Ca' Foscari
University
of Venice

Master's Degree in
Data Analytics for Business and Society

Final Thesis

Production scheduling optimization: A case study from the alcohol industry

Supervisor
Ch. Prof. Raffaele Pesenti

Graduand
Leonardo Maschio
Matriculation Number 867413

Academic Year
2022 / 2023

Table of Contents

Introduction

In manufacturing industries, each job, or products being produced, can be decomposed in smaller tasks. These tasks represent specific phases of the productive process that may require different resources, machinery and processing times. A job undergoes several phases of work at different workstations, which are essentially groups of machines. Each phase is composed of one or more tasks which have to be completed on more machines. Some operations might be dependent on others and others have to respect specific time constraints. In order to comply with those constraints it is important to put in place a process called *scheduling*. Scheduling is a process that consist in to defining when each task has to be processed, by which machine and with which resources [7]. Thus, Production Scheduling is a decision-making process used to assign raw materials and resources to different processes to produce different products in the most efficient and cost effective way.

There is a vast literature on scheduling problems, and there is an endless number of different scheduling models. However, some authors suggest that the scheduling process remains very difficult to apply in practice. Wight in [6] explains how difficult it is to apply scheduling processes. He observes that the manufacturing environment is subject to constant change [6]. Forecast is never accurate and the "true dates", in which the products should be ready, are constantly shifting. Jobs are moved one ahead of the other and priorities change quickly. There can be reworks, machines that break down, tooling that does not work properly and so on. Every change in one schedule can cause changes in other schedules and this could cause a ripple effect and affect the entire production.

All these issues are very common and can happen several times a day. If companies want to maintain valid delivery dates for their products, they have to continuously factor environmental changes into their schedules. A manual system, as Wight says, cannot cope with this kind of environment. Other authors back up this statement and underline the fact that humans have not the capability to control or optimize large and complex systems [2]. Updating a schedule manually is tedious, can take a lot of time and leaves room for human errors. We cannot expect a system like this to perform well in an ever changing environment. (Wight) Despite being quite dated, Wight's argument fits perfectly into the problem we want to tackle with this thesis.

The study deals with the production scheduling process of a family run small company that produces alcoholic beverages. The company is currently ran by two brothers, who are responsible for all the activities carried out inside the company. In the last couple of years, the owners felt a growing necessity to make their activities more efficient. Time is the scarcest resource for the business and they need to use it wisely. After some analysis, the owners realized that production is indeed the most time consuming activity inside the business. This fact, is due to the high number of inefficiencies that the activity causes. Some inefficiencies are related to malfunctioning of old machinery, while others are related to the lack of a valid scheduling system. Currently, scheduling is completely done by hand and, as seen in literature, this is not the optimal way to manage the production process.

Inefficiencies during production cause two main issues inside the business. The first one is the high number of overtime hours worked, while the second one is the difficulty to manage the warehouse. There is also a third problem tied to the poor scheduling of the production process activities. The owners are not able to evaluate correctly investments on new machinery. They can have an idea on which are the benefits of a new piece of equipment, but they cannot concretely quantify them.

Therefore, the goal of this thesis will be to help the business into developing an automated production scheduling system and improve the performance of the production process. The main objective will be to create a tool that minimizes the time dedicated to production. This tool will also help with warehouse management and evaluation of new investments.

As seen in the literature, implementing scheduling techniques does not always yield optimal results. "Successful implementations of scheduling techniques in practice are still scarce […] because of the complexity of scheduling in practice, implementing these systems has created many problems" [2]. For this reason, each solution found will have to be evaluated to understand its feasibility. Even if a solution seems optimal in theory, it is not for sure that it will be reasonable in practice. If, for whatever reason, the scheduler does not precisely follow the given schedules, it could end up violating important constraints [2].

