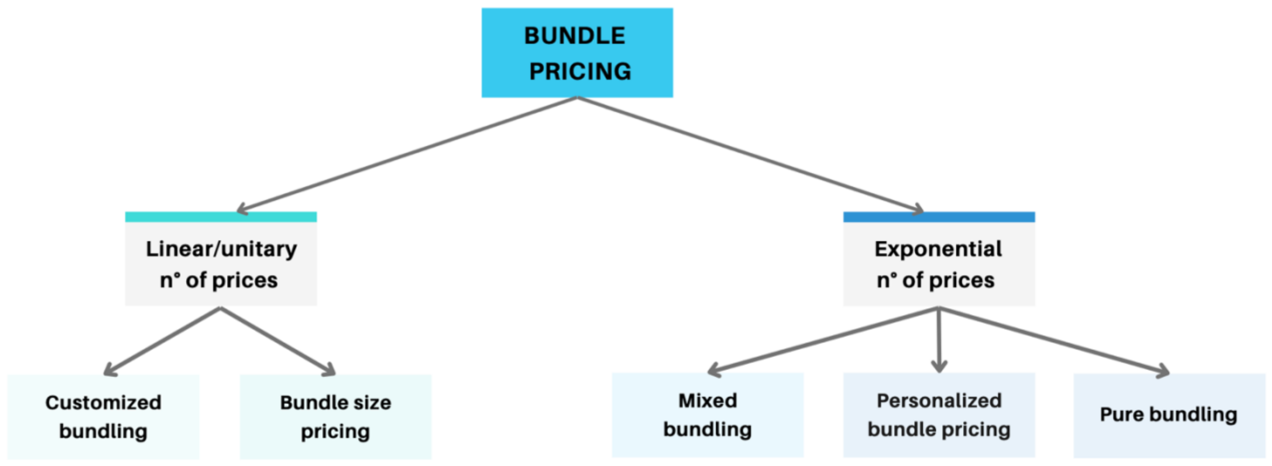


Figure 4: Taxonomy 2

The second proposed taxonomy concerns the complexity of the price structure. This discussion will be explored in the next chapter, but to give a sufficient idea to understand the figure above, we can think that if a company uses the mixed bundling strategy, it will have to establish a price for each possible purchase combination (even for products purchased separately), while if a customized bundling strategy is used, the price to be decided will only be one (or a maximum of N prices, in case the seller can offer even more than one bundle, but anyway a linear complexity). If, on the other hand, bundle size pricing is used, the prices to be fixed will be N (number of products for sale).

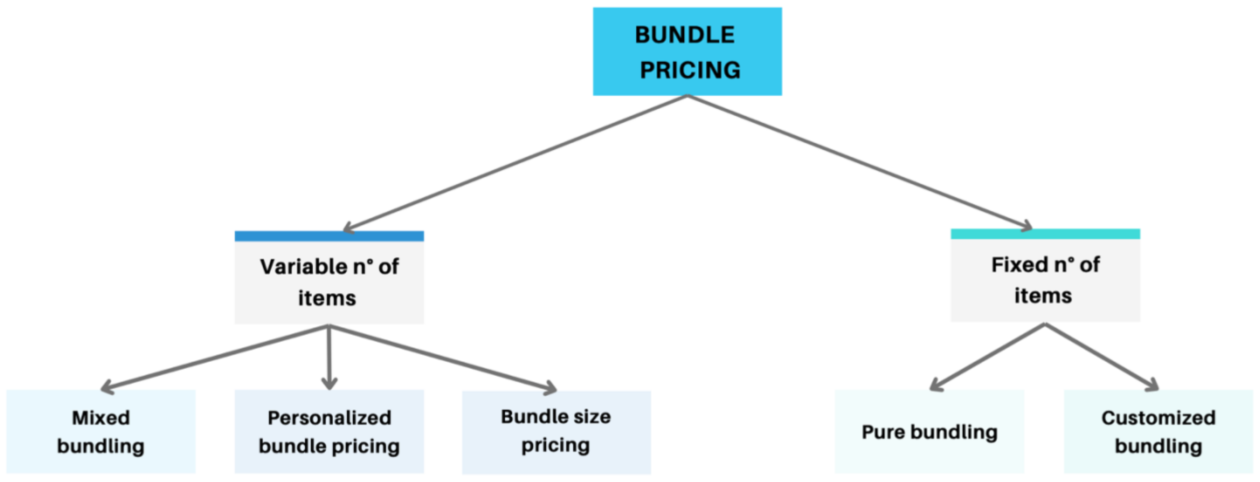


Figure 5: Taxonomy 3

In this case the criterion with which the distinction was made is whether the number of products that can be purchased by the customer is variable (and decided by the customer himself) or fixed and established by the company (please note that we are talking exclusively about the number of products, not which products are included in the bundle). In fact, in pure bundling and customized bundling, the company decides how many products to include in the bundle, and the customer can only accept or refuse. Instead, for example in personalized bundle pricing, it is the customer who composes his own bundle and therefore also decides how many products to include. Bundle size pricing, on the other hand, has been inserted in the left branch because, although the company decides the price for each bundle size, it is the customer who decides the size of the bundle he wants to buy. Mixed bundling, in the same way, is also on the left because the customer can choose from all possible combinations, so he can also decide how many products to buy.

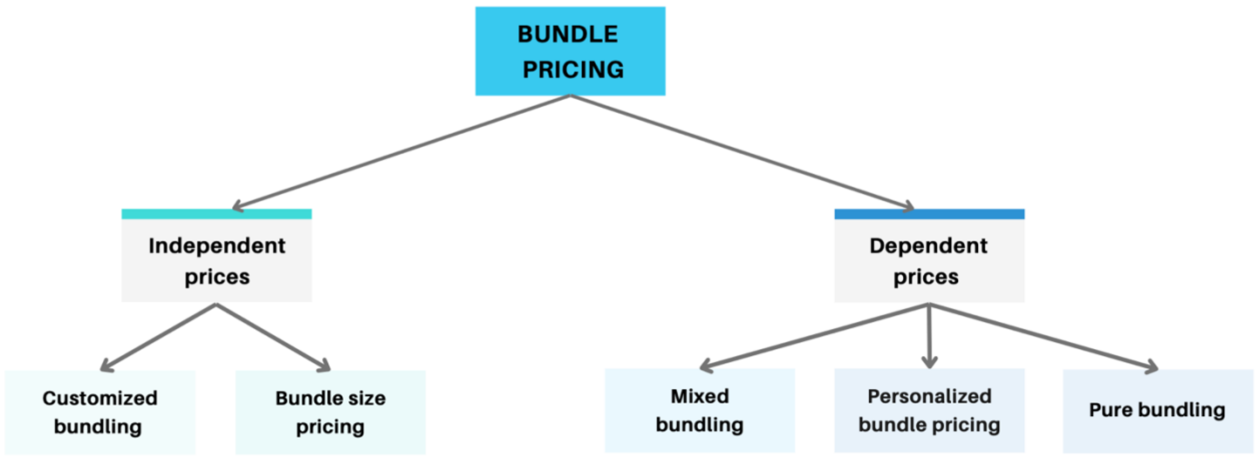


Figure 6: Taxonomy 4

The last taxonomy proposed is the one shown above, which divides the problems into two types: dependent and independent prices. In this case we refer to the dependence between the price of the bundle and the items included within it. In fact, in bundle size pricing and customized bundling, the company sets a price that depends only on the quantity of products included in the bundle, and not on its content. On the other hand, in right-branch problems, the price of the bundle depends on the content. In other words, it is possible to find two bundles of the same size, but with different prices, among the company's offers.

5. Bundle pricing problems

In this chapter we will describe and analyze all the types of problems that have been mentioned in the previous chapter and then explain the main differences using some examples for a better understanding of the topic. The problems that will be described are therefore: pure bundling, mixed bundling, bundle size pricing, personalized bundle pricing and customized bundling.

5.1 Pure bundling

Imagine you are a company and you want to sell your products in bundle. The simplest method would be to create a unique bundle containing all products and allow the customer to purchase only that bundle and not individual products. However, this strategy may only make sense in certain types of companies, for example companies that offer a streaming service (and therefore sell all their films in a package for a monthly fee), but in other sectors this can be a problem because it leaves little flexibility to customers. An alternative can be still selling product bundles (and not individual items), but creating several possible packages, this increases the customer's choice and can lead to a better profit in those companies where there are many products and they cannot be all placed in the same package (it is important to note that many of the cases available in the literature define pure bundling only as cases in which all the products are included in the unique available bundle. However, looking at the definition of pure bundling, the case where multiple bundles are offered to customers is not excluded). Somebody might think that pure bundling is a damage for the customer and that this will drive him away from the company, in search for another seller who allows him more freedom of choice. In reality, pure bundling has proved to be very effective in some cases and has led to very positive results. For example, it has encouraged the promotion of less popular products by including them in bundles of successful items, stimulated the purchase of multiple products in one transaction, and increased revenues as customers are forced to buy the whole bundle instead of single products. However, as just mentioned, some customers may prefer the flexibility and go to vendors who only let you buy what you want.

Regarding this difference in customer preferences, we can mention the paper "Pure Components versus Pure Bundling in a Marketing Channel" (Girju, Prasad, Ratchford - 2013), which analyzes a very similar case. As explained in the article, often in a manufacturer - retailer system there can be different interests, for example in a multichannel video programming market or in the digital music industry it happens that the manufacturer would like to sell in bundles, while the retailer is interested in buying of single items (think of a music album. The manufacturer, for example Sony, would like to sell the entire album, while the retailer, for example Apple Music, is only interested in some tracks). What the paper intends to do is to calculate the profits of retailer and manufacturer in case of both pure component and pure bundling strategies, by considering a Stackelberg game in which the manufacturer acts as the leader, while the retailer as the follower, with the aim of finding a price equilibrium that maximizes the profits of both. We assume that we have only two products and that the manufacturer sells to the retailer at a wholesale price of W_a for product A, W_b for product B and W_{ab} for the bundle. In turn, the retailer will sell the products to the final customer with the prices P_a, P_b and P_{ab} (increased compared to the wholesale prices). Furthermore, it is assumed that customers are heterogeneous with different reservation prices, in particular they belong to two segments: H and L. The first one is represented by those willing to pay V_H for a product, while segment L by those willing to pay V_L (with $V_H > V_L > 0$). It can immediately be seen that the presence of $V_H V_H$ and $V_L V_L$ customers reduce the attractiveness of bundle sales. Without going into details, the paper analytically calculates the optimality conditions of the retailer and therefore the optimal prices for the two products or for the bundle given the wholesale price of the manufacturer. Also for the latter, the optimality conditions and the optimal wholesale price are then calculated to induce the retailer to choose a specific strategy. These results are finally compared to find an equilibrium. As expected, there is no strategy that is

always better than the other. For example, from the point of view of the retailer, the optimal price (and therefore the optimal strategy) always depends on the wholesale price set by the manufacturer. In fact, if the wholesale price falls within a certain range, the best strategy is pure bundling, but if the wholesale price changes, the pure components strategy can also become optimal.
In the example just described, the problem was analyzed from the point of view of the manufacturer and the retailer. Another interesting way of looking at the pure bundling problem is to analyze it from the point of view of the end customer. For this purpose, we cite the paper "Transportation service bundling for whose benefit? Consumer valuation of pure bundling in the passenger transportation market" (Guidon, Wicki, Bernauer, Axhausen - 2020), in which the willingness to pay (WTP) of the customer is estimated final facing the possibility of using a package made up of transportation services, such as car sharing, taxi, bike sharing, etc. (the so-called MaaS, mobility as a service). To do this, a survey proposed to some citizens of Zurich (Switzerland) was used, in which various options were proposed, both for services alone (e.g., choose between 3h of bike sharing at a cost of 20 or 10h at a cost of 100), and for service packages (e.g., choose between a package with 5h of car sharing and 3h of bike sharing at the cost of 50 and another package with 4h of car sharing and 5h of bike sharing at the same price). To understand the results, a mixed logit model was used, i.e., a statistical model that makes it possible to analyze consumer behavior when they face multiple options, and unlike the classic logit model, takes into account the heterogeneity of consumer preferences, which may also vary over time. The result of this study shows that, although some services are valued more when taken individually, overall, the sum of the service valuations is higher in the case of bundling. Furthermore, even some services such as car sharing are valued more by customers when included in a transport service bundle.
To conclude this part on pure bundling and introduce the next type of problem, mixed bundling, it is necessary to think about the complexity (with complexity we mean the number of prices that the seller must set if he chooses a specific sales strategy). In fact, pure bundling has the advantage of simplifying the company's price structure, but still has an exponential complexity equal to $2^N - N - 1$ (even if this number is theoretical, since the seller can decide to make even one bundle, in this case the complexity is reduced to 1), i.e., the seller must set a price for

each combination of products (2^N), but without doing it for single items ($-N$). The -1 in the formula is used to exclude from the count the case of an "empty bundle", i.e., the possibility of not selling anything to the customer. An alternative is mixed bundling, whose complexity is always exponential, but which guarantees greater freedom of choice for the customer.

5.2 Mixed bundling

Mixed bundling is a pricing strategy similar to pure bundling but differs because it allows the customer to not only buy bundles of products, but also individual products separately. In other words, it is the pricing strategy that ensures greater freedom of choice for the customer. Consequently, it is also the type of problem that has the greatest complexity, in fact the number of prices that the seller must determine is equal to all possible combinations of products, also considering the possibility of purchasing single products, therefore $2^N - 1$. The goal of mixed bundling is to guarantee the customer the greatest possible flexibility and thus attract a large number of customers in order to maximize profits. The decision between adopting a pure bundling or mixed bundling strategy is not easy and depends on several factors, including the sector, the company considered, the products for sale, the elasticity of demand and the customers. In fact, a mixed bundling strategy could be more profitable in a case where customers are heterogeneous and have different preferences regarding the company's products. On the other hand, pure bundling, as we have seen, can be more profitable if you try to push the customer to buy unpopular products by placing them in bundles with successful products.
One area in which mixed bundling is used is reported among some examples in the paper "Mixed Bundling of Two Independently Valued Goods" (Bhargava - 2013), and is that of camera kits. think, for example, about the action cams used in sports to film the sporting activity from your point of view. These cameras have different supports, for example to fix it to the helmet or on the handlebar for those who ride motorcycles, a bib for those who run, the famous "selfie stick" etc. A customer may not be interested in all of these tools, but only in some combination

of them. In this case mixed bundling could be a preferable strategy to pure bundling. This paper therefore analyzes the case of a classic mixed bundling problem with two independent products for sale and proposes an analytical solution. It is assumed that the two products have marginal costs equal to w_1 and w_2 and that customers are heterogeneous in their evaluations and that these evaluations follow the cumulative distribution function F_1 and F_2 over the support $[0; a_1]$ and $[0; a_2]$ (with $a_2 \geq a_1$). The objective function to be maximized is therefore:

$$\max_{p_1, p_2, p_b} \pi = (p_1 - w_1) \cdot Q_1 + (p_2 - w_2) \cdot Q_2 + (p_b - w_1 - w_2) \cdot Q_b$$

where Q_1, Q_2, Q_b represent the sales levels for each product and the bundle and p_1, p_2 and p_b represent the prices of the products and the bundle. Without going into details, in this paper the optimization problem with three variables (p_1, p_2 and p_b) is transformed into a single variable problem (p_b) by writing p_1 and p_2 as a function of it, thus determining the optimal prices of the products and the bundle. Another interesting case concerning mixed bundling can be found in the paper "A simulation-based approach to price optimization of the mixed bundling problem with capacity constraints" (Mayer, Klein, Seiermann - 2013), already mentioned in chapter 3.1 talking about products complementary, substitute and independent. This article lists other areas in which mixed bundling is used, with particular reference to the automotive sector. In fact, when a customer wants to buy a car, in addition to the base version of the vehicle, he is offered a series of optional to choose from, which can be purchased individually or in packages. The article analyzes a general case of mixed bundling, but compared to the previous example, an additional element is added that complicates the problem, the capacity constraints. In fact, it is considered that there is a limited supply for each product on sale, and that customers arrive sequentially, so they can only buy the product if it is still available. The initial capacity of product j produced is indicated with c_{0j}, therefore the capacity of the bundle containing all products j (with $j = 1 \ldots J$) will be $c_{00} = \min_{j=1 \ldots J} \{c_{0j}\}$. Apart from this, the problem is similar to those of traditional mixed bundling, and is therefore solved by setting an objective function for maximizing the firm's profits, subject to various constraints (in this case 15). Therefore, we have a MILP (mixed-integer linear programming) formulation, i.e., an optimization problem in which the objective function and the constraints are linear and the variables can assume both integer and real values (basically a more general case of the linear programming problems). To solve this problem, the optimization software CPLEX was used and, in a second part of the paper, a method with metaheuristics is also examined.

5.3 Bundle size pricing

After talking about pure bundling and mixed bundling it might seem that there is no need for anything else, because all the combinations that can be offered to the customer are already contained in the mixed bundling and if the company wants to sell only certain bundles it can use pure bundling. In reality, however, as can be seen from Figure 4, these two pricing strategies have a common problem, the complexity of the pricing structure. Mixed bundling in particular has a complexity of 2^N and even pure bundling, if all possible combinations of products are considered, has a number of prices to fix that grows exponentially with the number of products. This problem can be significant for companies that have a large number of items for sale and for this reason there are strategies that aim to simplify the pricing structure. One of them is bundle size pricing.
Bundle size pricing is a pricing strategy used by companies where packages of products or services are offered at a price that is based only on the size of the bundle, i.e., the bundle price varies based on the number of items included in it. If a company therefore offers 20 products to its customers, it will have to decide the price for the unit bundle (with only one product), another price for a bundle consisting of two products, and so on. Returning to the discussion of complexity, it is easy to understand that now the number of prices that the company must set is no longer exponential, but linear and equal to the number of products on sale. This allows the company to simplify the cost structure and the customer to create a more personalized package, maximizing the perceived value. Bundle size pricing is therefore proposed as a middle ground between the two strategies just seen.
Some examples of bundle size pricing are presented in the article "Convex Optimization for Bundle Size Pricing Problem" (Li, Sun, Teo - 2022). This paper

refers to some areas where this pricing strategy is used, such as cable TV companies that offer customers combinations of channels, or the Netflix platform that offers users different plans based on the number of screens they want (e.g., 3 screens means that the same account can be used on 3 different screens at the same time). Furthermore, a bundle size pricing problem is also tackled in which a company sells n products to an audience of customers who have a valuation u_i for each product i ($\tilde{u} = \{\tilde{u}_1, \dots, \tilde{u}_n\}$ is the vector of valuations sorted in increasing order and follows a given distribution F). For each bundle of size s a price p_s is fixed which depends exclusively on the size of the bundle. for simplicity we assume only one type of customer and define a vector $\widetilde{W}_s(\tilde{u}) = \sum_{k=0}^{s-1}(\tilde{u}_{n-k})$, which represents the maximum valuation for each bundle size and follows a joint distribution G. The goal is to maximize the following objective function which represents the firm's profit: $\max\limits_{p \geq 0} \sum_s (p_s - c_s) \cdot q_s(p)$. where c_s is the cost of the s-size bundle and $q_s(p)$ is the expected demand for a bundle of size s sold at price p. This is therefore a standard optimization problem in which however the difficult part is determining the expected demand and also characterizing the G distribution.
A second paper, entitled "Bundle-Size Pricing as an Approximation to Mixed Bundling" (Chu, Leslie, Sorensen - 2011), addresses the same issue but from a different point of view. Recalling that one of the advantages of bundle size pricing is the reduction of the complexity of the price structure, he explains how, under certain conditions, the BSP can be considered an approximation of mixed bundling. This result is important because mixed bundling is often considered the pricing strategy that maximizes profits (thanks to the high level of freedom given to the buyers), but if bundle size pricing can be considered a good approximation, this means that by using this strategy the company can reduce complexity without decreasing profits significantly. In order to do this, the article proposes the case of a company that sells two independent products, for which customers have a certain evaluation $v_1 \sim U[0, \theta]$ for product 1 and $v_2 \sim U[0,1]$ for product 2 (with $\theta > 1$). Since some optimization models have already been shown in the previously described papers, the model of this problem is not reported in order not to complicate this document. Anyway, after several passages, the authors arrive at defining the optimal prices for each pricing strategy considered (pure

bundling, component pricing, mixed bundling and bundle size pricing), with the consequent profit for the company. These prices and profits are expressed as a function of θ, so as this parameter varies, we will have different results. Assuming $\theta = 1.7$, it is interesting to compare the optimal prices of mixed bundling and bundle size pricing: in the first case, prices obtained for product 1, product 2 and for the bundle are 1.13, 0.67 and 1.18 respectively. For the BSP, on the other hand, the price for the unitary bundle is 0.9, while for the bundle with two products it is 1.1. Unsurprisingly, the unit bundle price of the BSP is between the two individual product prices, while the price for the complete bundle is similar in both cases. By assuming the values of v_1 and v_2, other interesting observations can be made, such as for example how many and which products customers would buy if a specific pricing strategy were used.