An additional important aspect to consider is that almost all decisions are made under some degree of uncertainty [3]. This means that there is the need to see if a solution has the same performances even if subjected to different conditions. A possible approach that could be used in these situations is to optimize using expected values, and then perform a sensitivity analysis. The analysis will show for which set of parameters the solution is optimal [3]. The goal in this case is to provide the business with a sort of "sandbox" environment to test the effect of different variables on the production.

The study is divided in four Chapters. Chapter 1 talks about the system and its variables. It contains a detailed description of every production step, from the preparation of the product to the storage of the finished bottles. Chapter 2 deals with the heuristics used to solve the problem. Here we will discuss the heuristics that will be used to solve the problem. We will also create a mathematical representation of the system analysed. In chapter 3 we can find the creation and implementation of the code used for the optimization process. Finally, Chapter 4 deals with the results of the optimization process and sensitivity analysis.

Chapter 1

1 Description of the system

The first important step our research is to gather information about the business activities. Therefore, this chapter contains a general description of the structure of the company, how it operates, its range of products, and its relationship with external entities. The main focus however is on the productive process. There is an accurate description of the main activities, their relationships, their constraints, and timings. All this information will be essential in the creation of the mathematical model that will be used for the optimization. This chapter also provide some information about the data gathering process. Which data we were able to collect, how we collected and organized them.

1.1 Business environment

The case study concerns a small company situated in the province of Treviso, near Conegliano. The business was founded in 1920 by Antonio Maschio, and has always been family-run since then. Nowadays, the company is run by Antonio's grandkids, Mariano and Francesco.

Initially they used to produce only a few different types of distilled alcoholic beverages, called "grappa", derived from grape must. However, the product line expanded with time and now the company offers both grappa and liqueurs. In the last two years the business started to collaborate with third parties to begin the production of gin.

1.1.1 Internal processes

The two owners are the only people working inside the company. They personally do every activity and rely on external help for accountability, shipping, extraordinary maintenance and customer management. Each owner is specialized on a specific set activities and duties, but both of them must be able to do everything anyway.

1.1.2 Suppliers

All raw materials come from businesses located in the province of Treviso, the only exception being some flavourings coming from Trieste. Keeping everything local is extremely important for the business, since it allows for a better and easier control of the supply chain. It also encourage the creation of strong relationships with the suppliers. This can lead to more stable prices and an additional will to fulfil the requests of the business. If for example However, this is not always true and in some cases, the business has to deal with products with wrong specifications or late deliveries. For this reason, the research for more favourable suppliers is continuous.

1.1.3 Consumers

Typical customers of the business are restaurants, small distributors, wine shops, bars and private customers. The demand is mainly local, but there are also many international clients. This distinction is important since due to national legislations [5], bottles of beverages sold in Italy must have an additional label on the cap.

The demand is usually quite constant throughout the year, but it typically soars between November and December. Most of the increase is due to the increasing number of private customers that load on spirits before the festivities. In fact, it has been shown that drinks consumption is significantly higher during Christmas and New Year's Eve [4]. We will see that this increase in demand has also a negative effect on the production management.

1.1.4 Products

There are three different categories of products:

1. Grappa
2. Liqueurs
3. Gin

Grappa is a beverage that comes from grape alcohol. It is a simple blend of alcohol, water and, in some cases, sugar and natural aromas. Some types of grappa might be aged in wooden barrels, a process that could take between six months and several years.

Liqueurs are usually a little more complex. They are blend of grain alcohol, water, sugar, and various natural aromas. Preparation must follow more steps and requires a lot more attention. The high content of sugar makes the handling of the product very messy. Everything that has been in direct contact with the product has to be cleaned deeply.

As said before, the business has begun a working relationship with an external entity for the production of gin. The relationship is symbiotic, the business provides the production plant, labour and know how, while the third party manages the supply, recipes and market.

1.2 Production process

The production process can involve one to two people. The number of workers usually depends on the urgency or difficulty of a job. Having two workers in production usually reduces the total completion time of a job by almost 65%. Every machine needs supervision during operation and a single worker can work on two machines at a time at most. Therefore, if there is only a single worker, most of the machines will be idle.