5.4 Personalized bundle pricing

The first three strategies described, pure bundling, mixed bundling and bundle size pricing, are certainly the most used and known. However, there are others that can have important application fields. One of these is relatively recent and has been studied in an article entitled "Pricing personalized bundles: A new approach and an empirical study" (Xue, Wang, Ettl - 2016) and is called personalized bundle pricing. In this type of problem, a seller offers his customers a certain number of products and allows individual customers to create a personalized bundle by sending a request for quote (RFQ). At this point the seller analyzes the customer's request and decides the price for that bundle. In a third stage, the customer can decide whether to accept the whole bundle or to reject it completely. This article is based on a real-life case faced by an information technology service provider, which initially used human resources to establish sales prices. The approach proposed by the authors is instead to find a method to solve this type of problem automatically, using historical data of the company and the customers. One of the examples mentioned in the paper is that of an IT company, such as Google, when it has to buy a certain number of computers from a manufacturer (e.g., Dell). In this case, the customer composes his bundle

and asks the seller for a price. This concept can be extended to all those companies that use an RFQ process when they have to sell their products.
The advantage of this strategy compared to, for example, mixed bundling, is that companies that have many products for sale do not necessarily have to set a price for each product and combination of them. In fact, if a company has even just 10 products, the complexity of mixed bundling is 2^{10}, i.e., 1024 prices. Among these combinations there will certainly be highly unlikely ones, which means combinations of products that no one will choose. Personalized bundle pricing helps to exclude these combinations and price only those requested by customers. However, even personalized bundle pricing problems have, at least in theory, an exponential complexity and the difficulty for companies is precisely that of managing all customer requests.
As anticipated, this type of problem is relatively new, in fact, as the authors explain in the literature review, there are no other articles in the literature that analyze this case of customers who build their own bundle starting from the products offered by the company. The only similar case is that of customized bundle pricing, in which the company offers a certain number of products at a fixed price, but this type will be analyzed in detail in the following pages.
The method used in this case is a two-step approach, characterized by a top-down step and a bottom-up step. In the first, the bundle chosen by the customer is decomposed into its components, which are then grouped according to their characteristics and similarities and are assigned a value score. Then in the bottom-up step, these component value scores are aggregated at the bundle level. The model of the problem is the following. I components are considered, classified in product families (e.g., in the case of an IT company, hardware, software, maintenance, etc.). each component has a series of attributes $a_{i1}, a_{i2}, \dots, a_{ik}$ (e.g., cost, list price, etc.). Customer's request can be seen as a vector $d = (d_1, \dots, d_I)$ where $d_i > 0$ means that the customer has included product i in the bundle. A vector z is then defined which represents the characteristics of the customer (purchase history, region, industry sector, loyalty, etc.). It is therefore now possible to define a maximization problem for the seller's profits using the following objective function: $\max_p (p - c(d)) \cdot q(d, z, p)$, where $c(d)$ is the cost to offer bundle d. obviously, knowing the demand $q(d, z, p)$ the

problem reduces to a single variable optimization problem, but what is difficult is to determine the demand using the historical data available. The two previously mentioned steps, bottom-up and top-down, are used to calculate the customer's utility function $u(d, z, p)$ and component value scores, respectively.

5.5 Customized bundling

The last category of bundle pricing problems is customized bundling, also called customized bundle pricing. In this case the company/seller offers N products to its customers and allows to select a subset (bundle) M of them (with $M < N$) at a fixed price p decided by the company. The content of the bundle, on the other hand, is at complete discretion of the customer. Think, for example, about a situation that everyone has seen at least once, when you enter a clothes shop and see a sign that says "3 t-shirts for 30 euros". This could be an example of customized bundling, as the company decides the number of t-shirts to include in the offer and the price at which to sell them, while the customer can choose which t-shirts to buy. Obviously, this type of strategy is used when the marginal costs of the products for sale are similar and not high, in fact it would not make sense in a company with very different products in terms of cost structure and prices. The owner of a large jewelry store with all kinds of items (expensive and cheap) should not make such an offer. The two most obvious advantages of customized bundling for customers and vendors are that, on the customer's side, freedom of choice is very high because everyone can choose the products they want and buy them at a fixed price, obviously respecting the number of products chosen by the company. This is very efficient when customers only value a few items, but the company doesn't know which ones, so a possible solution, if the marginal costs are similar, is to let the customer choose what they want. From the company's side, instead, the most evident advantage is the reduction of complexity, in fact this is one of the few cases of unitary complexity, i.e., the company only has to decide the fixed price p of the M products. It is also interesting to note that the extreme case of customized bundling, when $M = N$, is pure bundling, i.e., the company sells all the products at a fixed price. The

opposite case instead, in which $M = 1$, could be considered a case of uniform pricing, in which the products are sold at a uniform price, regardless of the product sold.
The article "Customized Bundle Pricing for Information Goods: A Nonlinear Mixed-Integer Programming Approach" (Wu, Hitt, Chen, Anandalingam - 2008) compares the result of some bundling strategies with customized bundle pricing, explaining in which cases this last approach can be efficient. In particular, the case of companies that sell information goods, such as newspapers, magazines, music, CDs, etc., is shown, underlining how traditional strategies (pure bundling, mixed bundling, etc.) are not very efficient, especially if the number of products sold is very high (e.g., iTunes which sells tens of millions of songs). The paper therefore analyzes the case of an information goods provider who intends to adopt a customized bundling strategy to sell its products. However, the approach used is quite complex. In fact, the company has the ability to sell multiple bundles of different sizes (and not just one-sized bundle). Let's consider the example of a company that sells CDs, with the approach used in the article it could sell a bundle of 3 CDs for 20 euros and then a bundle of 5 CDs for 30. So, the complexity of the problem is no longer unitary, but still linear. The interesting thing is that this type of problem can be found in several real cases and also underlines an apparent overlap with pure bundling: if in a pure bundling case the company is allowed to create multiple bundles, this could be similar to the case of customized bundling that we are considering. However, the difference lies in the fact that in customized bundling the price of the bundle depends only on the number of items included, while in pure bundling also on the content (which is chosen by the seller and not by the buyer). Also in this case, The problem is to maximize the seller's profits, with several constraints to be respected.
A second paper entitled "Bundling With Customer Self-Selection: A Simple Approach to Bundling Low-Marginal-Cost Goods" (Hitt, Chen - 2005) analyzes a similar case of customized bundling, in which a monopolist sells N information goods and wants to determine the optimal size of the customized bundle M and its price p. The size of this bundle is calculated as a percentage m of all the seller's products (hence $M = m \cdot N$). Furthermore, it is assumed that customers buy at most one unit of each product and that they want to buy a bundle

represented by the vector of binary variables $x = (x_1, \dots, x_j, \dots, x_N)$, where $x_j = 1$ if the customer wants to buy product j, 0 otherwise. There are different types of customers and α_i represents the percentage of customers of type i (with $\sum_{i=1}^{I} a_i = 1$). Each customer has a willingness to pay $W_i(x)$ for bundle x and the utility deriving from the consumption of this bundle can therefore be defined as $U_i\big(x, p(x)\big) = W_i(x) - p(x)$. As in the previous cases, this is a seller's profit maximization problem, where customers are of different types and the seller pays a cost $C(m)$ to offer a bundle of size m.

In the next chapter the type of problem discussed in the rest of the thesis, and for which some resolution algorithms will be proposed, will be illustrated in detail. This type of problem is called the single-minded bundle pricing problem.

6. Single-minded bundle pricing problem

Once the most common bundle pricing problems have been analyzed in details, it is possible to enter into the specific problem that will be discussed in this thesis. The paper that served as inspiration is quite recent and is titled "Models and algorithms for the product pricing with single-minded customers requesting bundles" (Bucarey, Elloumi, Labbè, Plein - 2021). In this article, published in Computers and Operations Research, a problem called the single-minded bundle pricing problem (SMBPP) is analyzed. Returning to the definitions provided in chapter 2 (paragraph 2.2), it is possible to understand this type of problem. In fact, single-minded refers to a problem in which customers are interested in purchasing only one bundle, for which they have a certain valuation and a certain budget, while they have no valuation for all the other bundles (in other words they are not willing to purchase a different bundle than the desired one).

6.1 Introduction to the problem

The single-minded bundle pricing problem has two categories of actors, a seller and several buyers. The first has a certain number of products for sale and his task is to determine the price of each of them in order to maximize his profits, knowing that each customer wants to purchase a certain bundle of products, which can contain a variable number of items (just one, but also all of them) and is willing to pay a maximum of a certain amount of money (i.e., his budget). Different customers requesting the same bundle might also have different budgets. This reflects reality, in fact for the same product/service people may be willing to pay different prices (think for example of a ticket for a football match, some more passionate fans may be willing to pay more than others less interested in that match). From the single-mindedness of customers comes a fundamental constraint of SMBPP, which is that customers will buy the desired bundle if and only if the price of it does not exceed their budget. In this model

therefore, we can say that customers intend to maximize a certain utility function which is represented by the difference between the budget and the price of the bundle. Some variants of the SMBPP are also found in the literature, for example it is possible to complicate the problem by inserting the concepts of limited supply and envy free allocation. In the case of limited supply, in addition to the price determination problem, a bundle assignment problem must also be introduced, because if the number of products is limited, it is possible that the desired products are not available for some customers. Or, if we allow customers to request more than one unit of a single bundle, customers' budgets would become a function of the number of bundles requested, further complicating the solution. In this article, and also in this thesis, we will consider a simplified variant, characterized by unlimited supply (remember that, if supply is unlimited, the allocation is always envy-free). One of the most common cases of unlimited supply is that of digital goods, which by definition are available in unlimited amount. Another possible approach is to consider a seller who sells not different products, but product features, and this is exactly the case discussed in this thesis. In fact, referring to the topics covered during the internship at Stellantis, we will consider a seller interested in selling not different cars, but the optional available on a single car model. Think for example when you go to buy a new car and the seller offers you different optional packages, for example a set of options that you can choose whether to purchase or not. Among these vehicle features you can be interested in some of them, but not in others (maybe you are interested in automatic air conditioning, but not in cruise control). The problem is exactly identical to the case seen so far, but instead of having products for sale we now have product features.
As already explained in chapter 3 (paragraph 3.3), the SMBPP has elements of similarity with combinatorial auctions, in which bidders request a certain bundle and their bid is the maximum price they are willing to pay for that bundle (i.e., their budget). The difference, however, lies in the fact that auctioneers do not have the task of giving a price to the items they sell, but of establishing an envy-free allocation for the products (available in limited quantities) that maximizes their revenues.
In this article, the formulation of which we will present in the next paragraph, they propose a linear and non-linear formulation for the SMBPP, also providing a

resolution model based on an exact method and mathematical optimization, which has never been done before in the literature, while in the next part of the thesis we will try to create an algorithm to solve this problem.

6.2 Problem statement and formulation

In this part, we will analyze in details the SMBPP of the paper "Models and algorithms for the product pricing with single-minded customers requesting bundles", already mentioned in the previous paragraph, and in particular the model will be defined with all its parameters, variables, constraints and the objective function to maximize. As already mentioned, we analyze the case of a seller who sells $N = \{1, \dots i, \dots n\}$ products to $M = \{1, \dots j, \dots m\}$ customers interested in purchasing a subset (bundle) of them. It is assumed that customers have a single-minded behavior with a reservation price (or budget) b for the desired bundle, therefore each customer j is completely represented by a bundle S_j and a budget b_j. S_j can be defined as follows:

$$S_j = \begin{Bmatrix} 1, \text{if product i} \in S_j \\ 0, otherwise \end{Bmatrix} \text{ for } i \in \text{N and } j \in \text{M}$$

As already mentioned, single-minded customers will buy bundle S_j if and only if the price of the bundle $P(S_j)$ is lower than their budget b_j, where the price of the bundle is simply defined as the sum of the prices of the products contained in it, i.e.: $P(S_j) = \sum_{i \in S_j} p_i$. The single-minded bundle pricing problem consists of determining the prices of all $i \in N$ products in order to maximize the seller's profits, which means: $\max\limits_{p \geq 0} \sum_{j \in M} Revenue_j(p)$.

To better understand these concepts just described in mathematical terms, it may be useful to take a simple example with a few customers and a few products. This example will be referred to later as "*Example 1*".

Let's consider the case of a seller who sells only 2 products (or product features) to 3 customers. The first customer wants to buy both products and has a budget of 2, the second only wants product 1 and has a budget of 3, while the third customer only wants the second product with a budget of 4. The problem data is summarized below.

$$S = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad b = (2 \quad 3 \quad 4)$$

As already explained, the price of a bundle is equal to the sum of the prices of the products contained in it, so for example the price of the bundle requested by customer 1 will be $P(S_1) = p_1 + p_2$. What we want to find are the prices of the two products that maximize the seller's revenues, which can be calculated as the sum of the revenues obtained from each customer, therefore $Revenues = R_1 + R_2 + R_3$ (with R_j = revenues obtained from customer j). In this case the solution is very simple and the problem can be solved immediately without doing any calculations. In fact, the optimal solution is to set $p_1 = 3$ and $p_2 = 4$. In this way customer 1 will not purchase the bundle because its price exceeds his budget. Instead, customers 2 and 3 will purchase their bundles at the price of 3 and 4 respectively. The total revenues of the seller will therefore be $0 + 3 + 4 = 7$.
We can now move on to the formulation of the problem and therefore to the definition of the constraints and the objective function. Following the procedure illustrated in the paper cited above, we will start from a mixed-integer non-linear programming formulation for this SMBPP and then linearize the non-linear terms of it obtaining a mixed-integer linear programming formulation. In order to do this, it is necessary to introduce a binary variable denoted by X_j which will be equal to 1 if customer j purchases the desired bundle and equal to 0 otherwise. At this point we can report the objective function and the constraints of the nonlinear problem, which will be explained in detail immediately afterwards.

$$\max_{p,x} \sum_{j \in M} \sum_{i \in S_j} p_i \cdot X_j \qquad (1a)$$

$$s.t. \quad X_j \cdot \left(\sum_{i \in S_j} p_i - b_j \right) \leq 0, \;\; j \in M \qquad (1b)$$

$$(1 - X_j) \cdot \left(\sum_{i \in S_j} p_i - b_j \right) \geq 0, \;\; j \in M \qquad (1c)$$

$$p_i \geq 0, \quad i \in N \quad (1d)$$
$$X_j \in \{0,1\}, \; j \in M \quad (1e)$$

The objective function $(1a)$ consists in maximizing the seller's total profits, calculated as the sum of the profits from all customers (as shown in example 1). Constraints $(1b)$ ensure that if the total price of the bundle requested by customer j is higher than the budget, then X_j must be equal to 0 (i.e., the customer does not buy the bundle). Constraint $(1c)$ establishes that when the budget of customer j is higher than the bundle price, then X_j must be equal to 1. Finally, constraints $(1d)$ and $(1e)$ ensure respectively that prices must be non-negative and that are binary variables.

To make the problem simpler it is possible to linearize it, in order to make it a mixed-integer linear programming formulation. To this end, we introduce a further element denoted by U_i which represents the upper bound of the price of product i, i.e., the maximum price of that product, therefore $U_i = \max_{j \in M}\{b_j : i \in S_j\}$. If we set a higher price, no customer will buy that product, as its price exceeds the budget. We can then define the upper bound for a certain bundle S_j as follows: $U(S) = \sum_{i \in S} U_i$. To linearize the problem, it is necessary to insert a further variable, which will be the variable that we are going to maximize in the objective function and represents the revenue obtained from a single customer j and is defined as follows: $r_j = P(S_j) \cdot X_j = \sum_{i \in S_j} p_i \cdot X_j, \quad j \in M$. Now all the elements necessary to define the constraints and the objective function of the linearized problem are present. We can now define the linearized constraints and objective function and we will call this model $M1$:

$$\max_{p,x,r} \sum_{j \in M} r_j \quad (2a)$$
$$s.t. \; r_j \leq b_j \cdot X_j, \; j \in M \quad (2b)$$
$$r_j \leq P(S_j), \; j \in M \quad (3c)$$
$$r_j \geq P(S_j) - U(S) \cdot (1 - X_j), \; j \in M \quad (2d)$$
$$r_j \geq 0, \quad j \in M \quad (2e)$$
$$p_i \geq 0, \quad i \in N \quad (2f)$$

$$X_j \in \{0,1\},\ \ j \in M \quad (2g)$$

Note that in this formulation the constraint $(2b)$ is nothing other than the linearization of the constraint $(1b)$ of the nonlinear formulation.
In the cited article, a further model is also reported which consists of a second linearization alternative which no longer uses the variable r_j but another variable s_{ij} defined as follows $s_{ij} = p_i \cdot X_j,\ \ j \in M,\ \ i \in S_j$. Below are the constraints and the objective function of this second model which will be called $M2$.

$$\max_{p,x,s} \sum_{j \in M} \sum_{i \in S_j} s_{ij} \quad (3a)$$

$$s.t. \ \sum_{i \in S_j} s_{ij} \le b_j \cdot X_j,\ \ j \in M \quad (3b)$$

$$s_{ij} \le p_i,\ \ i \in S_j,\ \ j \in M \quad (3c)$$

$$s_{ij} \ge p_i - U_i \cdot \left(1 - X_j\right),\ \ i \in S_j,\ \ j \in M \quad (3d)$$

$$s_{ij} \ge 0,\ \ i \in S_j,\ \ j \in M \quad (3e)$$

$$p_i \ge 0,\ \ \ i \in S_j \quad (3f)$$

$$X_j \in \{0,1\},\ \ j \in M \quad (3g)$$

After having defined the linear models in this chapter, in the next one a resolution algorithm for this type of problem will be proposed using Python as the programming language and Python MIP library, of which all the characteristics and functionality will also be described.

7. The algorithm

7.1 The solver: Python-MIP

Once the model of the problem has been defined and the constraints and objective function of it are understood, it is necessary to find an algorithm that allows finding a solution. There are many alternatives, we could write an algorithm from scratch starting for example from a particular solution (e.g., starting from all the prices at the maximum, gradually decreasing the prices until the optimal solution is found), but fortunately a tool, available for free, can be very useful, python-MIP. Python-MIP is a Python library, consisting of a collection of tools for modeling and solving combinatorial optimization problems including mixed-integer linear programming problems. In fact, MIP indicates a type of problem in which some variables of the model can assume integer values and others also continuous values. The areas in which this tool is used are not limited to what we are using it for, but it also has various applications in other fields such as logistics, production, financial, planning etc. The advantages of python-MIP are various, first of all the fact that it is very simple thanks to a user-friendly interface which is very easy to use and allows you to solve many types of problems quickly, for example the traveling salesman problem, 0/1 knapsack problem, job shop scheduling problem or, as in our case, bundle pricing problems. In general, it can be stated that python-MIP is a collection of python tools that uses powerful solvers to solve to find optimal or approximate (heuristic) solutions for some types of problems proposed by users and this makes it a fundamental tool for optimizers, operations researchers and anyone who has to face with real-world problems based on decision variables, constraints and objective functions. Going into more detail, python-MIP allows you to model a problem by inserting constraints, variables and objective functions of your problem in an intuitive way and with a simple syntax, and then using methods and functions to find a solution.

What is most interesting for this thesis, is how python-MIP can be applied to our specific problem, how the constraints, variables and objective function can be written and above all it is important to evaluate its efficiency in terms of resolution time and seller revenues, this last efficiency evaluation will be carried out in the next chapter. So, let's start by looking at some examples to understand how it is possible to insert all the elements of our model.
While the parameters of our model can be entered using the normal Python syntax (so for example to enter the number of customers it will be sufficient to write $M = a$), for the variables, constraints and the objective function there is a specific Python-MIP syntax. First you need to enable python-MIP in your python code, for this operation the following syntax is sufficient:

$$from\,mip\ import\ *$$

Then it is necessary to create the model and specify whether what we are trying to solve is a maximization or minimization problem, so in our case we can write:

$$m\ =\ Model(sense = MAXIMIZE)$$

As regards the model variables, the syntax is as follows:

$$y\ = [m.add\_var(var\_type(BINARY), lb = -10, ub = 10\ for\ i\ in\ range(5)\ ]$$

This code will insert 5 binary variables into the model, each of them between -10 (lower bound) and 10 (upper bound). Other types of variables can be, for example, INTEGER or CONTINOUS.
Instead, to introduce a constraint, we can simply write:

$$m \mathrel{+}= \ y + x <= 5$$

In this way the constraint inserted will be $y + x <= 5$. If instead we wanted to insert a summation expression we can write:

$$m{+}{=}\ xsum\big(w[i] * x[i]\ for\ i\ in\ range\ (n)\big) \leq c$$

This expression is the equivalent of $\sum_{i=0}^{n} w_i \cdot x_i \leq c$.
Finally, to insert a maximization objective function the syntax is as follows:

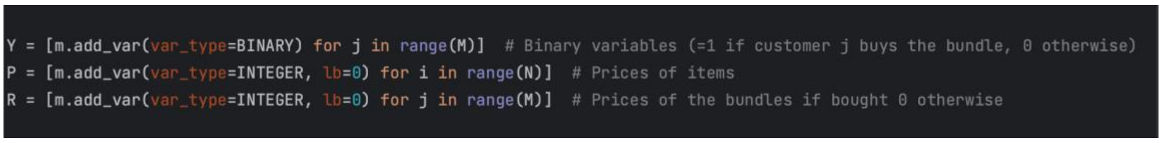
$$m.objective\ =\ maximize(*\ expression\ to\ maximize\ *)$$

The next paragraph will introduce in detail the algorithm used and the instances created to test it.

7.2 Model code and creation of instances

At this point we can return to the topics covered in the internship to understand how this algorithm can be used in a real case, in particular the context in which RAM and Dodge brands operate, and create instances that simulate one of the problems that they have to face on a daily basis in their relationship with their customers. Let's imagine that some traders make a request for a particular car (all the same for simplicity, for example a RAM 1500 Limited) and have to choose which optional to order for the chosen car. Going to the RAM website and selecting the "Limited" version you will notice that there are several packages available, each of which consists of a subset of all the optional available for this specific version. For example, if a trader wanted to order a limited night, this would mean that of all the optional available, he would like to purchase only those included in the night package (black grille badge, 22-inch wheel, 19 Speaker Premium Sound system, etc.). This, in the model illustrated in the previous chapters, can be considered as the bundle chosen by that particular trader. In fact, another trader might still want to order a RAM 1500 Limited, but with a different optional package, for example the level 1 equipment group (which includes options such as adaptive cruise, head-up display, lane keep assist etc.). In this case we would have two traders who want the same product but with different optional, i.e., two different columns of the S matrix introduced in chapter 6. To clarify this concept even better, let's return to example 1 of chapter 6. In this case the matrix S and the budgets b can be interpreted as follows: The seller (e.g., RAM) offers two optional for its pick-ups and there are 3 customers interested in purchasing them: the first wants the vehicle with both optional and is willing to pay 2 (euros/dollars), the second only wants the first optional at a maximum price of 3, while the third only wants the second optional at a maximum price of 4.

Now we can see how the model variables, constraints and objective function have been set. In this chapter only what has just been mentioned will be shown, while the complete code will be included in the appendix. As regards the variables, the code is the one shown in the figure below.

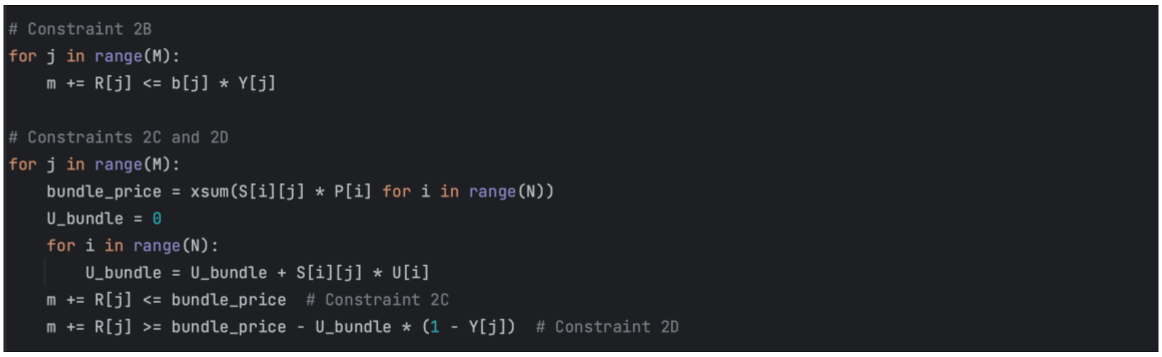


```
Y = [m.add_var(var_type=BINARY) for j in range(M)]  # Binary variables (=1 if customer j buys the bundle, 0 otherwise)
P = [m.add_var(var_type=INTEGER, lb=0) for i in range(N)]  # Prices of items
R = [m.add_var(var_type=INTEGER, lb=0) for j in range(M)]  # Prices of the bundles if bought 0 otherwise
```

Figure 7: Variables of the model

The three vectors of variables have already been introduced in the previous chapter during the presentation of the model. Y are the variables that were called X in chapter 6 (The name of these variables has been changed to avoid confusion with the optimal value of other variables, in fact python-MIP uses the wording "test.x" to refer to the optimal value of the variable "test" once the problem has been solved.). The other variables, P and R, are exactly those illustrated in the previous chapter (in this case P represents prices of optional). The lower bound ($lb = 0$) inserted on P and R was inserted to satisfy the constraints $(2e)$ and $(2f)$ of the $M1$ model.

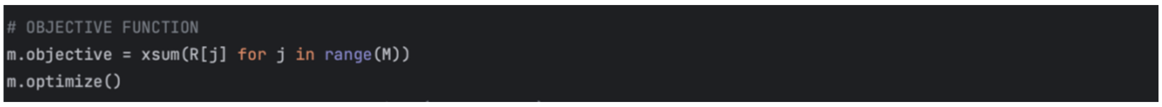
To insert the remaining constraints of model $M1$, i.e., constraints $(2b)$, $(2c)$ and $(2d)$ the following code is sufficient:



```
# Constraint 2B
for j in range(M):
    m += R[j] <= b[j] * Y[j]

# Constraints 2C and 2D
for j in range(M):
    bundle_price = xsum(S[i][j] * P[i] for i in range(N))
    U_bundle = 0
    for i in range(N):
        U_bundle = U_bundle + S[i][j] * U[i]
    m += R[j] <= bundle_price  # Constraint 2C
    m += R[j] >= bundle_price - U_bundle * (1 - Y[j])  # Constraint 2D
```

Figure 8: Constraints of the model

In the code for inserting constraints $(2c)$ and $(2d)$, U_bundle is exactly what was called $U(S)$ in chapter 6 and calculated as $U(S) = \sum_{i \in S} U_i$.
Finally, the objective function will be very simple and will consist, as already explained in the previous chapter, in maximizing the sum of the variables r_j, therefore:



```
# OBJECTIVE FUNCTION
m.objective = xsum(R[j] for j in range(M))
m.optimize()
```

Figure 9: Objective function

In Figure 9, the $m.optimize()$ command is used to start the optimization process contained in model m and must be inserted after all constraints, variables and the objective function. In fact, when this command is called, the MIP solver is activated to find the optimal solution to the previously defined problem.
Once all these elements have been defined, it is possible to generate instances to test the algorithm. To do this it is necessary to create an $N \times M$ matrix and a vector representing the customer budgets. This information can be read and inserted directly from a file, or, if we want to carry out statistical tests as in our case, we can write a code that dynamically and randomly generates the matrix S and the vector b every time the program is run. To avoid burdening this chapter with non-essential information, we will not show the code for generating S and b now, but the reader is encouraged to see this code in the appendix. What I have done is to write a method that initially creates a matrix with all elements equal to 0, and then sets exactly d% elements equal to 1 in random positions. The value d represents the so-called density of the matrix S and is calculated as follows:

$$d = \frac{\sum_{i \in N} \sum_{j \in M} S_i^j}{n \cdot m}$$

Density d represents the percentage of elements of the matrix equal to 1. In the real case that we are analyzing, the density of the matrix S is the percentage of optional requested on average by customers. If the density of the matrix is equal to 0.3, it means that customers want, on average, 30% of the options offered by the company. In the specific case of the automotive sector and in particular of RAM and Dodge, the density is usually quite low due to the high offer of optional from the seller. Therefore, the densities used in the tests will not exceed 0.5.
As regards the vector of customer budgets b instead, we can simply generate a vector of M random numbers between a maximum and a minimum decided by the user. However, to make the algorithm more realistic and to be able to carry out more tests in the next phase, 3 types of customers were created based on their budget: the first are the low-budget customers who have a budget between 20 and 80, then there are medium-budget customers with a budget between 80 and 140 and finally high-budget customers with a budget between 140 and 200. So, in order to create a population of 50 customers who have a large amount of money that can be spent on the vehicles of the company, you can create 35 high-budget customers, 10 medium-budget and 5 low-budget. This, in the algorithm code, is made possible through the use of three methods (one for each type of customer) that create three budget vectors with the number of customers decided by the user. Then these three vectors are concatenated and their values randomly mixed to obtain the final vector b that will be used in the algorithm.
Thanks to all this information just listed, it is possible to generate any type of instance to test the algorithm and obtain results in terms of resolution time (CPU time) and total revenues of the seller. In the next chapter, in fact, several instances will be created and statistical tests will be performed on them to verify the efficiency of the algorithm.

8. Statistical tests

8.1 First instances

When the program is run, python-MIP allows you to print the optimal value of the variables using the syntax "variable.x". In our case, therefore, this command was used to print the values of the variables of our interest, i.e., the prices of all the products and the total revenue of the seller, but it is however also possible to print the other variables such as Y, the individual revenues R_j obtained from all customers individually etc. In order to perform statistical tests on the results, it is also essential to keep track of the resolution times. This information is automatically printed by the solver at the end of the optimization process, but it is also possible to do it manually using the python module "time" and calculating the resolution time as the difference between the time calculated before the $m.optimize()$ command and the time calculated immediately after the same command. This operation was done to check whether the two times coincided, and they were always almost identical in all the tests carried out.
The first test that was carried out with the algorithm consisted of the creation of 15 different instances, in each of which the resolution time (CPU time) was monitored to verify its efficiency. What changes from one instance to another is the number of optional/items and customers. Below you will find the table which will be described in detail in the next lines.

Trial number	N° of items	N° of customers	Total time (CPU seconds)	% increase in time
1	2	3	0,0068	-
2	4	6	0,0123	81,95%
3	6	9	0,0186	50,81%
4	8	12	0,0261	40,70%
5	10	15	0,2320	788,81%
6	14	20	0,5007	115,84%
7	16	24	0,6368	27,18%
8	20	28	1,1978	88,10%
9	24	32	1,6675	39,21%
10	28	38	4,1691	150,02%
11	30	44	9,2970	123,00%
12	32	48	10,4799	12,72%
13	34	50	12,8934	23,03%
14	38	52	24,5236	90,20%
15	40	55	66,1885	169,90%

Table 1: First instances (N≈M)

The first column shows the trial identification number. The second and third columns instead, show the number of items and customers respectively. Each of the 15 tests was repeated 10 times (for a total of 150 tests) in order to have a more reliable resolution time. In fact, the time reported in the fourth column is calculated as the average of the resolution times obtained by running the program 10 times (this is because running the program with the same data does not always lead to the same result in terms of time). Finally, the percentage reported in the last column indicates the percentage increase in resolution time from one trial to the next. The 81.95% of the second row is therefore calculated as the difference between the resolution time of trial 2 and trial 1 divided by the resolution time of trial 2 (in percentage). This last column gives an idea of how much the resolution times increase by increasing the number of items and customers, in fact at each trial these two values are both increased. The graph below shows the evolution of resolution times as the number of items and customers increases, in other words moving from one trial to another.

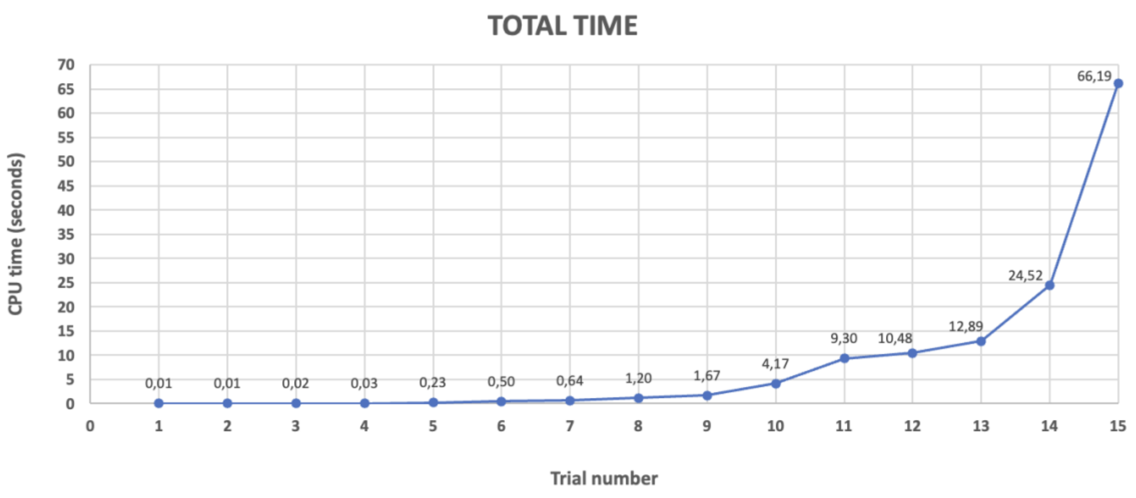


Figure 10: Evolution of resolution times

From this graph it is clear that the time to solve the graph increases exponentially as the number of optional and customers increases. Instances larger than the one contained in trial 15 were not created because times longer than 60 seconds were considered highly inefficient in a company context.
The newly created instances, however, present a critical issue. In fact, if we think about any business context and in particular the automotive sector, it is usually more common to have a large variety of customers and a more limited number of products/product features. In the created instances, however, the number of customers and items grows in parallel, making the tests carried out less realistic. In fact, if we try to create an instance like $N = 20$ and $M = 100$ we discover that the algorithm no longer finds the best solution quickly, and we should let it run for some time to find the optimal value. So, this brings us to the second type of tests, which will be carried out on more realistic instances to evaluate the efficiency of the algorithm in a real case.

8.2 Efficiency of the algorithm

To evaluate the efficiency of the algorithm in more realistic cases, where efficiency means the ability of the algorithm to find an optimal solution in a short time, instances different from those just seen were created. In fact, from this moment on, all the instances used in the statistical tests will be the same: the number of items will be equal to 10 or 20, while the number of customers can be equal to 50, 100, 150, 250 and 500. By modifying these two parameters we will obtain different instances which will allow us to make some evaluations on the algorithm used and its implications. To evaluate the efficiency of the algorithm, all possible combinations of these instances were used, density of the matrix S was equal to 0.1 and the following distribution of customer budgets were used: 20% low-budget, 60% medium-budget and 20 % high-budget (so in the case of 50 customers we will have 10 low-budget, 30 medium-budget and 10 high-budget). With this data the algorithm was run and the resolution times and the total revenues obtained were monitored. Furthermore, a time limit of 15 seconds has been inserted, beyond which the algorithm stops and returns the best result obtained up to that moment (which therefore may not be the optimal one). In fact, the algorithm searches for the optimal solution by searching among all the possible solutions, but if the resolution time is greater than the time limit it is interrupted before the end, therefore the total revenues returned may not be the optimal one as not all the solutions are been explored. However, the python-MIP solver helps us understand how much the solution found deviates from the theoretical optimum by providing what is called "best bound". The best bound, as just said, is a theoretical optimum, which means it does not necessarily be optimal. This is due to the fact that the best bound is obtained by solving a relaxed problem in which the constraints are not all respected (or better they are relaxed). Let's think for example about the constraint on prices that requires them to be integers. In the relaxed problem this constraint could be relaxed, thus obtaining a result that is not actually the true optimal value of the initial problem. Despite this, the best bound gives us an indication of how good our solution is. In particular, the best bound allows us to calculate the so-called gap, i.e., the percentage difference between the best bound and the optimal solution found by the algorithm and consequently the average gap, that is the average of the gaps obtained by running the algorithm several times with the same instance. The gap will be equal to 0 if the resolution time is less than the time limit and greater than or equal to 0 if the resolution time exceeds the time limit. Below are the tables showing the results of the algorithm with the instances mentioned above.

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
1	10	50	0,1	0,4278	2518	2518		0,00%	
2	10	50	0,1	0,3745	2920	2920		0,00%	
3	10	50	0,1	0,3342	2350	2350	0,37344	0,00%	0%
4	10	50	0,1	0,3542	2432	2432		0,00%	
5	10	50	0,1	0,3765	2440	2440		0,00%	

Table 2: Efficiency. Trial 1-5

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
6	10	100	0,1	1,8122	4108	4108		0,00%	
7	10	100	0,1	2,8260	4037	4037		0,00%	
8	10	100	0,1	1,6900	4020	4020	2,36772	0,00%	0%
9	10	100	0,1	1,9261	4010	4010		0,00%	
10	10	100	0,1	3,5843	2966	2966		0,00%	

Table 3: Efficiency. Trial 6-10

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
11	10	150	0,1	10,7438	6050	6050		0,00%	
12	10	150	0,1	10,0709	6035	6035		0,00%	
13	10	150	0,1	Time limit	6203	7111	-	12,77%	5,65%
14	10	150	0,1	Time limit	6000	7099		15,48%	
15	10	150	0,1	10,2798	6239	6239		0,00%	

Table 4: Efficiency. Trial 11-15

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
16	10	250	0,1	Time limit	9113	11932		23,63%	
17	10	250	0,1	Time limit	7857	10384		24,34%	
18	10	250	0,1	Time limit	9277	12004	-	22,72%	22,75%
19	10	250	0,1	Time limit	8824	11100		20,50%	
20	10	250	0,1	Time limit	9152	11823		22,59%	

Table 5: Efficiency. Trial 16-20

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
21	10	500	0,1	Time limit	13546	22166		38,89%	
22	10	500	0,1	Time limit	11696	22412		47,81%	
23	10	500	0,1	Time limit	13449	23253	-	42,16%	42,95%
24	10	500	0,1	Time limit	12242	22416		45,39%	
25	10	500	0,1	Time limit	13817	23224		40,51%	

Table 6: Efficiency. Trial 21-25

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
26	20	50	0,1	0,7174	2983	2983		0,00%	
27	20	50	0,1	1,1583	3278	3278		0,00%	
28	20	50	0,1	1,8601	3413	3413	1,16962	0,00%	0%
29	20	50	0,1	1,3405	3775	3775		0,00%	
30	20	50	0,1	0,7718	3388	3388		0,00%	

Table 7: Efficiency. Trial 26-30

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
31	20	100	0,1	Time limit	5526	7322		24,53%	
32	20	100	0,1	Time limit	5389	6772		20,42%	
33	20	100	0,1	Time limit	6027	7141	-	15,60%	18,92%
34	20	100	0,1	Time limit	5819	6987		16,72%	
35	20	100	0,1	Time limit	5268	6374		17,35%	

Table 8: Efficiency. Trial 31-35

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
36	20	150	0,1	Time limit	7662	10792		29,00%	
37	20	150	0,1	Time limit	7865	10824		27,34%	
38	20	150	0,1	Time limit	7415	10181	-	27,17%	26,83%
39	20	150	0,1	Time limit	7515	10192		26,27%	
40	20	150	0,1	Time limit	8278	10946		24,37%	

Table 9: Efficiency. Trial 36-40

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
41	20	250	0,1	Time limit	10342	17460		40,77%	
42	20	250	0,1	Time limit	11131	17434		36,15%	
43	20	250	0,1	Time limit	10054	17504	-	42,56%	39,73%
44	20	250	0,1	Time limit	11394	17303		34,15%	
45	20	250	0,1	Time limit	8933	16243		45,00%	

Table 10: Efficiency. Trial 41-45

Trial number	N° of items	N° of customers	Density of S	Total time (CPU seconds)	Best solution	Best bound	Average time	Gap	Average Gap
46	20	500	0,1	Time limit	16691	35019		52,34%	
47	20	500	0,1	Time limit	20885	35088		40,48%	
48	20	500	0,1	Time limit	19103	33329	-	42,68%	46,06%
49	20	500	0,1	Time limit	16842	34028		50,51%	
50	20	500	0,1	Time limit	18813	33769		44,29%	

Table 11: Efficiency. Trial 46-50

The first thing we can notice by looking at the tables shown is that the algorithm always finds a solution without getting stuck. This is positive from the seller's point of view as it means that the algorithm always suggests a price vector, even if not optimal. However, the results obtained are not always optimal, in fact the time limit is exceeded several times as the size of the instances increases, both in terms of number of customers and number of items. This is because both N and M affect the size of the matrix making it more complex and consequently the complexity of the problem also increases. As mentioned before, the python-MIP solver provides a theoretical best bound with which we can calculate the gap with the following formula:

$$Gap = \frac{Best\ bound - Best\ solution}{Best\ bound} \cdot 100$$

Then, averaging the gaps obtained we obtain the value which in the tables is indicated as average gap. In tables 2 and 3, it is equal to zero because the time taken to explore all the solutions is less than the time limit, while for example in table 4 there are two cases in which the time limit was not sufficient to analyze all the possible solutions, so we start to have a positive gap. By inserting high densities (d=0.5) and setting the number of customers and items to the maximum

(N=10, M=500), we end up with very high gaps, this means that the solution becomes unreliable (i.e., far from the optimal solution).
The increase in the gap as the number of customers increases can be seen more easily by observing the two graphs in figures 11 and 12 below.