Between each machine there is a space called **buffer** (or storage capacity). This area contains all the outputs of a machine that are waiting to be processed in the following machine. A limited buffer between two machines can cause **blockings**. Blocking happens when the buffer is full and so the upstream machine no longer can release an output [19].

In the production line considered, blockings are a constant. Most of the machines in fact are idle during the processing of a job. When a blocking occurs, the upstream machines are stopped and the operator starts the processing on the downstream machine.

Figures 1 - 3 show a graphical representation of this situation.

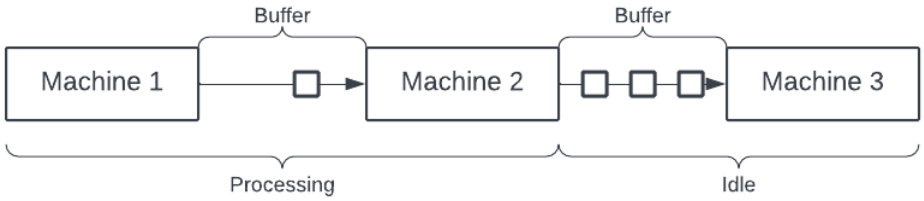

Figure 1 – Example of machine on idle

Figure 1 represents a process that requires three machines. While the worker operate Machines 1 and 2, machine 3 is idle and the buffer i_2 - i_3 (between machine 2 and machine 3) fills up, machines 1 and 2 will move to work on machine 3 (Figure 3).

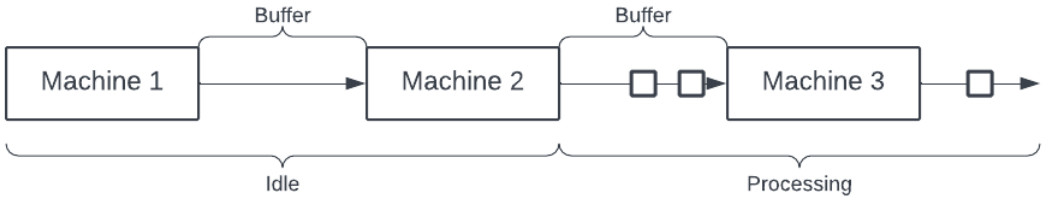

Figure 2 – Example with 2 machines on idle.

Machines are set in sequence and all jobs follow the same route. While similar jobs usually undergo the same tasks, different jobs require different transformations. The number of tasks done during the processing of a job varies based on the product type and destination.

Figure 4 shows an example of the processing of two different products (A & B). Both jobs follow the same path, but product B requires an additional transformation on machine 3.

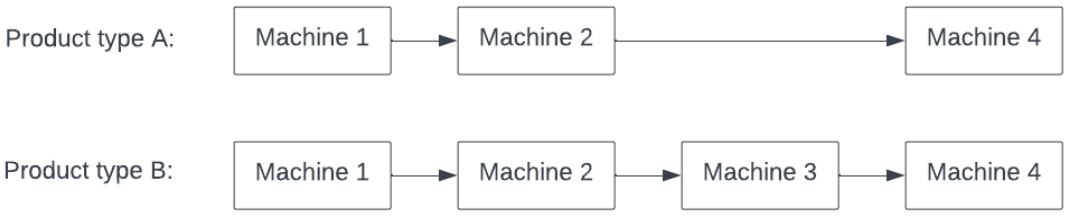

Figure 3 – Path of different products

1.3 Tasks

Before diving in to the optimization problem, it is important to understand how production works. This will allow us to detect the main interactions, decision variables and constraints inside the system. It will also help to propose solutions that are coherent with the current business abilities. In the following, we describe the different tasks involved in the process (see Figure 4).

1.3.1 PRODUCT PREPARATION

This is always the first task of the production process. It consists in two main steps, mixing the ingredients and filtering the product.