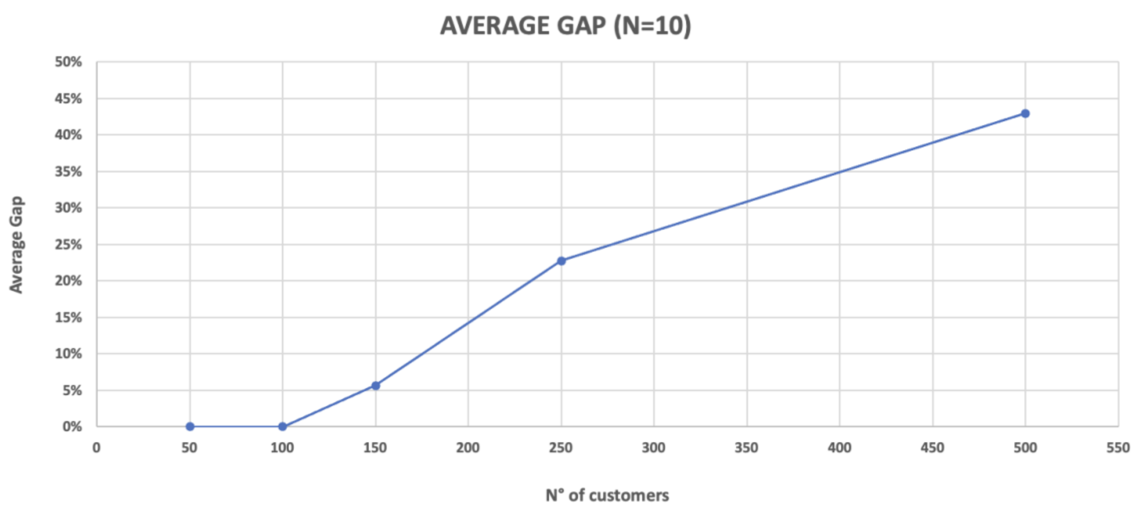


Figure 11: Average gap (N=10)

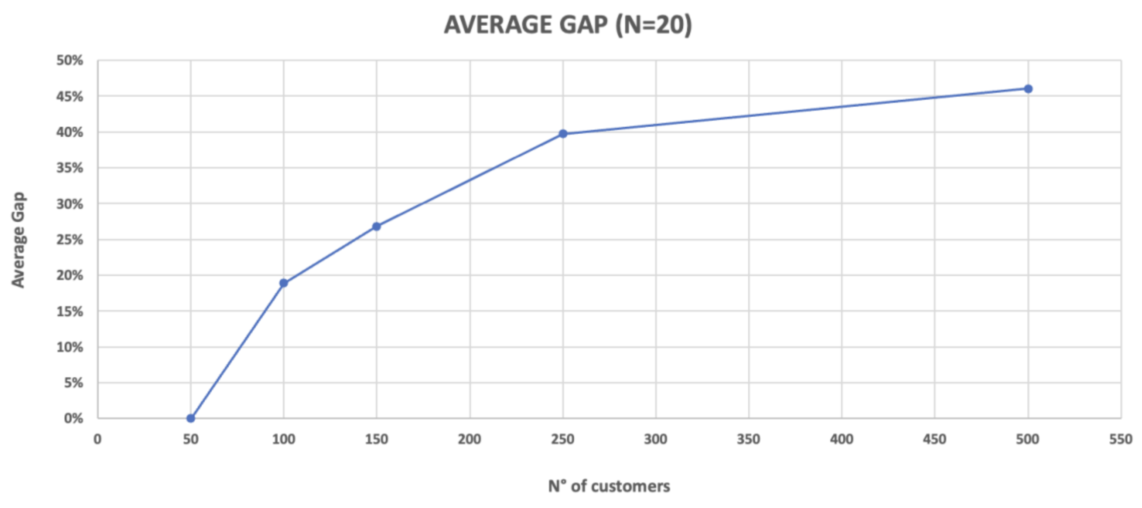


Figure 12: Average gap (N=20)

The mere existence of gaps, however, does not mean that the solution is unreliable. In fact, if the gap is very low, we could still be very close to optimal as almost all the solutions have been analyzed. Taking table 4 as an example, it is possible to see that only in two of the 5 trials carried out a gap occurred and consequently the average gap is around 5%. Values of this kind can certainly be considered acceptable, but looking at the latest tables, for example tables 10 and

11, we find much higher gaps. In these cases, the algorithm generates a solution that is too far from the optimal solution and therefore it can no longer be considered efficient. As regards the intermediate gaps, such as those in tables 5 and 8, an additional consideration can be made. In these cases, the gap is around 20%, therefore quite high, but we could try to relax the time limit a little in order to make these results acceptable. The relaxation of the time limit and the concept of reliability are obviously a decision of the seller and depend on his predisposition to accept longer resolution times and non-optimal results (for example, with the same time limit, a company could accept a gap of 10% while another does not). By carrying out some tests it was proven that bringing the time limit to 30 seconds in the instances of table 5 it is possible to reduce the gap to 15% instead of 23%. This relaxation concept cannot be used for higher gaps as too much relaxation would be needed to achieve acceptable results.
From these tables it is also clear that the algorithm used appears to be fast and efficient for small instances, but less for larger ones. For this reason, a possible alternative solution for larger instances will be proposed in the next chapter.

8.3 Density tests

The tests performed up to this moment have always been done using a density of 0.1 and a fixed budget distribution. To better understand the implications of the algorithm used, statistical tests were performed to determine how the results change with density and budget variations. In this paragraph we will see the so-called density tests.
This type of tests was carried out always keeping the same budget distribution (20% low-budget, 60% medium-budget and 20% high-budget), the same number of customers and items, but modifying the density. The following tables therefore show how revenues vary as the density of the S matrix varies. The density values used are 0.1, 0.2, 0.3, 0.4 and 0.5. As regards the number of customers, in this case not all possible instances were used, but only the smallest ones because they allow obtaining a result in a short time. For this purpose, a time limit of 60 seconds was used and all instances for which a result was not found in this time

interval were not considered. Furthermore, it is important to remember that the budget vector used is always the same, to make the revenues of the trials comparable. The table in which the wording "NOT FOUND" appears in red is the first table in which the time limit is not sufficient to find the optimal result. From that moment on, therefore, tables with higher densities were no longer calculated as the optimal result would not have been found even in subsequent ones.

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
1	10	50	0,1	2368	2226
2	10	50	0,1	2070	
3	10	50	0,1	2001	
4	10	50	0,1	2542	
5	10	50	0,1	2148	

Table 12: Density tests. Trial 1-5

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
6	10	50	0,2	2848	2891
7	10	50	0,2	2835	
8	10	50	0,2	3016	
9	10	50	0,2	2633	
10	10	50	0,2	3123	

Table 13: Density tests. Trial 6-10

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
11	10	50	0,3	3280	3406
12	10	50	0,3	3287	
13	10	50	0,3	3748	
14	10	50	0,3	3261	
15	10	50	0,3	3454	

Table 14: Density tests. Trial 11-15

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
16	10	50	0,4	3475	3444
17	10	50	0,4	3763	
18	10	50	0,4	3451	
19	10	50	0,4	3116	
20	10	50	0,4	3415	

Table 15: Density tests. Trial 16-20

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
21	10	50	0,5	3490	
22	10	50	0,5	3349	
23	10	50	0,5	3564	3493
24	10	50	0,5	3708	
25	10	50	0,5	3352	

Table 16: Density tests. Trial 21-25

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
26	10	100	0,1	4060	
27	10	100	0,1	4474	
28	10	100	0,1	3981	4264
29	10	100	0,1	4168	
30	10	100	0,1	4635	

Table 17: Density tests. Trial 26-30

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
31	10	100	0,2	5482	
32	10	100	0,2	5434	
33	10	100	0,2	4965	5305
34	10	100	0,2	5503	
35	10	100	0,2	5143	

Table 18: Density tests. Trial 31-35

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
36	10	100	0,3		
37	10	100	0,3		
38	10	100	0,3	NOT FOUND	
39	10	100	0,3		
40	10	100	0,3		

Table 19: Density tests. Trial 36-40

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
41	20	50	0,1	3126	
42	20	50	0,1	3724	
43	20	50	0,1	3349	3451
44	20	50	0,1	3533	
45	20	50	0,1	3525	

Table 20: Density tests. Trail 41-45

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
46	20	50	0,2	3523	3712
47	20	50	0,2	3937	
48	20	50	0,2	3513	
49	20	50	0,2	3811	
50	20	50	0,2	3777	

Table 21: Density tests. Trial 46-50

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
51	20	50	0,3	3709	3851
52	20	50	0,3	3757	
53	20	50	0,3	3873	
54	20	50	0,3	3928	
55	20	50	0,3	3989	

Table 22: Density tests. Trial 51-55

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
56	20	50	0,4	4103	3920
57	20	50	0,4	3681	
58	20	50	0,4	3894	
59	20	50	0,4	3995	
60	20	50	0,4	3928	

Table 23: Density tests. Trial 56-60

Trial number	N° of items	N° of customers	Density of S	Total Revenues	Average revenues
61	20	50	0,5	3715	3966
62	20	50	0,5	4076	
63	20	50	0,5	4123	
64	20	50	0,5	3794	
65	20	50	0,5	4124	

Table 24: Density tests. Trial 61-65

As can be seen from the tables, what has been done is to keep the number of customers fixed by trying to increase the density of matrix S, in order to see the effects on revenues. Unfortunately, for large size instances it was not possible, because for example by setting N=10 and M=150 the optimal result was found only for d=0.1, in all other cases the time limit was exceeded. The first thing we can say about these results is therefore that there seems to be an increase in resolution times as the density increases and in particular, when the number of customers increases, the increase in these times grows exponentially, for

example with N=10, M=100 and d=0.3 (table 19) we obtain very inefficient times (>60 seconds).
The most interesting result, however, concerns the revenues, which follow a particular trend. The first thing you notice is an increase in revenues as density increases, however this increase does not seem to be linear. The graphs in figures 13 and 14 show more clearly the trend of revenues as the density of S increases for the instances N=10, M=50 (figure 13) and N=20, M=50 (figure 14).

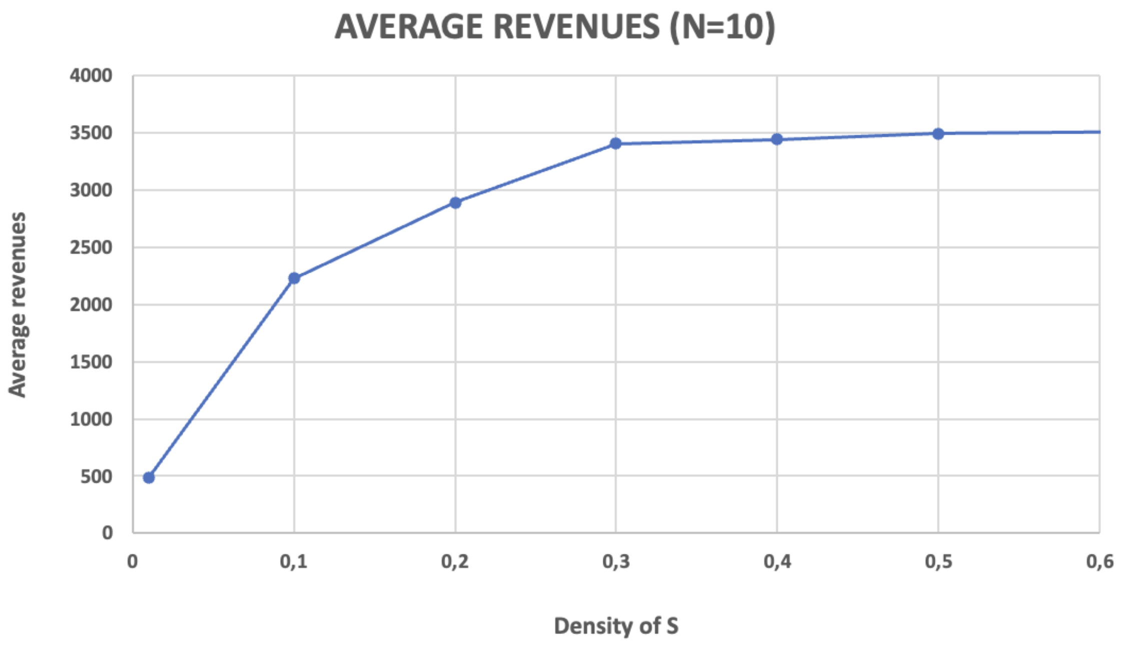


Figure 13: Variation of average revenues in relation to the density of matrix S (N=10)

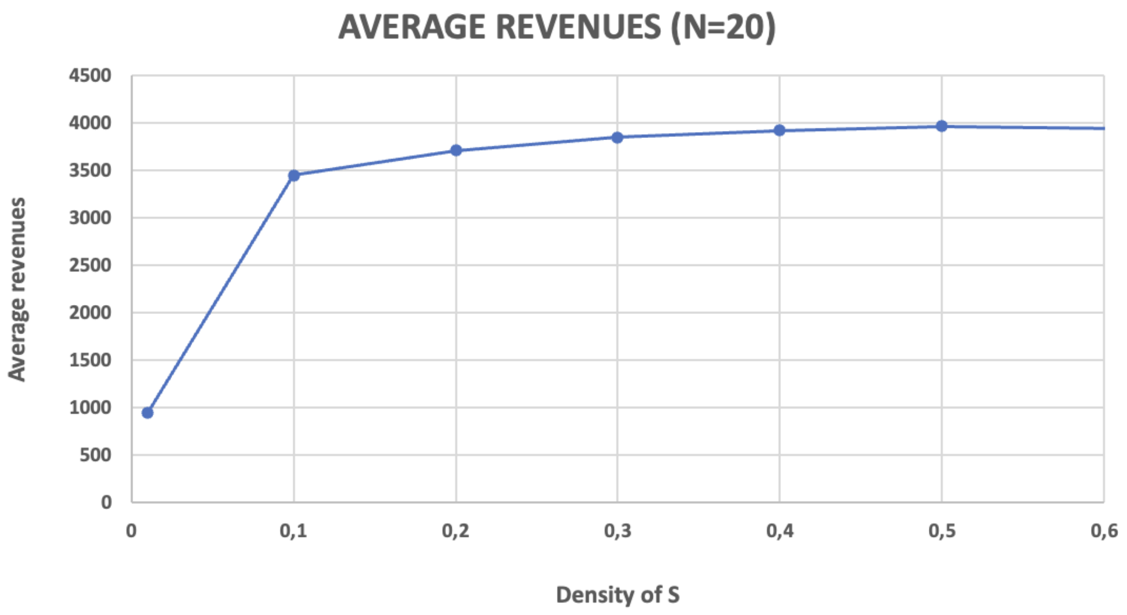


Figure 14: Variation of average revenues in relation to the density of matrix S (N=20)

The relationship between total revenues and S density appears to be logarithmic. To confirm this trend, a very low density (less than 0.1) and a very high density (0.9) were chosen in both graphs, in order to have a clearer idea of the trend of the graph at the extremes. What has been noticed is that in reality for very high densities, a phenomenon of reduction in total revenues can occur. Therefore, by increasing the density beyond a certain limit we can have a reduction in revenues. For very low densities, however, revenues grow very quickly as d increases, and then stabilize around a certain value when quite high densities are reached (e.g., 0.4, 0.5, 0.6). The explanation of this curve is not immediate and will require an example to understand it better. First of all, in the first part of the graph, where the densities are very low, the rapid increase in revenues as the density increases is due to a greater number of sales possibilities for the seller. In fact, densities of 0.1 are very low and this can be seen by looking at the S matrix which shows how customers are interested in few items. Therefore, by increasing the density of the matrix a little, it is possible to find price vectors that satisfy many customers and allow higher revenues to be obtained. However, when the density becomes very high, for example above 0.6 (remember that a density of 0.6 means that on average customers are interested in 60% of the cars' optional), a new problem is encountered, namely that customers want many options but at the same budget as before. The seller can therefore take two paths: Lower prices

or agree not to sell to some customers who have too low a budget. In both cases, however, this leads to a reduction in revenues.
To better clarify this concept, we use a simple example characterized by a $3x3$ S matrix and we will call it example 2.

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad b = \begin{pmatrix} 3 & 5 & 6 \end{pmatrix}$$

In this simple problem there are 3 optional and 3 customers and the budgets are shown in vector b. The density of the matrix S is equal to 0.33. It is clear that the optimal solution in this case is the price vector $P_i = \{3 \quad 5 \quad 0\}$ which guarantees a total revenue of 13 for the seller.

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad b = \begin{pmatrix} 3 & 5 & 6 \end{pmatrix}$$

If in the same problem we increase the density of the matrix, for example supposing that customer 3 wants, in addition to optional 2, also optional 3 (thus obtaining $d = 0.44 > 0.33$), it is possible to notice that now at the optimum, the seller would have a total revenue of 14 using the price vector $P_i = \{3 \quad 5 \quad 1\}$.
Therefore, as explained previously, the increase in density in this case led to an increase in revenues. Another important thing to notice is that the latter revenues obtained are the maximum obtainable, in fact each customer is currently paying a price for the bundle equal to his budget.
In fact, if at this point, we try to further increase the density of S, for example by saying that customer 3 also wants product 1, we would start to see a reduction in revenues.

$$S = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad b = \begin{pmatrix} 3 & 5 & 6 \end{pmatrix}$$

In this case we have a higher density than before ($d = 0.55 > 0.44$), but a lower total revenue ($R = 12 < 14$).

To make the concept of reducing revenues with high densities even clearer, let's calculate also the optimal solution in the case in which all customers want all the optional.

$$S = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad b = \begin{pmatrix} 3 & 5 & 6 \end{pmatrix}$$

In this case with $d = 1 > 0.55$ we obtain revenues at the optimum of $R = 10 < 12$. The reason is precisely that initially increasing the density meant increasing the seller's chances of selling (at the beginning when no one wanted the optional 3 the total revenues were not maximized as customer 3 paid for his bundle 5 but had a budget of 6. But if we assume that he also wants optional 3 we can capture all of his customer's surplus by setting $P_3 = 1$). However, if we continue to increase density (with the same budget) it means that customers want more products but they are not willing to pay more, this inevitably leads to a stabilization or even, as in this case, a contraction of total revenues.
The implications of these results are not easy to deduce and may require additional studies and testing. What do these results actually tell us? It would seem to suggest that companies should offer many interesting optional to customers so that the latter can buy them, increasing seller's revenues. However, the results obtained show, on the other hand, that having an excessive number of optional interesting for customers could lead to an unnecessary complication of the car, since, beyond a certain limit of optional, revenues could stabilize or even contract. The density tests therefore seem to suggest to the company that it might be a good idea to sell a certain number of optional not desired by customers, in order to maintain the density of the S matrix within a certain limit, maximizing total revenues.
In the next paragraph we will analyze how revenues vary as a function of another parameter, which is the distribution of budgets, thus carrying out the so-called budget tests.

8.3 Budget tests

This type of test is very interesting to see if and how revenues vary in relation to the distribution of customer budgets. To do this, 5 different budget distributions were created:

1. Distribution 1: 80% low-budget, 20% medium-budget, 0% high budget;
2. Distribution 2: 0% low-budget, 20% medium-budget, 80% high-budget;
3. Distribution 3: 10% low-budget, 80% medium-budget, 10% high-budget;
4. Distribution 4: 30% low-budget, 40% medium-budget, 30% high-budget;
5. Distribution 5: 44% low-budget, 10% medium-budget, 46% high-budget.

This type of test was carried out for all instances for which it is possible to find an optimal solution within a time limit of 60 seconds. The densities were also all tested as long as possible, because in fact with N=10 and M=150, densities higher than 0.1 did not allow the optimal result to be found within the time limit. Each trial in the following tables was repeated 10 times, in order to obtain a more reliable result (in terms of revenues). In other words, the last column of the tables shows a total revenue which is equal to the average of the total revenues obtained by running the program 10 times with the same data, for a total of 600 runs to compute all the tables below.
It is also important to remember that the 50 tests of each table were performed with the same matrix (but obviously different budget vectors), in order to make the results comparable.
In the total revenues column, the results are highlighted with colors: dark green indicates the highest revenues, light green the second highest, dark red the lowest, light red the second lowest while white the intermediate.

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
1	10	50	0,1	40	10	0	1311
2	10	50	0,1	0	10	40	3667
3	10	50	0,1	5	40	5	2556
4	10	50	0,1	15	20	15	2402
5	10	50	0,1	22	5	23	2498

Table 25: Budget tests. Trial 1-5

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
6	10	100	0,1	80	20	0	2214
7	10	100	0,1	0	20	80	6136
8	10	100	0,1	10	80	10	4196
9	10	100	0,1	30	40	30	3876
10	10	100	0,1	44	10	46	3991

Table 26: Budget tests. Trial 6-10

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
11	10	150	0,1	120	30	0	3238
12	10	150	0,1	0	30	120	9155
13	10	150	0,1	15	120	15	6090
14	10	150	0,1	45	60	45	6236
15	10	150	0,1	66	15	69	6390

Table 27: Budget tests. Trial 11-15

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
16	10	50	0,2	40	10	0	1665
17	10	50	0,2	0	10	40	4886
18	10	50	0,2	5	40	5	3408
19	10	50	0,2	15	20	15	3043
20	10	50	0,2	22	5	23	2965

Table 28: Budget tests. Trial 16-20

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
21	10	50	0,3	40	10	0	1776
22	10	50	0,3	0	10	40	5293
23	10	50	0,3	5	40	5	3474
24	10	50	0,3	15	20	15	3371
25	10	50	0,3	22	5	23	3410

Table 29: Budget tests. Trial 21-25

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
26	10	50	0,4	40	10	0	1820
27	10	50	0,4	0	10	40	4914
28	10	50	0,4	5	40	5	3508
29	10	50	0,4	15	20	15	3279
30	10	50	0,4	22	5	23	3160

Table 30: Budget tests. Trial 26-30

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
31	10	50	0,5	40	10	0	1809
32	10	50	0,5	0	10	40	5518
33	10	50	0,5	5	40	5	3615
34	10	50	0,5	15	20	15	3188
35	10	50	0,5	22	5	23	3564

Table 31: Budget tests. Trial 31-35

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
36	20	50	0,1	40	10	0	1990
37	20	50	0,1	0	10	40	5136
38	20	50	0,1	5	40	5	3489
39	20	50	0,1	15	20	15	3570
40	20	50	0,1	22	5	23	3686

Table 32: Budget tests. Trial 36-40

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
41	20	50	0,2	40	10	0	2063
42	20	50	0,2	0	10	40	5875
43	20	50	0,2	5	40	5	3954
44	20	50	0,2	15	20	15	3817
45	20	50	0,2	22	5	23	3945

Table 33: Budget tests. Trial 41-45

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
46	20	50	0,3	40	10	0	2041
47	20	50	0,3	0	10	40	6122
48	20	50	0,3	5	40	5	3945
49	20	50	0,3	15	20	15	3763
50	20	50	0,3	22	5	23	3856

Table 34: Budget tests. Trial 46-50

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
51	20	50	0,4	40	10	0	2121
52	20	50	0,4	0	10	40	6331
53	20	50	0,4	5	40	5	4134
54	20	50	0,4	15	20	15	3735
55	20	50	0,4	22	5	23	3971

Table 35: Budget tests. Trial 51-55

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	Total revenues
56	20	50	0,5	40	10	0	2091
57	20	50	0,5	0	10	40	6369
58	20	50	0,5	5	40	5	4086
59	20	50	0,5	15	20	15	3793
60	20	50	0,5	22	5	23	3876

Table 36: Budget tests. Trial 56-60

The first result that can be noticed, which is also the result we expected to obtain, is that when the customers are high-budget you obtain greater revenues than when the population you are targeting is mainly made up of low-budget customers. This is obvious because, since the S matrix used within the same table is always the same, the only thing that changes are the budgets. If

customers are willing to pay more for the same number of products (therefore they are high budget rather than low budget) the prices may be higher, and therefore the revenues too.
What is more interesting, however, concerns the other 3 results obtained because they suggest something that may not be obvious. To help the reader, two summary tables have been created for N=10 and N=20, which highlight how many times the different distributions were the best result, the worst or an intermediate result. Taking distribution 5 with N=10 as an example (table 37) and reading the table by rows, we discover that in the tests carried out this distribution was 2 times the second worst, 4 times intermediate (white color) and 1 time the second best.

Summary (N=10)					
Trial/color	🟥	🟥	⬜	🟩	🟩
Distribution 1	7	0	0	0	0
Distribution 2	0	0	0	0	7
Distribution 3	0	1	0	6	0
Distribution 4	0	4	3	0	0
Distribution 5	0	2	4	1	0

Table 37: Summary table (N=10)

Summary (N=20)					
Trial/color	🟥	🟥	⬜	🟩	🟩
Distribution 1	5	0	0	0	0
Distribution 2	0	0	0	0	5
Distribution 3	0	1	0	4	0
Distribution 4	0	4	1	0	0
Distribution 5	0	0	4	1	0

Table 38: Summary table (N=20)

Let's start from the observations that can be made on distribution 5. This type of budget distribution is bimodal and represents a situation in which the company decides to sell to a customer population made up of many low and high budget customers and a few medium budget customers. It is interesting to analyze this situation to understand whether there are particular advantages in addressing a population of customers of this type, or whether it is convenient to have all customers with similar budgets. From the results obtained in the tests, we can

see that the bimodal never provides the best solution, but always positions itself in intermediate positions (8 times out of 12 it has the white color in the summary table). This suggests that there is no reason why companies should target customer populations with a bimodal budget distribution.
The last two distributions (3 and 4) are also the most interesting for the comments and observations that can be made. They represent a normal distribution of budgets with the difference that distribution 3 has a lower variance, i.e., customers have more similar budgets to each other, while in distribution 4, although the average of the budgets is the same, customers are more heterogeneous in terms of budget. The results of the tables seem to show better performances, in terms revenues for the seller, with distribution 3. In fact, it is the second best 10 times out of 12, while distribution 4 is the worst 8 times out of 12. This result suggests a very important thing, that is, it is not only the average of the budget distribution that influences the seller's revenues, but also the variance. In fact, when customers are homogeneous in terms of budget, it will be easier to find a price vector that "satisfies" everyone, which means that convinces almost all customers to buy their own bundle. If, however, the budgets are heterogeneous, it will be more complicated to find a price that satisfies everyone and at the same time keeps revenues high. If, for instance, we think about a population with many low and high budget customers, it will be difficult to set a price for items that satisfies everyone, because if the price is high, only the high budget customers will buy the bundle, while if the price is low almost everyone will buy but the revenues of the sellers will be lower due to lower prices.
To confirm this observation, a test was done by running the program first with very "concentrated" normal distributions, and then with a distribution with very high variance. What emerged is that when the variance is very low (e.g., all customers are medium-budget) the number of customers who buy the bundle is around 100%. In other words, the solver is able to find a price vector that convinces all customers to buy its bundle. However, when distributions with very high variances are used, the number of customers who buy the bundle drops between 70% and 80%, thus causing a decrease in revenues. It must also be considered that in the tests carried out there are only 3 categories of customers based on budget (low, medium and high), but if this number were higher, for example if we had 10 categories of customers, the difference just highlighted

would be much more significant. If with 10 customer categories we had all the customers belonging to category 6, for example, the total revenues would be much greater than the ones we would have with a population in which the customers are equally distributed within all 10 categories.
What can therefore be deduced from these results is that if we think from the point of view of the company (the seller), it is certainly more convenient to serve customers who belong to the same category in terms of budget, rather than serving different targets (for instance high budget and low budget at the same time).
All the observations that have been made in this last chapter are interesting and allow us to reach important conclusions from a business point of view, such as what we have just seen on the variance of customer budgets. However, to have an even clearer vision, other tests and demographic studies would be needed which were left out in this thesis as it was decided to focus on another problem. In fact, as was explained in paragraph 8.2, the algorithm used showed good results for relatively small instances, but proved to be a bit lacking when it came to larger instances, leading to very high gaps in the case of largest ones (e.g., N=20, M=500). For this reason, it was decided not to continue further into statistical tests, but to focus on writing a second algorithm that allows the total revenues of the first to be improved in the case of large size instances. This second algorithm will be presented in the next chapter.

9 Alternative solutions for large size instances

9.1 Heuristic approach

When dealing with an optimization problem, there are two macro categories of solution algorithms: exact and heuristic algorithms. The objective of the first type is to find the optimal solution, therefore, given an objective function, the constraints of the problem and the variables, the algorithm tries to find among all the possible solutions the one that maximizes (or minimizes) the objective function. The problem with exact methods, however, is that when the problem becomes large, the search for the optimal solution could take a very long time, even hours, days or weeks. Precisely for this reason, in the last 30 years, the so-called approximate methods have become increasingly important. What characterizes this second category of solution algorithms is the fact that the certainty of finding the optimal solution is given up in exchange for a very good result obtained in a much shorter time interval. For example, let's imagine the real case of a company that has to make an offer to customers who show up, but there are millions and millions of possible solutions. What would be better, finding the optimal solution in a week, or finding a slightly worse solution but in 30 seconds? The first option is not only worse, but sometimes can also be unfeasible in a business context, as the customer may not be willing to wait a week for the answer. This is the main reason that led to the development of heuristic algorithms.
The second algorithm developed in this thesis is based precisely on these observations. In concrete terms, what will be done is to use a local search algorithm which, starting from an initial solution, will analyze the so-called neighborhoods, i.e., the solutions close to the initial one, then trying to modify the solution a little to see if it improves or worsens. This process of starting from an initial solution and trying to improve it iteratively can be defined as intensification, i.e., a process of exploitation of accumulated experience up to that moment.

Going into more detail, the algorithm will be characterized by two fundamental processes: solution construction and solution improvement. Therefore, starting from an initial solution generated with a certain logical criterion, we will try to iteratively improve this solution by adding elements step-by-step. An ordered list of candidates will be created to be added at each iteration and, if the addition of the candidate improves the solution, that will be the new optimal solution, otherwise we move on to the second candidate on the list. This process will be repeated until the time limit is exceeded.
In the next paragraph we will go into more detail about the algorithm, explaining how the criterion for creating the list of candidates was defined and the search method used.

9.2 Research criterion and candidate list

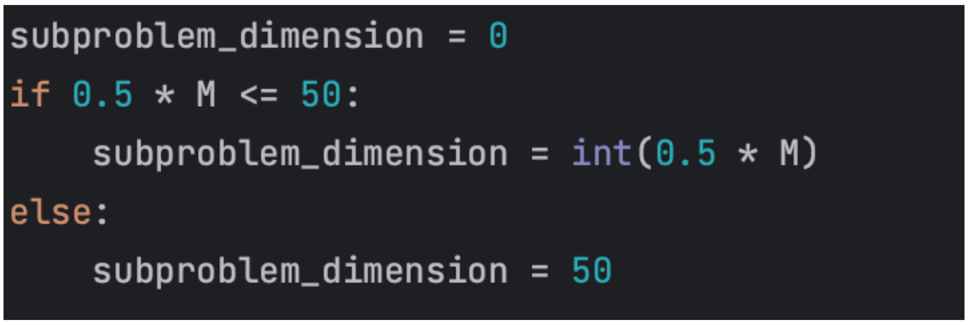
As anticipated in the previous chapter, we will now try to find a second algorithm that allows us to improve the results of the first for large size instances and, to this end, we will carry out some tests to evaluate the percentage improvement in terms of total revenues compared to the first algorithm. Given that what we want to improve is the total revenues for large size instances, this second algorithm will be based on a more rational approach in the search for solutions, which means that not all possible solutions will be analyzed until the time limit is exceeded, but we will proceed with a precise search criterion: initially the data of the problem will be reorganized and then a search will be carried out on those solutions that are candidates to be the optimal solution. This algorithm obviously does not lead to the optimal solution within the time limit, but using a precise and rational search criterion will lead to better results than the first in some cases. The first thing to do is therefore to establish a search criterion, and, in order to do this, it is necessary to carry out some tests by running the first algorithm.
The first thing we can do is try to run the program with instances small enough to allow us to find the optimal solution within reasonable times (around 60 seconds). At this point we can try to understand how an optimal solution is structured on average. How many customers are served at the optimum? Which customers

bring the majority of revenues and which ones contribute in a less significant way? What would happen if we sorted customers in ascending order based on budget and only served the first half? What if we only served the second half? These are just some of the questions we can ask ourselves to understand how the optimal solution is structured. Starting from these results it is possible to understand which are the possible candidate solutions to be the optimal one and begin to analyze those instead of trying all the possible solutions without a search criterion (as in the first algorithm).

The first test that was carried out concerns the number of customers who are served in the optimal solutions. To do this the algorithm was run with the instances that did not exceed the time limit in chapter 8.2. In particular, the sum of the Y_j variables were verified at the optimum, which are equal to 1 if customer j buys the bundle and equal to 0 if he does not, the sum therefore tells us how many customers have been served (have bought the bundle). From this test it emerged that in the optimal solution approximately 80% of customers buy the bundle while 20% give up, because the price exceeds their budget. Furthermore, it doesn't seem to be a category of customers who tend to buy their bundle more, in other words the percentage of customers who buy the bundle is the same for low, medium and high budget customers.

Then some observations were made on the origin of the revenues. In fact, using budget distribution 3 (20% low budget, 60% medium and 20% high), it seems that most of the revenues of the optimal solution come from high budget customers. This intuition comes from the observation of budget tests, in fact high budget customers are those who can spend more on the seller's items, therefore they are the ones from whom we expect to earn the most. By carrying out some tests it was seen that high budget customers, who represent only 20% of the population, contribute to around 30% of total revenues, while low budget customers only for 5%. This led to a second type of test, in which the customers of the S matrix were sorted in ascending order of budget. By running the program, we obtain that the customers in the second half of the matrix (the 50% with the highest budget) bring 70% of the revenues to the seller. This means that if there is no possibility of serving all customers, from the seller's point of view it is better to focus on high budget customers to maximize the revenues.

A final test also revealed some information about consumer surplus of the customers. In fact, in the optimal solution the consumer surplus of high budget customers is not fully exploited. In particular, for example with M=100 and d=0.3, high budget customers spend only 50% of their budget on the desired bundle. So, if we want to find a good solution, without the expectation of finding the optimal one, we could exploit this consumer surplus by giving up the sale of some bundles to low budget customers but charging higher prices to high budget customers. As anticipated, the optimal solution serves approximately 80% of customers, so with this last proposal we do not expect to arrive at the optimal solution, but to arrive at a good solution by focusing only on high budget customers reducing the complexity of the problem.
Given this information and observations it was decided to sort the customers in ascending order of budget (therefore sorting the matrix S and the vector of budgets b) and to create an initial subproblem that is simpler than the initial problem, or better, less complex. Infact, the initial subproblem will have a maximum size of 50 customers. In particular, only 50% of the customers are taken into consideration (those with the highest budget), but if 50% of the number of customers exceeds the number 50, only the first 50 are considered in the initial subproblem. In other words, if the number of customers is less than 100, the subproblem will take into consideration 50% of the customers with the highest budget, but if the number of customers is greater than 100, only the 50 customers with the highest budget will be considered in the subproblem. This was done because if the number of customers becomes very high (for example 500), creating an initial subproblem with 50% of the customers could still cause problems. In particular, the time limit may not be sufficient to solve even just the initial subproblem. To better understand how the size of the subproblem was decided, look at figure 15 below.



```
subproblem_dimension = 0
if 0.5 * M <= 50:
    subproblem_dimension = int(0.5 * M)
else:
    subproblem_dimension = 50
```

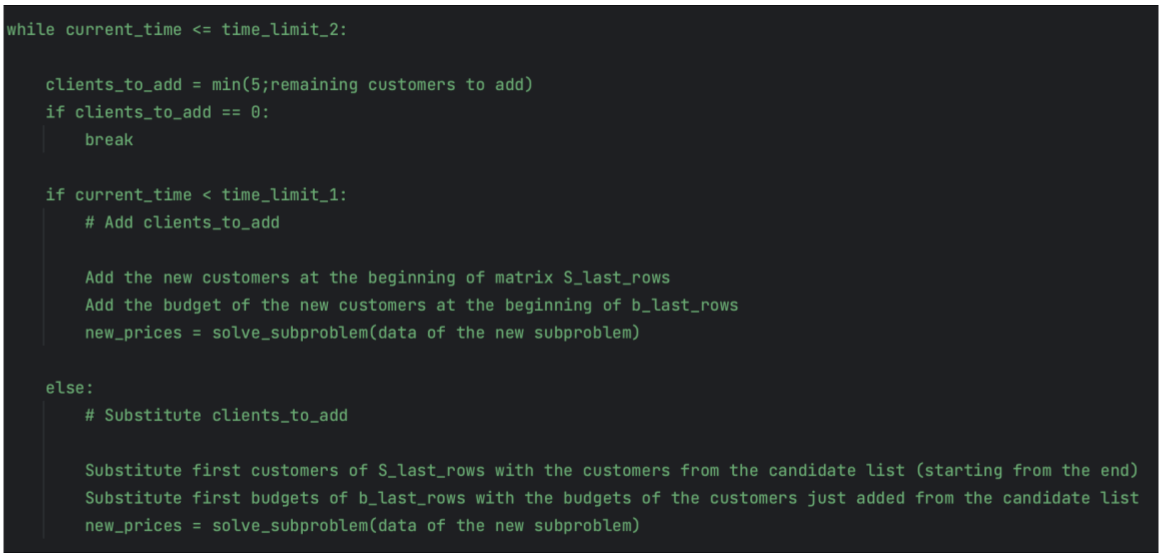
Figure 15: Dimension of the initial subproblem

Furthermore, given that each column of the S matrix represents a customer and that the S matrix has been sorted based on budgets, we can consider this matrix as the candidate list of our problem. In fact, after having generated the subproblem and the initial solution associated with it, we will try to improve with a mechanism based on two procedures: insertion of new customers from the candidate list and replacement of customers from the candidate list with already considered customers. This mechanism will be explained in detail in the next paragraph, where the algorithm used will be illustrated.

9.3 Explanation of the algorithm and pseudocode

Let's now see in detail how the code of the second algorithm was created and how it works. First of all, the first part of it, for generating the S matrix, the budget vector, etc., is the same as the one of the first algorithm. Consider that in a real case the S matrix and the budget vector are not created by a function as in our case, but are inserted directly by the seller based on the data of his customers. After creating the data of the initial problem, we need to sort the matrix S and the budget vector b, in order to have the candidate list to be used later in the algorithm (in order not to lose the initial matrix S and the vector b, they have been created copies of S and b called S_ordered and b_ordered and only these were ordered). It is important to also sort the vector b not to confuse the customer data, in fact the first element of b represents the budget of the first customer of the matrix S, so if this customer is moved to position 3 of the matrix, also the first element of b will have to be exchanged with the third element. At this point, knowing the size of the subproblem (let's temporarily call it K), it is possible to

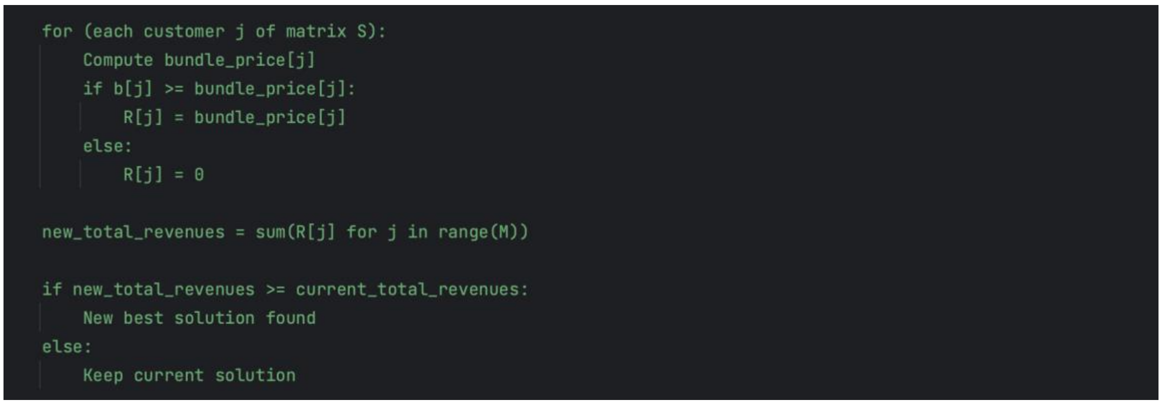
generate a sub-matrix S called S_last_rows, which is a matrix that contains only the last K columns of the sorted matrix S. The same thing was done with the vector b, thus creating a vector b_last_rows containing only the last K elements of b_ordered.
At this point, once the subproblem has been generated, it is possible to generate the initial solution. In this specific case a solution is represented by a vector of prices and the corresponding total revenues of the seller, therefore a method will be needed to solve the sub-problem. This method will be the python-MIP solver but applied to the subproblem and will receive 4 elements as parameters: number of items, number of customers, matrix S of the subproblem and corresponding vector b. Within this method, the upper bounds of the prices, the constraints, the variables and the objective function to be maximized will therefore be defined exactly as in the first algorithm. To avoid making this document heavier, the code to solve the subproblem will not be reported in this chapter, but the reader is invited to look at it in the appendix. The result returned by this method will be, as anticipated, a price vector which can be used to calculate the initial solution by multiplying the individual prices by the elements of the starting S matrix, thus obtaining the seller's total revenues. Starting from this first solution, we can try to improve the seller's revenues with the mechanism explained in paragraph 9.1, based on a dual process of adding and replacing customers. The first thing to decide is when the addition process will end and when the replacement process will begin. In the algorithm used, two different time limits were chosen, the first marks the limit beyond which the addition process stops, the second marks the end of the replacement process. Time limit 1 was set to 10 seconds, while the second to 15. Then, after generating the initial solution, customers are added for 10 seconds, checking whether this addition improves the seller's revenues. After 10 seconds this first process stops and the second begins, which checks whether there are better combinations of customers than the current one, thus replacing some of the customers present in the current solution with others of the candidate list (S_last_rows). At each iteration, the total revenue of the seller is always recalculated and at the end only the solution that guarantees the maximum revenue found is kept. Below is the pseudocode of the algorithm for a better understanding of how it works.



```
while current_time <= time_limit_2:

    clients_to_add = min(5;remaining customers to add)
    if clients_to_add == 0:
        break

    if current_time < time_limit_1:
        # Add clients_to_add

        Add the new customers at the beginning of matrix S_last_rows
        Add the budget of the new customers at the beginning of b_last_rows
        new_prices = solve_subproblem(data of the new subproblem)

    else:
        # Substitute clients_to_add

        Substitute first customers of S_last_rows with the customers from the candidate list (starting from the end)
        Substitute first budgets of b_last_rows with the budgets of the customers just added from the candidate list
        new_prices = solve_subproblem(data of the new subproblem)
```

Figure 16: Pseudocode. Add/Substitute customers

This first pseudocode is responsible for adding and replacing customers and also for calculating the new price vector. At this point it is necessary to check whether the new price vector leads to a higher or lower revenue for the seller. The second piece of pseudocode below takes care of this task (The code in figure 17 is always contained within the while loop of figure 16).



```
for (each customer j of matrix S):
    Compute bundle_price[j]
    if b[j] >= bundle_price[j]:
        R[j] = bundle_price[j]
    else:
        R[j] = 0

new_total_revenues = sum(R[j] for j in range(M))

if new_total_revenues >= current_total_revenues:
    New best solution found
else:
    Keep current solution
```

Figure 17: Pseudocode. New revenues calculation

In figure 16, the number of customers added and replaced at each iteration is equal to 5 (unless the number of remaining customers is less than 5). This was done because adding only one customer at a time slows down the algorithm, considering that every time a customer is added, the python-MIP solver must be run. By adding more clients to each iteration, you can run the solver only once for all 5 clients instead of 5 times. Furthermore, it is also important to remember that a different time limit has been inserted in the $solve\_subproblem()$ method compared to that of the first algorithm. When using the $m.optimize()$ command to run the python-MIP solver it is necessary to set a time limit equal to the time left to solve the problem, i.e., $current\_time - start\_time$. In fact, given that the overall time limit of the algorithm is 15 seconds, if the $solve\_subproblem()$ method is called after 13 seconds, the time left to find a solution is only 2 seconds. For this reason, the start time was calculated immediately before sorting the S matrix, in this way after 15 seconds from that moment the program stops in any case and returns the best result found.

9.4 Comparison

As was said in previous chapters, this heuristic algorithm was developed to try to improve the solution for large size instances, i.e., those instances that were starting to cause problems for the first algorithm. What we need to do is therefore test and compare the two algorithms written with this type of instances and understand if the second leads to an improvement compared to the first. In order to do this, other statistical tests were carried out starting from quite large instances (N=10, M=250) and then arriving to the largest instances (N=20, M=500) with different density values. Given that it has already been observed that the problem becomes more complicated with high densities, the tests on the largest instances were carried out only for high density values (0.3, 0.4, 0.5). The following tables show the comparison between the two algorithms using the usual budget distribution (20% low, 60% medium, 20% high) and a time limit of 15 seconds for the two algorithms (as already explained, for the heuristic algorithm there are two time limits: one of 10 seconds for adding customers and another 5 seconds were given for replacing customers). The last column of the tables also shows the percentage improvement (or worsening) of the average revenues obtained with the second algorithm compared to those obtained with the first.

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
1	10	250	0,1	50	150	50	9113	8240	
2	10	250	0,1	50	150	50	7857	8788	
3	10	250	0,1	50	150	50	9277	8605	-3%
4	10	250	0,1	50	150	50	8824	7865	
5	10	250	0,1	50	150	50	9152	9376	

Table 39: Comparison between algorithms. Trial 1-5

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
6	10	250	0,2	50	150	50	11431	10510	
7	10	250	0,2	50	150	50	10553	10520	
8	10	250	0,2	50	150	50	10364	10795	-3%
9	10	250	0,2	50	150	50	10205	9376	
10	10	250	0,2	50	150	50	11615	11165	

Table 40: Comparison between algorithms. Trial 6-10

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
11	20	250	0,1	50	150	50	10342	10650	
12	20	250	0,1	50	150	50	11131	9227	
13	20	250	0,1	50	150	50	10054	10607	-4%
14	20	250	0,1	50	150	50	11394	9103	
15	20	250	0,1	50	150	50	8933	10042	

Table 41: Comparison between algorithms. Trial 11-15

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
16	20	250	0,2	50	150	50	13336	10369	
17	20	250	0,2	50	150	50	12233	11967	
18	20	250	0,2	50	150	50	10338	12099	-3%
19	20	250	0,2	50	150	50	11723	13046	
20	20	250	0,2	50	150	50	12053	10619	

Table 42: Comparison between algorithms. Trial 16-20

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
21	10	500	0,3	100	300	100	20182	18603	
22	10	500	0,3	100	300	100	18650	20104	
23	10	500	0,3	100	300	100	15963	17679	4%
24	10	500	0,3	100	300	100	20622	20105	
25	10	500	0,3	100	300	100	19170	21877	

Table 43: Comparison between algorithms. Trial 21-25

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
26	10	500	0,4	100	300	100	18646	19088	
27	10	500	0,4	100	300	100	20313	20026	
28	10	500	0,4	100	300	100	19393	21632	10%
29	10	500	0,4	100	300	100	19435	17810	
30	10	500	0,4	100	300	100	13718	22305	

Table 44: Comparison between algorithms. Trial 26-30

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
31	10	500	0,5	100	300	100	17897	18494	
32	10	500	0,5	100	300	100	20845	19988	
33	10	500	0,5	100	300	100	9490	21439	13%
34	10	500	0,5	100	300	100	18788	20541	
35	10	500	0,5	100	300	100	21591	19407	

Table 45: Comparison between algorithms. Trial 31-35

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
36	20	500	0,3	100	300	100	15065	19653	
37	20	500	0,3	100	300	100	11611	18492	
38	20	500	0,3	100	300	100	14006	21356	27%
39	20	500	0,3	100	300	100	14791	18365	
40	20	500	0,3	100	300	100	21018	18899	

Table 46: Comparison between algorithms. Trial 36-40

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
41	20	500	0,4	100	300	100	15967	19527	
42	20	500	0,4	100	300	100	20814	21757	
43	20	500	0,4	100	300	100	15133	19004	24%
44	20	500	0,4	100	300	100	17288	18887	
45	20	500	0,4	100	300	100	11262	20847	

Table 47: Comparison between algorithms. Trial 41-45

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	Heuristic algorithm	Average difference
46	20	500	0,5	100	300	100	17307	19015	
47	20	500	0,5	100	300	100	15075	20153	
48	20	500	0,5	100	300	100	17391	18802	25%
49	20	500	0,5	100	300	100	13475	20049	
50	20	500	0,5	100	300	100	14656	19095	

Table 48: Comparison between algorithms. Trial 46-50

The first thing we can notice is that the first algorithm seems to be even better than the heuristic one for the instances in tables 39, 40, 41 and 42. In these cases, in fact, although the difference between the two methods is not significant, the results obtained with the first algorithm seems to be even better than those obtained with the second (by about 3%). However, moving on to the largest instances, i.e., tables 43 to 48, we notice that the situation is different. First of all, all the results of the second algorithm are better and the revenues are greater the larger the size of the instances. This improvement is less significant for N=10, where we have revenues greater than 4% in the case of density equal to 0.3 and almost 15% with density equal to 0.5. But the instances with which the difference is most noticeable are the largest ones of this model, in which an increase in revenues of approximately 25-30% is achieved compared to the first algorithm. The reason for this is that the first algorithm tries to find the optimal solution by analyzing all possible solutions among the solution tree. When the instances are large and a short time limit is set (as in this case, in which we have a time limit of only 15 seconds), not all solutions are considered and, above all, they are analyzed without a precise search method. The second algorithm instead introduces a search criterion which consists of focusing on high-budget customers, as they are the ones responsible for the greatest revenues. Therefore, we can conclude that when the instances are small, it is better to use the first method (the exact algorithm), as it is faster to analyze the possible solutions, while when the instances become large, it is more convenient to use a precise search criterion to study only those candidate solutions that could bring high revenues to the seller.

9.4 Last alternative solution

As a final solution to the problem, a second heuristic algorithm was developed to better understand the efficiency of the one just described in this chapter. The algorithm does not differ much in terms of code from the first two, but it can lead to different results. We always start from the initial problem and focus on a smaller sub-problem. In particular, the matrix S and the budget vector b are ordered exactly with the same criterion used in the first heuristic algorithm (i.e., in ascending budget order), then, instead of using a double mechanism for adding/replacing customers, it simply solves the subproblem with the python-MIP solver. Once the price vector has been obtained (always using a time limit of 15 seconds), the total revenues of the seller are calculated. We can therefore consider this algorithm as a hybrid between the two previous algorithms: the matrix S and the vector b are sorted as in the second algorithm and then the subproblem is solved with the solution method used in the first algorithm. In this case the subproblem always has a size equal to 50% of that of the initial problem, while in the first heuristic algorithm the maximum size of the subproblem was 50. So, when we have M=500, the first heuristic algorithm started with a subproblem of size 50, the algorithm we are going to use will instead start directly from a subproblem of size 250. The objective of writing this third algorithm is to show that, by changing the code even slightly, it is possible to obtain many different algorithms that give different results. Since this third method does not have parts of code different from the other two, the pseudocode will not be reported, but the reader is invited to see the algorithm code in the appendix.
Below are the tables with the revenues of the 3 algorithms created and the summary graphs that allow you to better visualize the results. The tables were made for the large size instances already described in this chapter (M=500 and

density > 0.3) and always using the same budget distribution. Regarding this point, consider that the results obtained may vary based on the budget distribution, for example this third algorithm seems to give better results with this budget distribution rather than with "homogeneous" customers (i.e., with very similar budgets).

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	1st Heuristic algorithm	2nd Heuristic algorithm
1	10	500	0,3	100	300	100	20182	18603	24339
2	10	500	0,3	100	300	100	18650	20104	23877
3	10	500	0,3	100	300	100	15963	17679	24601
4	10	500	0,3	100	300	100	20622	20105	23282
5	10	500	0,3	100	300	100	19170	21877	23359

Table 49: Second heuristic algorithm. Trial 1-5

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	1st Heuristic algorithm	2nd Heuristic algorithm
6	10	500	0,4	100	300	100	18646	19088	26518
7	10	500	0,4	100	300	100	20313	20026	26639
8	10	500	0,4	100	300	100	19393	21632	27306
9	10	500	0,4	100	300	100	19435	17810	25168
10	10	500	0,4	100	300	100	13718	22305	26836

Table 50: Second heuristic algorithm. Trial 6-11

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	1st Heuristic algorithm	2nd Heuristic algorithm
11	10	500	0,5	100	300	100	17897	18494	27728
12	10	500	0,5	100	300	100	20845	19988	28543
13	10	500	0,5	100	300	100	9490	21439	26011
14	10	500	0,5	100	300	100	18788	20541	28056
15	10	500	0,5	100	300	100	21591	19407	27857

Table 51: Second heuristic algorithm. Trial 11-15

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	1st Heuristic algorithm	2nd Heuristic algorithm
16	20	500	0,3	100	300	100	15065	19653	28460
17	20	500	0,3	100	300	100	11611	18492	27116
18	20	500	0,3	100	300	100	14006	21356	27016
19	20	500	0,3	100	300	100	14791	18365	27515
20	20	500	0,3	100	300	100	21018	18899	27113

Table 52: Second heuristic algorithm. Trial 16-20

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	1st Heuristic algorithm	2nd Heuristic algorithm
21	20	500	0,4	100	300	100	15967	19527	29317
22	20	500	0,4	100	300	100	20814	21757	20970
23	20	500	0,4	100	300	100	15133	19004	20907
24	20	500	0,4	100	300	100	17288	18887	28259
25	20	500	0,4	100	300	100	11262	20847	28895

Table 53: Second heuristic algorithm. Trial 21-25

Trial number	N° of items	N° of customers	Density of S	Low budget	Medium budget	High budget	First algorithm	1st Heuristic algorithm	2nd Heuristic algorithm
26	20	500	0,5	100	300	100	17307	19015	29827
27	20	500	0,5	100	300	100	15075	20153	27785
28	20	500	0,5	100	300	100	17391	18802	30080
29	20	500	0,5	100	300	100	13475	20049	28926
30	20	500	0,5	100	300	100	14656	19095	29630

Table 54: Second heuristic algorithm. Trial 26-30

The tables shown compare the results obtained with the 3 algorithms when they are run with the same data: same density, same budget distribution and same number of customers and items. The last 3 columns show the revenues obtained with the 3 algorithms and allow us to make some considerations. To support the tables, two graphs have been created which represent the average revenues contained in the tables.

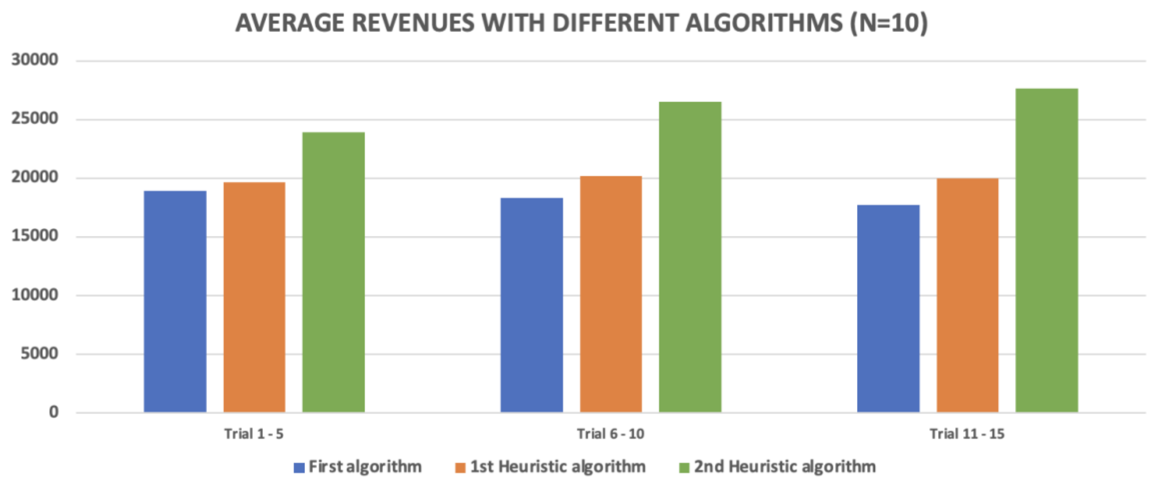


Figure 18: Average revenues with different algorithms (N=10)

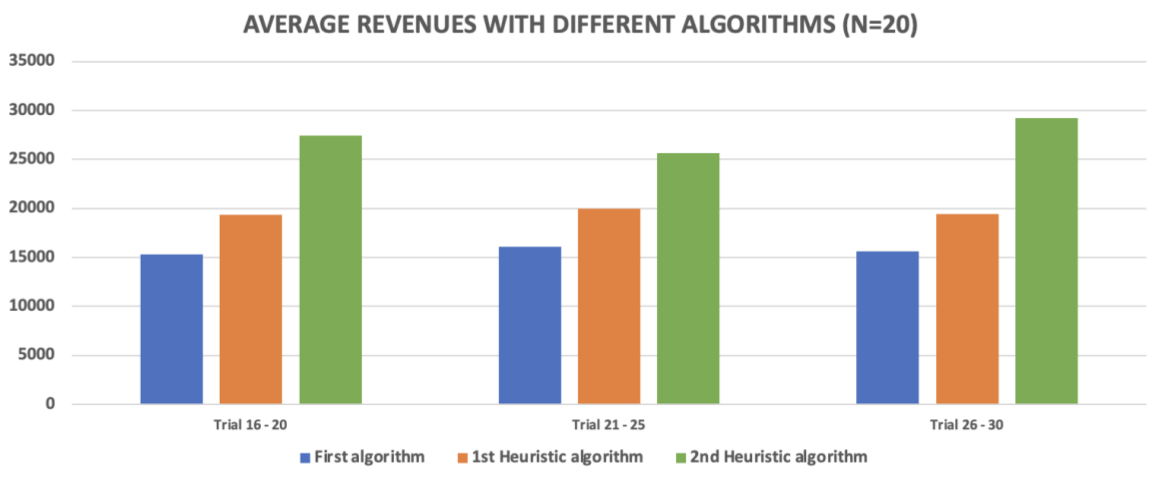


Figure 19: Average revenues with different algorithms (N=20)

For example, the first column on the left of figure 18 represents the average of the revenues obtained with the first algorithm in the 5 trials carried out.
The first thing we can notice is that heuristic algorithms always seem to work better than the first exact algorithm. Secondly, the second heuristic algorithm is always the best. By modifying some aspects of the code, it is possible to create

thousands and thousands of heuristic algorithms and these tables allow us to understand that the results can also be very different from each other and to obtain the highest revenues we should do several tests to understand which algorithm best suits our problem. As we said before, the difference between the first heuristic algorithm and the second is that the first starts from a maximum subproblem size of 50 customers, and then adds and replaces customers, looking for the best solution. The second, instead, starts immediately with a subproblem size equal to 50% of that of the initial problem. Given that the second heuristic algorithm is better than the first heuristic algorithm but the exact algorithm is the worst, we expect that if instead of a subproblem size equal to 50% we used a larger size, sooner or later the total revenues would tend to decrease. To find the best heuristic algorithm it would therefore be necessary to do more tests, but the goal of this analysis was only to demonstrate that when the size of the problem becomes large, heuristic algorithms seem to work better than the exact one.

10 Conclusions

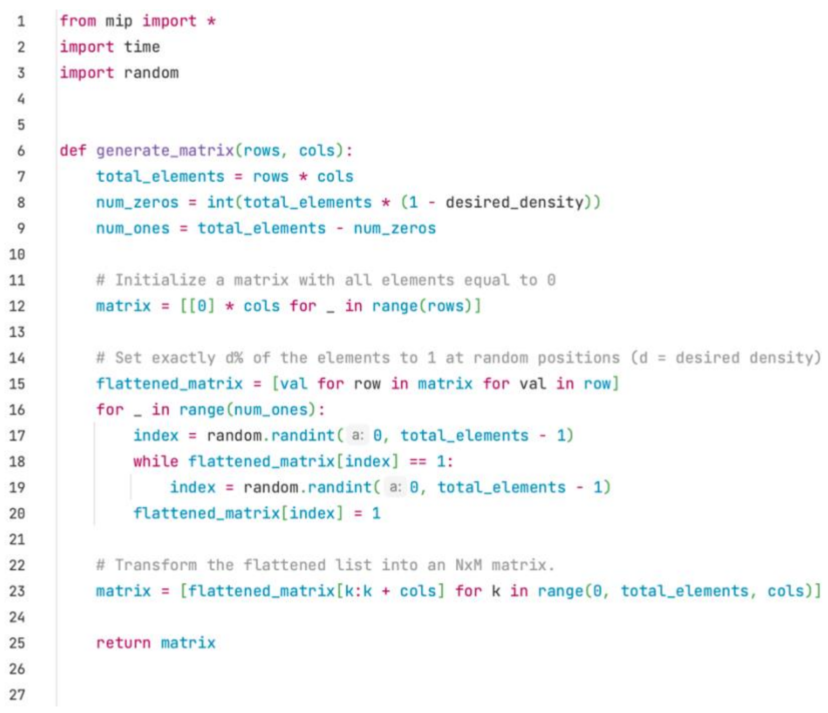
To conclude this thesis, let's review the work carried out from the beginning and focus on the results obtained and their implications. In the first part I tried to frame the topic of bundle pricing, writing a literature review on the macro topic in which it is inserted, algorithmic pricing. Some algorithmic pricing topics were then mentioned before going into more detail about bundle pricing problems. Some taxonomies have been proposed to the reader to classify all the most common types of problems available in the literature and then we focus on a specific problem called SMBPP, single-minded bundle pricing problem. After defining the model and therefore the variables, constraints, objective function etc., 3 solution methods were proposed: a first algorithm, based on an exact method and which uses the Python-MIP solver and two heuristic algorithms which differ a bit into the logic behind their code. After having written the first exact algorithm, some statistical tests were carried out which allow us to make conclusions and draw ideas for possible future analyses: first of all, we can conclude that this method is very efficient when the instances are small, therefore, in the case of a real company, when the products on sale and the customers are not so many. In fact, in these cases, it takes less than 1 second to find the optimal solution. Density tests then allow us to make important considerations on the number of items desired by customers. In the case of Stellantis, if the number of optional requested is too low (i.e., there are many optional that no one wants), the seller's total revenues will be low, but it still has a large margin for improvement. In fact, if we included more "attractive" optional, the company's sales possibilities could grow, thus exploiting better its customers' consumer surplus. This, however, is only true up to a certain point. In fact, beyond a certain density limit (i.e., the percentage of items desired by customers), total revenues tend to stabilize (or in some cases to decrease) as the budget is always the same, but the desired items increase, so I should sell more and more products to customers who do not have the ability to spend more. This leads to very interesting considerations, for example the company should evaluate whether to offer optional to its customers

such that the density remains at a percentage, decided by the company, which maximizes revenues. Finally, budget tests were carried out which demonstrated some results that we already expected, for example that, other data being equal, customers with the highest budget bring greater revenues than those with the lowest budget, but also to others less obvious. For example, companies should focus on a specific market segment and not try to sell to all types of customers (in terms of budget). In fact, this strategy has never led to the best result in terms of revenues. Another important observation that was made thanks to the budget tests is that the variance of the clients' budgets also influences the result, and not just the average. In fact, by serving "homogeneous" customers, i.e., with very similar budgets, better results are obtained than if the company decides to serve very different customers. This confirms the observations just made on the importance of carefully choosing the customers to whom you want to sell your products: a luxury brand should avoid customers with limited budgets and similarly a brand that produces "low-cost" cars should avoid to look for high-budget clients.
However, despite the effectiveness of this algorithm for small problems, when the number of optional and customers of the company grows, it starts to have some problems. In real life, car brands have many customers, so the inefficiency of this algorithm with large instances could be problematic in a real case. This led us to develop two more heuristic algorithms. The main difference with the first is that heuristic algorithms do not explore all possible solutions, but use a specific search criterion to analyze only some solutions that are candidates to be the optimal solution. In other words, the second and third algorithms, given a certain time limit, explore only some of the possible solutions and look for the best result that can be found in that time limit. The results in this case are what we expected: when using these two algorithms with small instances, the results are similar to or worse than those obtained with the exact algorithm. However, when the number of customers and optional become high, the heuristic algorithms give significantly better results than the first. This result is satisfactory because the heuristic algorithm codes were written precisely for these latter cases, so it doesn't matter if they are not efficient with small instances.
What we can therefore conclude is that these methods are all very effective and useful in a business context, because they solve a very common problem of all

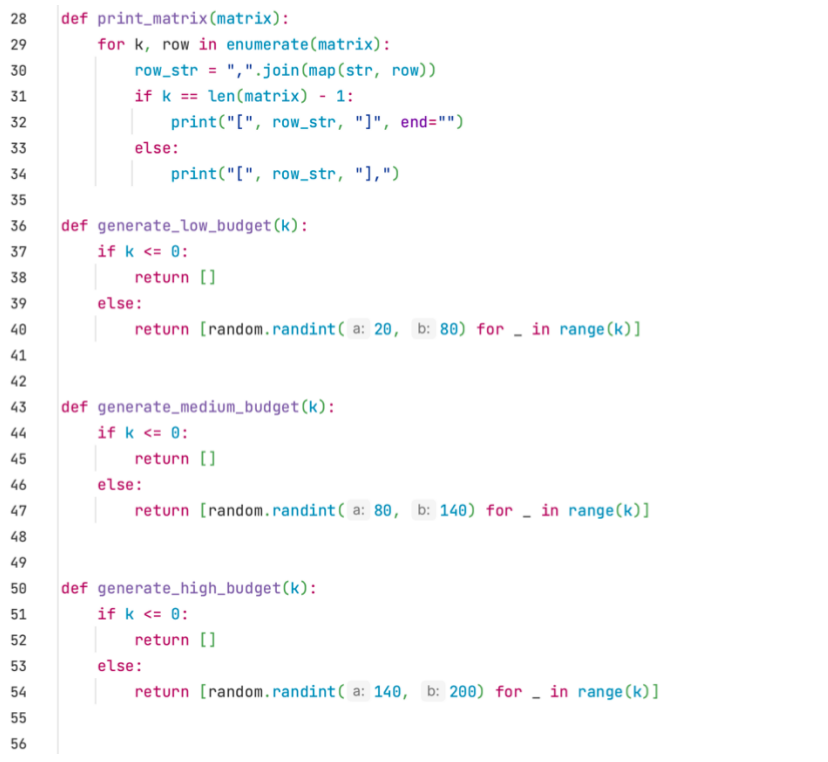
automotive brands. The tools illustrated and developed in this thesis offer an important support to business decisions in pricing problems, providing algorithms that automate the decision process of the prices of optional and more generally of the vehicles requested by customers. Obviously, each of the proposed solutions has proven to be more efficient in some cases and with certain data, but the use of all these algorithms in the corporate context can lead to a significant increase in the company's revenues, as it has been demonstrated by the tests carried out.

Appendix

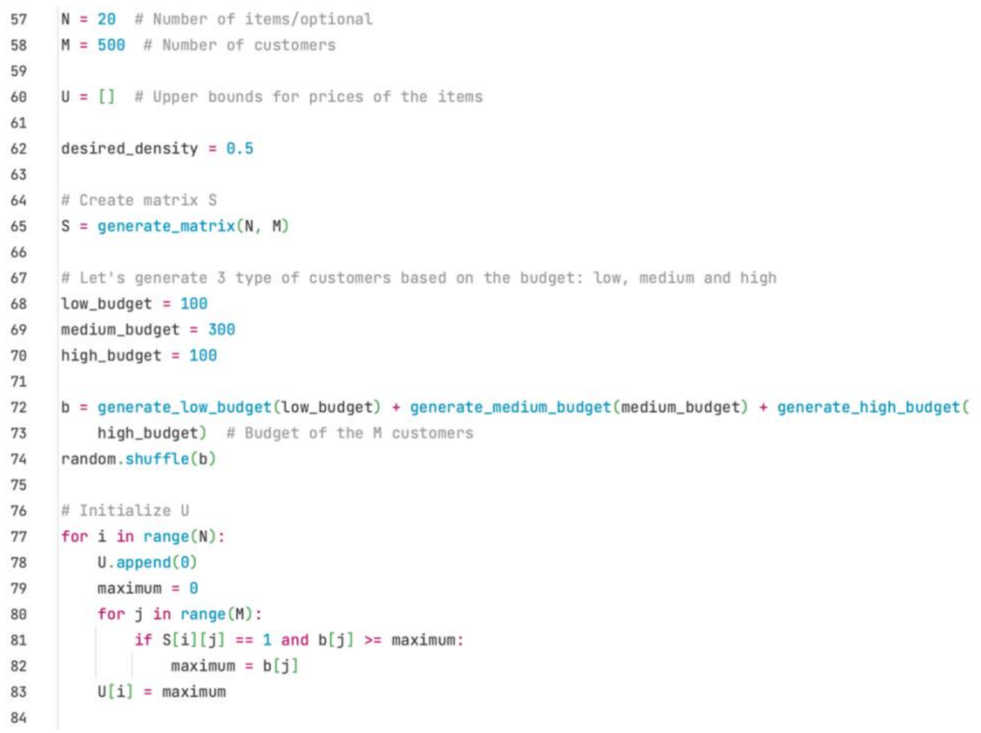
First algorithm (exact algorithm):



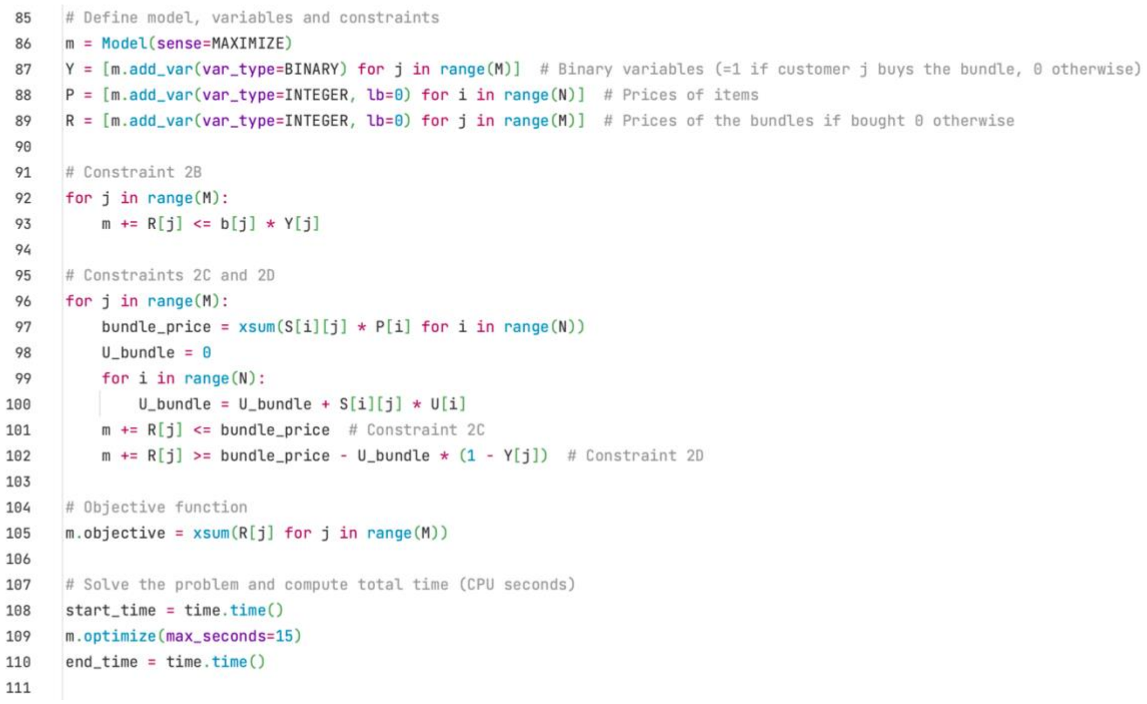
```python
from mip import *
import time
import random


def generate_matrix(rows, cols):
    total_elements = rows * cols
    num_zeros = int(total_elements * (1 - desired_density))
    num_ones = total_elements - num_zeros

    # Initialize a matrix with all elements equal to 0
    matrix = [[0] * cols for _ in range(rows)]

    # Set exactly d% of the elements to 1 at random positions (d = desired density)
    flattened_matrix = [val for row in matrix for val in row]
    for _ in range(num_ones):
        index = random.randint( a: 0, total_elements - 1)
        while flattened_matrix[index] == 1:
            index = random.randint( a: 0, total_elements - 1)
        flattened_matrix[index] = 1

    # Transform the flattened list into an NxM matrix.
    matrix = [flattened_matrix[k:k + cols] for k in range(0, total_elements, cols)]

    return matrix


```



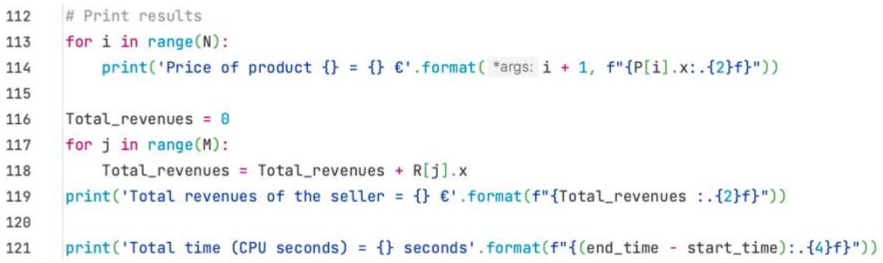
```python
def print_matrix(matrix):
    for k, row in enumerate(matrix):
        row_str = ",".join(map(str, row))
        if k == len(matrix) - 1:
            print("[", row_str, "]", end="")
        else:
            print("[", row_str, "],")

def generate_low_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 20,  b: 80) for _ in range(k)]


def generate_medium_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 80,  b: 140) for _ in range(k)]


def generate_high_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 140,  b: 200) for _ in range(k)]


```



```python
N = 20  # Number of items/optional
M = 500  # Number of customers

U = []  # Upper bounds for prices of the items

desired_density = 0.5

# Create matrix S
S = generate_matrix(N, M)

# Let's generate 3 type of customers based on the budget: low, medium and high
low_budget = 100
medium_budget = 300
high_budget = 100

b = generate_low_budget(low_budget) + generate_medium_budget(medium_budget) + generate_high_budget(
    high_budget)  # Budget of the M customers
random.shuffle(b)

# Initialize U
for i in range(N):
    U.append(0)
    maximum = 0
    for j in range(M):
        if S[i][j] == 1 and b[j] >= maximum:
            maximum = b[j]
    U[i] = maximum
```

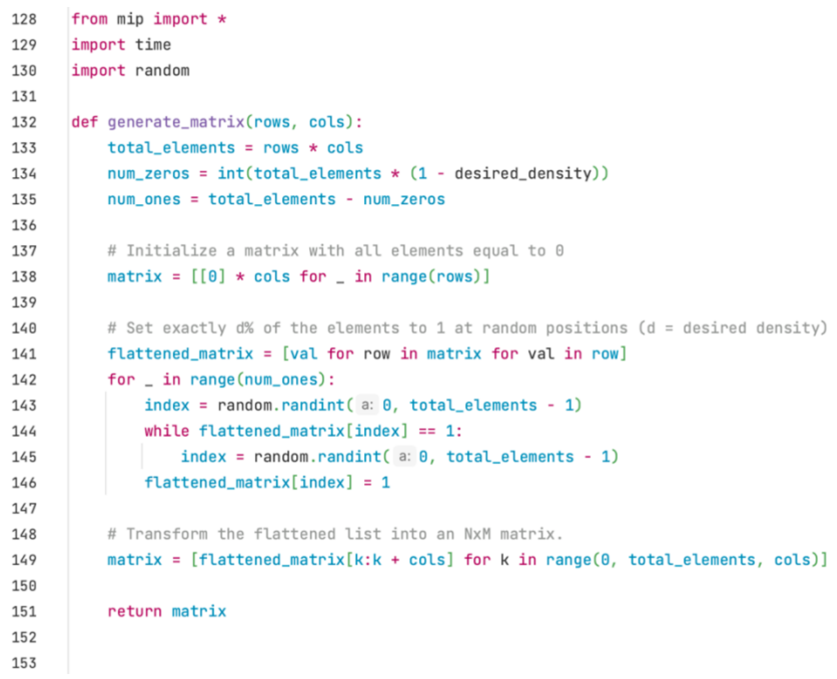


```python
# Define model, variables and constraints
m = Model(sense=MAXIMIZE)
Y = [m.add_var(var_type=BINARY) for j in range(M)]  # Binary variables (=1 if customer j buys the bundle, 0 otherwise)
P = [m.add_var(var_type=INTEGER, lb=0) for i in range(N)]  # Prices of items
R = [m.add_var(var_type=INTEGER, lb=0) for j in range(M)]  # Prices of the bundles if bought 0 otherwise

# Constraint 2B
for j in range(M):
    m += R[j] <= b[j] * Y[j]

# Constraints 2C and 2D
for j in range(M):
    bundle_price = xsum(S[i][j] * P[i] for i in range(N))
    U_bundle = 0
    for i in range(N):
        U_bundle = U_bundle + S[i][j] * U[i]
    m += R[j] <= bundle_price  # Constraint 2C
    m += R[j] >= bundle_price - U_bundle * (1 - Y[j])  # Constraint 2D

# Objective function
m.objective = xsum(R[j] for j in range(M))

# Solve the problem and compute total time (CPU seconds)
start_time = time.time()
m.optimize(max_seconds=15)
end_time = time.time()
```

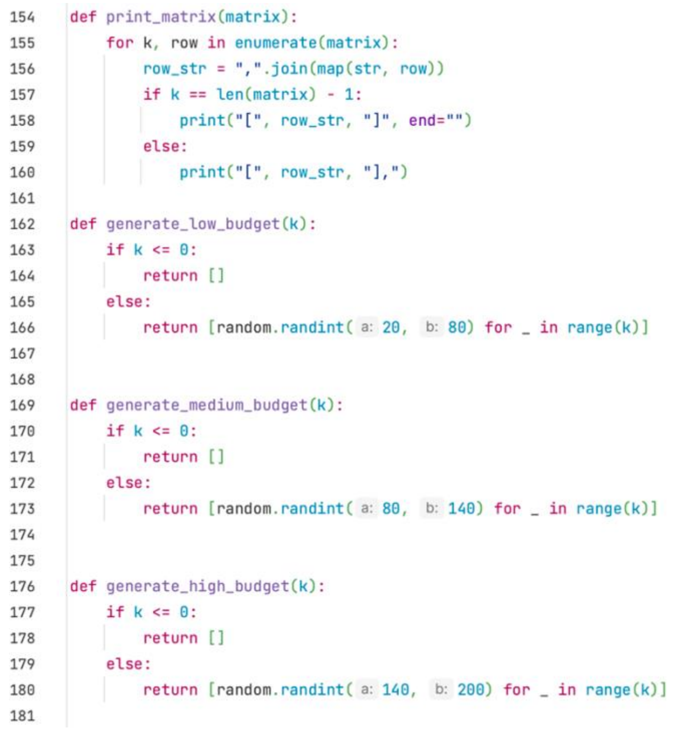


```python
# Print results
for i in range(N):
    print('Price of product {} = {} €'.format( *args: i + 1, f"{P[i].x:.{2}f}"))

Total_revenues = 0
for j in range(M):
    Total_revenues = Total_revenues + R[j].x
print('Total revenues of the seller = {} €'.format(f"{Total_revenues :.{2}f}"))

print('Total time (CPU seconds) = {} seconds'.format(f"{(end_time - start_time):.{4}f}"))
```

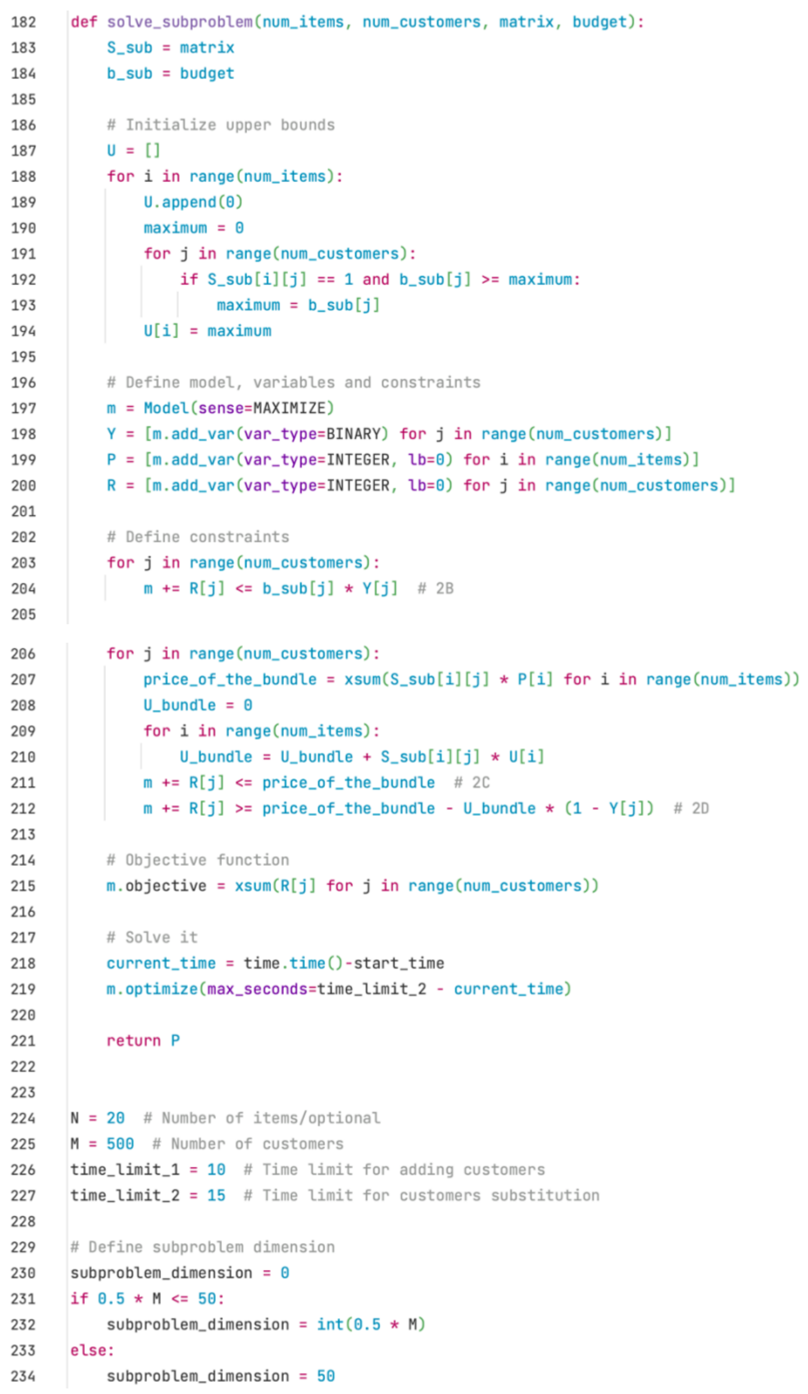
Second algorithm (first heuristic algorithm):



```python
from mip import *
import time
import random

def generate_matrix(rows, cols):
    total_elements = rows * cols
    num_zeros = int(total_elements * (1 - desired_density))
    num_ones = total_elements - num_zeros

    # Initialize a matrix with all elements equal to 0
    matrix = [[0] * cols for _ in range(rows)]

    # Set exactly d% of the elements to 1 at random positions (d = desired density)
    flattened_matrix = [val for row in matrix for val in row]
    for _ in range(num_ones):
        index = random.randint( a: 0, total_elements - 1)
        while flattened_matrix[index] == 1:
            index = random.randint( a: 0, total_elements - 1)
        flattened_matrix[index] = 1

    # Transform the flattened list into an NxM matrix.
    matrix = [flattened_matrix[k:k + cols] for k in range(0, total_elements, cols)]

    return matrix


```



```python
def print_matrix(matrix):
    for k, row in enumerate(matrix):
        row_str = ",".join(map(str, row))
        if k == len(matrix) - 1:
            print("[", row_str, "]", end="")
        else:
            print("[", row_str, "],")

def generate_low_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 20,  b: 80) for _ in range(k)]


def generate_medium_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 80,  b: 140) for _ in range(k)]


def generate_high_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 140,  b: 200) for _ in range(k)]

```

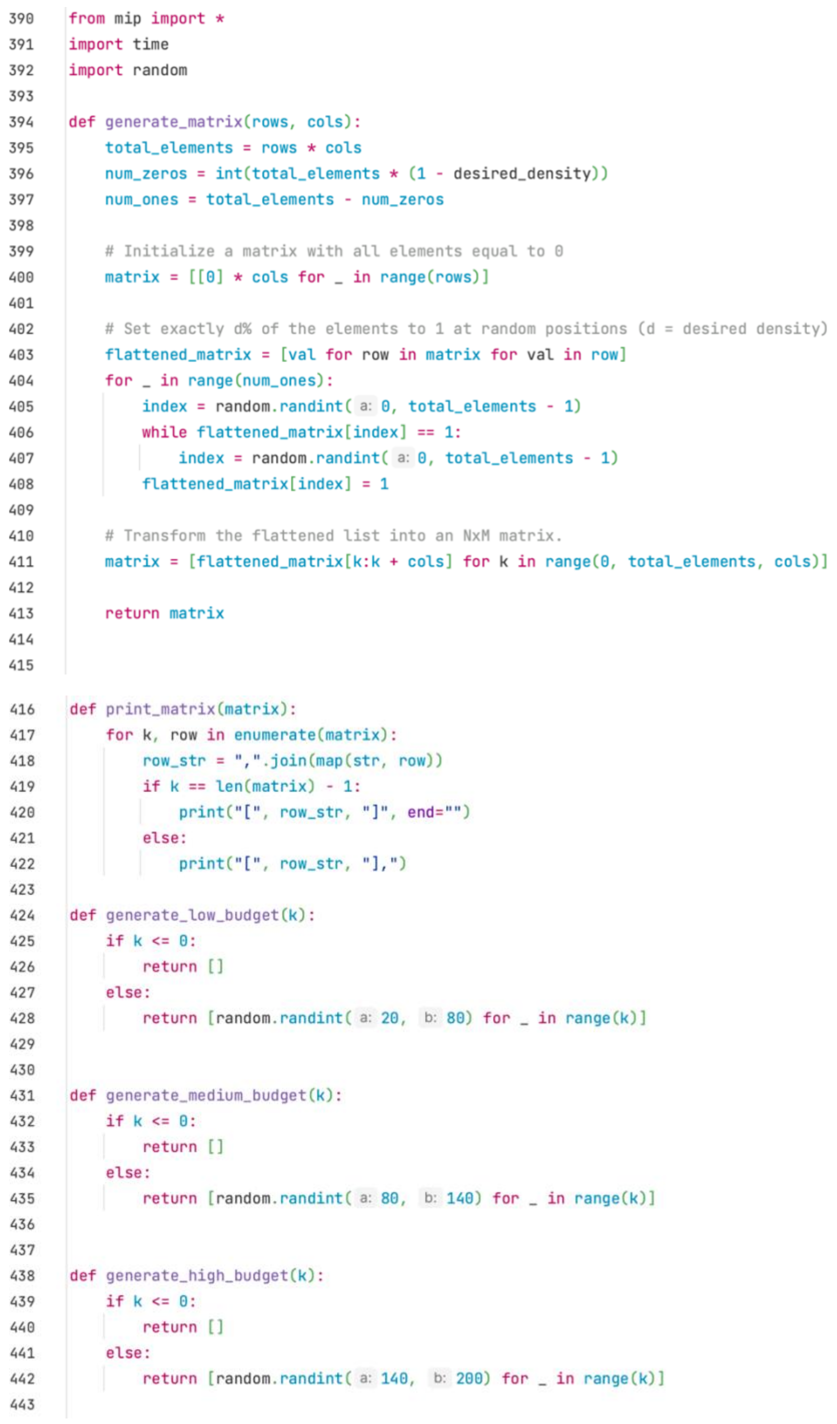


```python
def solve_subproblem(num_items, num_customers, matrix, budget):
    S_sub = matrix
    b_sub = budget

    # Initialize upper bounds
    U = []
    for i in range(num_items):
        U.append(0)
        maximum = 0
        for j in range(num_customers):
            if S_sub[i][j] == 1 and b_sub[j] >= maximum:
                maximum = b_sub[j]
        U[i] = maximum

    # Define model, variables and constraints
    m = Model(sense=MAXIMIZE)
    Y = [m.add_var(var_type=BINARY) for j in range(num_customers)]
    P = [m.add_var(var_type=INTEGER, lb=0) for i in range(num_items)]
    R = [m.add_var(var_type=INTEGER, lb=0) for j in range(num_customers)]

    # Define constraints
    for j in range(num_customers):
        m += R[j] <= b_sub[j] * Y[j]  # 2B

    for j in range(num_customers):
        price_of_the_bundle = xsum(S_sub[i][j] * P[i] for i in range(num_items))
        U_bundle = 0
        for i in range(num_items):
            U_bundle = U_bundle + S_sub[i][j] * U[i]
        m += R[j] <= price_of_the_bundle  # 2C
        m += R[j] >= price_of_the_bundle - U_bundle * (1 - Y[j])  # 2D

    # Objective function
    m.objective = xsum(R[j] for j in range(num_customers))

    # Solve it
    current_time = time.time()-start_time
    m.optimize(max_seconds=time_limit_2 - current_time)

    return P


N = 20  # Number of items/optional
M = 500  # Number of customers
time_limit_1 = 10  # Time limit for adding customers
time_limit_2 = 15  # Time limit for customers substitution

# Define subproblem dimension
subproblem_dimension = 0
if 0.5 * M <= 50:
    subproblem_dimension = int(0.5 * M)
else:
    subproblem_dimension = 50
```

```python

desired_density = 0.5

# Create matrix S
S = generate_matrix(N, M)

# Let's generate 3 type of customers based on the budget: low, medium and high
low_budget = 100
medium_budget = 300
high_budget = 100

b = generate_low_budget(low_budget) + generate_medium_budget(medium_budget) + generate_high_budget(
    high_budget)  # Budget of the M customers
random.shuffle(b)

start_time = time.time()

# Let's order S and b based on the budget
sorted_indices = sorted(range(len(b)), key=lambda k: b[k])
S_ordered = [[S[i][j] for j in sorted_indices] for i in range(N)]
b_ordered = [b[i] for i in sorted_indices]
S_last_rows = []
for i in range(N):
    row = []
    for j in range(subproblem_dimension):
        row.append(S_ordered[i][M - (subproblem_dimension - j)])
    S_last_rows.append(row)
b_last_rows = b_ordered[-subproblem_dimension:]


# Generate initial solution
prices = solve_subproblem(N, subproblem_dimension, S_last_rows, b_last_rows)

clients_served = 0
total_revenues = 0
for j in range(M):
    bundle_price = 0
    for i in range(N):
        row = []
        bundle_price += (S[i][j] * prices[i].x)
    if b[j] >= bundle_price:
        clients_served += 1
        total_revenues += bundle_price

# Try to improve initial solution until you have time
clients_added = 0
replaced_customers = 0
```

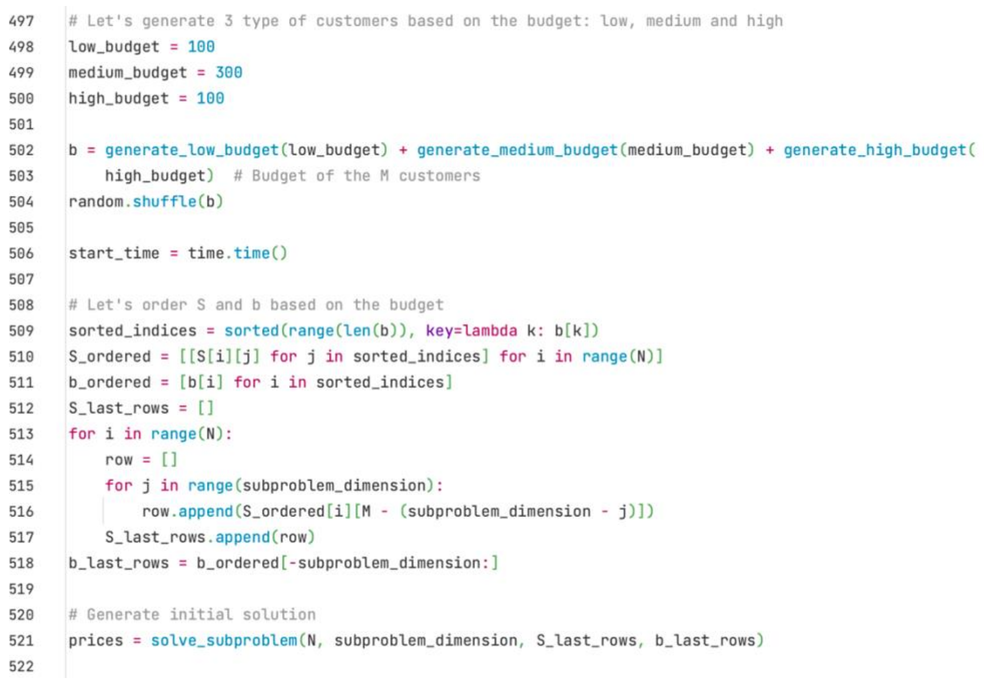
```python
while time.time()-start_time <= time_limit_2:
    replaced = False
    new_S_last_rows = S_last_rows
    new_b_last_rows = b_last_rows

    clients_to_add = min(5, M - (subproblem_dimension + clients_added))
    if clients_to_add == 0:
        break

    if (time.time()-start_time) < time_limit_1:
        # Add 5 client
        for i in range(N):
            for j in range(clients_to_add):
                S_last_rows[i].insert( 0, (S_ordered[i][M - subproblem_dimension - clients_added - j - 1]))
        clients_added += clients_to_add
        b_last_rows = b_ordered[-(subproblem_dimension+clients_added):]
        new_prices = solve_subproblem(N, subproblem_dimension + clients_added, S_last_rows, b_last_rows)

    else:
        # Substitute 5 clients
        replaced = True
        for i in range(N):
            for j in range(clients_to_add):
                new_S_last_rows[i][j]=S_ordered[i][M-subproblem_dimension-clients_added-clients_to_add-replaced_customers+j]
                new_b_last_rows[j] = b_ordered[M-subproblem_dimension-clients_added-clients_to_add-replaced_customers+j]
        replaced_customers += clients_to_add
        new_prices = solve_subproblem(N, subproblem_dimension + clients_added, new_S_last_rows, new_b_last_rows)

    # Compute new total revenues and clients served
    new_clients_served = 0
    new_total_revenues = 0
    for j in range(M):
        bundle_price = 0
        for i in range(N):
            bundle_price += (S[i][j] * new_prices[i].x)
        if b[j] >= bundle_price:
            new_clients_served += 1
            new_total_revenues += bundle_price

    # Verify if the solution is better than the current one
    if new_total_revenues >= total_revenues:
        total_revenues = new_total_revenues
        prices = new_prices
        clients_served = new_clients_served
        if replaced:
            S_last_rows = new_S_last_rows
            b_last_rows = new_b_last_rows

end_time = time.time()

# Print results
for i in range(N):
    print('Price of product {} = {} €'.format( i + 1, f"{prices[i].x:.{2}f}"))

print('Total revenues of the seller = {} €'.format(f"{total_revenues :.{2}f}"))

print('Total time (CPU seconds) = {} seconds'.format(f"{(end_time - start_time):.{4}f}"))

print("Client served = {}/{}".format( clients_served, M))

print("Clients added = {}".format(clients_added))
```

Third algorithm (second heuristic algorithm):

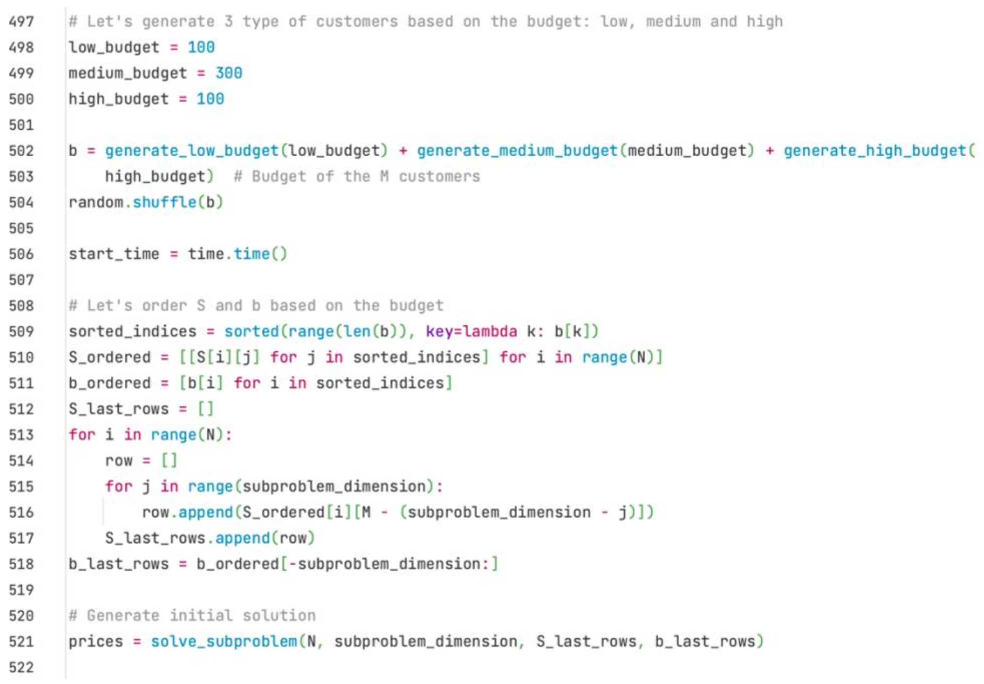


```python
from mip import *
import time
import random

def generate_matrix(rows, cols):
    total_elements = rows * cols
    num_zeros = int(total_elements * (1 - desired_density))
    num_ones = total_elements - num_zeros

    # Initialize a matrix with all elements equal to 0
    matrix = [[0] * cols for _ in range(rows)]

    # Set exactly d% of the elements to 1 at random positions (d = desired density)
    flattened_matrix = [val for row in matrix for val in row]
    for _ in range(num_ones):
        index = random.randint( a: 0, total_elements - 1)
        while flattened_matrix[index] == 1:
            index = random.randint( a: 0, total_elements - 1)
        flattened_matrix[index] = 1

    # Transform the flattened list into an NxM matrix.
    matrix = [flattened_matrix[k:k + cols] for k in range(0, total_elements, cols)]

    return matrix


def print_matrix(matrix):
    for k, row in enumerate(matrix):
        row_str = ",".join(map(str, row))
        if k == len(matrix) - 1:
            print("[", row_str, "]", end="")
        else:
            print("[", row_str, "],")

def generate_low_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 20,  b: 80) for _ in range(k)]


def generate_medium_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 80,  b: 140) for _ in range(k)]


def generate_high_budget(k):
    if k <= 0:
        return []
    else:
        return [random.randint( a: 140,  b: 200) for _ in range(k)]

```

```python
def solve_subproblem(num_items, num_customers, matrix, budget):
    S_sub = matrix
    b_sub = budget

    # Define upper bounds
    U = []
    for i in range(num_items):
        U.append(0)
        maximum = 0
        for j in range(num_customers):
            if S_sub[i][j] == 1 and b_sub[j] >= maximum:
                maximum = b_sub[j]
        U[i] = maximum

    # Define model and variables
    m = Model(sense=MAXIMIZE)
    Y = [m.add_var(var_type=BINARY) for j in range(num_customers)]
    P = [m.add_var(var_type=INTEGER, lb=0) for i in range(num_items)]
    R = [m.add_var(var_type=INTEGER, lb=0) for j in range(num_customers)]

    # Define constraints
    for j in range(num_customers):
        m += R[j] <= b_sub[j] * Y[j]  # 2B


    for j in range(num_customers):
        price_of_the_bundle = xsum(S_sub[i][j] * P[i] for i in range(num_items))
        U_bundle = 0
        for i in range(num_items):
            U_bundle = U_bundle + S_sub[i][j] * U[i]
        m += R[j] <= price_of_the_bundle  # 2C
        m += R[j] >= price_of_the_bundle - U_bundle * (1 - Y[j])  # 2D

    # Define objective function
    m.objective = xsum(R[j] for j in range(num_customers))

    # Solve it
    current_time = time.time()-start_time
    m.optimize(max_seconds=time_limit)

    return P

N = 20  # Number of items/optional
M = 500  # Number of customers
time_limit = 15

# Define subproblem dimension
subproblem_dimension = int(0.5 * M)

desired_density = 0.5

# Create matrix S
S = generate_matrix(N, M)

```



```python
# Let's generate 3 type of customers based on the budget: low, medium and high
low_budget = 100
medium_budget = 300
high_budget = 100

b = generate_low_budget(low_budget) + generate_medium_budget(medium_budget) + generate_high_budget(
    high_budget)  # Budget of the M customers
random.shuffle(b)

start_time = time.time()

# Let's order S and b based on the budget
sorted_indices = sorted(range(len(b)), key=lambda k: b[k])
S_ordered = [[S[i][j] for j in sorted_indices] for i in range(N)]
b_ordered = [b[i] for i in sorted_indices]
S_last_rows = []
for i in range(N):
    row = []
    for j in range(subproblem_dimension):
        row.append(S_ordered[i][M - (subproblem_dimension - j)])
    S_last_rows.append(row)
b_last_rows = b_ordered[-subproblem_dimension:]

# Generate initial solution
prices = solve_subproblem(N, subproblem_dimension, S_last_rows, b_last_rows)
```



```python
# Let's generate 3 type of customers based on the budget: low, medium and high
low_budget = 100
medium_budget = 300
high_budget = 100

b = generate_low_budget(low_budget) + generate_medium_budget(medium_budget) + generate_high_budget(
    high_budget)  # Budget of the M customers
random.shuffle(b)

start_time = time.time()

# Let's order S and b based on the budget
sorted_indices = sorted(range(len(b)), key=lambda k: b[k])
S_ordered = [[S[i][j] for j in sorted_indices] for i in range(N)]
b_ordered = [b[i] for i in sorted_indices]
S_last_rows = []
for i in range(N):
    row = []
    for j in range(subproblem_dimension):
        row.append(S_ordered[i][M - (subproblem_dimension - j)])
    S_last_rows.append(row)
b_last_rows = b_ordered[-subproblem_dimension:]

# Generate initial solution
prices = solve_subproblem(N, subproblem_dimension, S_last_rows, b_last_rows)
```

References

- Models and algorithms for the product pricing with single-minded customers requesting bundles - Bucarey, V., Elloumi, S., Labbé, M., Plein, F.
- A reinforcement learning approach to dynamic pricing - Monica Ferrara
- The rank pricing problem: Models and branch-and-cut algorithms - Calvete, H.I., Domínguez, C., Galé, C., Labbé, M., Marín, A.
- On profit-maximizing envy-free pricing - Guruswami, V., Hartline, J.D., Karlin, A.R., ...Kenyon, C., McSherry, F.
- Offline and online algorithms for single-minded selling problem - Zhang, Y., Chin, F.Y.L., Poon, S.-H., ...Xu, D., Yu, D.
- Algorithm Pricing: a new challenge for competition policy – Michele Angino
- Game Theory Meets Network Security and Privacy – Manshaei, Zhu, Alpcan, Basar, Hubaux
- Mixed-integer formulations for the Capacitated Rank Pricing Problem with envy - Domínguez, C., Labbé, M., Marín, A.
- Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison – Blum, Roli
- Analysis of a Problem in Product Pricing – Plein
- Algorithmic pricing – Van Loon
- A numerical optimization approach for pricing components in customer defined bundles in a B2B market - Raj, R., Karwan, M.H., Murray, C., Sun, L.
- Facility location and pricing problem: Discretized mill price and exact algorithms - Lin, Y.H., Tian, Q.
- The hub location and pricing problem - Erdoğan, G., Battarra, M., Rodríguez-Chía, A.M.
- Traveling salesperson problem with unique pricing and stochastic thresholds - Afsar, H.M.

- Vehicle routing problem with zone-based pricing - Afsar, H.M., Afsar, S., Palacios, J.J.
- The rank pricing problem with ties - Domínguez, C., Labbé, M., Marín, A.
- Mixed-integer formulations for the Capacitated Rank Pricing Problem with envy - Domínguez, C., Labbé, M., Marín, A.
- The rank pricing problem: Models and branch-and-cut algorithms - Calvete, H.I., Domínguez, C., Galé, C., Labbé, M., Marín, A.
- The newsvendor problem with capacitated suppliers and quantity discounts - Mohammadivojdan, R., Geunes, J.
- A Catalog of Formulations for the Network Pricing Problem - Bui, Q.M., Gendron, B., Carvalho, M.
- Network pricing problem with unit toll - Castelli, L., Labbé, M., Violin, A.
- On the complexity of the highway problem - Elbassioni, K., Raman, R., Ray, S., Sitters, R.
- A Network Pricing Formulation for the revenue maximization of European Air Navigation Service Providers - Castelli, L., Labbé, M., Violin, A.
- Online pricing for bundles of multiple items - Zhang, Y., Chin, F.Y.L., Ting, H.-F.
- Dynamic Pricing for Electric Vehicle Charging at a Commercial Charging Station in Presence of Uncertainty: A Multi-armed Bandit Reinforcement Learning Approach - Qureshi, U., Mushtaq, M., Qureshi, J., ...Ali, M., Ali, S.
- Matching of everyday power supply and demand with dynamic pricing: Problem formalisation and conceptual analysis - Théate, T., Sutera, A., Ernst, D.
- Algorithmic pricing via virtual valuations - Chawla, S., Hartline, J.D., Kleinberg, R.
- Models and algorithms for the product pricing with single-minded customers requesting bundles - Bucarey, V., Elloumi, S., Labbé, M., Plein, F.
- Mapping the Ethicality of Algorithmic Pricing: A Review of Dynamic and Personalized Pricing – Seele, Dierksmeier, Hofstetter, Schultz
- The Landscape of Pricing and Algorithmic Pricing – Lee
- Competitive Personalized Pricing – Chen, Choe, Matsushima

- When to introduce an online channel, and offer money back guarantees and personalized pricing? - Bintong Chen, Jing Chen
- Personalized Pricing and Quality Customization – Ghose, Huang
- Personalized Pricing and Quality Differentiation – Choudhary, Ghose, Mukhopadhyay, Rajan
- Envy, multi envy, and revenue maximization - Fiat, A., Wingarten, A.
- An Optimal Multi-Unit Combinatorial Procurement Auction with Single Minded Bidders – Gujar, Narahari
- Convex Optimization for Bundle Size Pricing Problem - Li, X., Sun, H., Teo, C.P.
- Large-scale bundle-size pricing: A theoretical analysis - Abdallah, T., Asadpour, A., Reed, J.
- Bundle-size pricing as an approximation to mixed bundling - Chu, C.S., Leslie, P., Sorensen, A.
- A data-driven approach to personalized bundle pricing and recommendation - Ettl, M., Harsha, P., Papush, A., Perakis, G.
- Pricing personalized bundles: A new approach and an empirical study - Xue, Z., Wang, Z., Ettl, M.
- Bundling with customer self-selection: A simple approach to bundling low-marginal-cost goods - Hitt, L.M., Chen, P.-Y.
- Customized bundle pricing for information goods: A nonlinear mixed-integer programming approach - Wu, S.-Y., Hitt, L.M., Chen, P.-Y., Anandalingam, G.
- A pricing scheme for combinatorial auctions based on bundle sizes - Briskorn, D., Jørnsten, K., Zeise, P.
- Online pricing for bundles of multiple items - Zhang, Y., Chin, F.Y.L., Ting, H.-F.
- A simulation-based approach to price optimisation of the mixed bundling problem with capacity constraints - Mayer, S., Klein, R., Seiermann, S.
- Combinatorial auctions: A survey - De Vries, S., Vohra, R.V.
- Combinatorial auctions in the procurement of transportation services - Sheffi, Y.
- Mixed bundling of two independently valued goods - Bhargava, H.K.

- Choosing options for products: The effects of mixed bundling on consumers' inferences and choices - Hamilton, R.W., Koukova, N.T.
- Transportation service bundling – For whose benefit? Consumer valuation of pure bundling in the passenger transportation market – Guidon, Wicki, Bernauer, Axhausen
- Pure Components versus Pure Bundling in a Marketing Channel - Girju, M., Prasad, A., Ratchford, B.T.