In the mixing process, the ingredients are measured and blended together inside a stainless steel tank. Preparation has to follow specific recipes for each product. The complexity of the process depends on the quantity and typology of raw materials used.
The preparation of grappa is quite easy and straightforward. The ingredients are few and errors are easily fixable. For example, adding an extra litre of alcohol can be simply fixed by adding a few litres of water to the mix.
The preparation of liqueurs and gins instead, is much more delicate. They require many different ingredients and some of them, like natural aromas, have to be measured very carefully. Adding even a couple of extra centilitres of some aroma could ruin an entire batch of product.

In theory, after the mixing process, the product is ready to be drank. However, since the alcohol has some impurities, it is necessary to filter it.
First, the product is cooled until it reaches the temperature of about -10 C°. This process crystallizes all the impurities but leaves the alcoholic mix liquid (alcohol alone solidifies at a much lower temperature). Then a specialized pump pushes the cooled product through various layers of so-called "filtering cardboards" that will retain the crystalized impurities.

Preparation is independent from all the other tasks in the process. This means that when a product is ready, it is not mandatory to start immediately the following processes. Products can be stored until the production line is ready to process it. Most of them are stored in stainless steel tanks, but some types of grappa are stored in wooden barrels for the aging process. The storing capacity is limited. Therefore, it would not be wise to create big batches of a product if the production is far in time or not even scheduled.

1.3.2 BOTTLING AND CAPPING

Bottling and capping are the first two tasks after the preparation process and are mandatory for every single product. They are carried out in two different machines that can have different configurations based on the shape of the bottles used. This means that, if two consecutive jobs require the same bottle, setup times for the machines are zero.

The bottling machine processes four bottles at a time and requires a lot of manual work. The operator has to load the bottles, wait for them to fill and unload them on a conveyor belt that brings them to the capping machine. Since the machine is very old, it needs to be regulated every 200 bottles more or less.

Capping is a completely automatic task. The machine simply puts the cap on the bottle and then closes it applying pressure with a rotating head. Machine regulation is necessary only if two consecutive jobs have different bottles.

1.3.3 CAPSULING

As capping, capsuling is a completely automatic task carried out by a single machine. It consists into putting a heat shrinkable capsule on the bottle and sealing it through a small oven.

Capsuling is not mandatory since it depends on the cork type. Some bottles require a capsule and other do not. This means that some products will entirely skip this task. Set-up times of the machines are low, but not zero, for bottles with the same shape. In fact, it is necessary to change the capsule type for every different product.

1.3.4 SEALING

Italian Customs Agency, with the newsletter n. 36/D of the 6 July 2004, regulates the duty to apply state labels on alcoholic beverages sold on the Italian territory. The newsletter states that specific alcoholic beverages are subject to a particular label called “contrassegno di Stato”. This duty concerns only products that are sold on the Italian territory.

The labels in question have to be glued on the cap (they have to break when the bottle is opened) with a glue approved by the Agency. This task is carried on by a single machine and should be, in theory, completely automated. In practice however, there is a lot of manual work involved in the process. The old machine always jams due to the strong glue. For this reason, an operator has to monitor the process constantly.

Fortunately, sealing has to be done only for products sold on the Italian market. International products will always skip this step.

1.3.5 LABELLING

Every job has to undergo labelling. The task is done by a single machine that prints the production lot on the labels and automatically applies them on the bottle. Labels can be two or one, but the number does not change the processing time. The machine does not need constant supervision, however from time to time it jams. For this reason, it is necessary to keep an eye on it constantly. After the bottles exit the labelling machine they end on a dedicated area, called *waiting area*, where they wait to be packed.

1.3.6 PACKING

Packing is the last task of the production process. When it ends, the job is considered complete and the product can be brought to the warehouse. The activity is manual and consists of two simple tasks; prepare the box and put the bottles inside it. The first task can be done at any moment during the production process. Boxes have different dimensions, but the preparation time is the same for each one of them. They can contain six or twelve bottles, so for some product types the boxing process takes more time. This second task is usually done when the waiting area of the bottles is full. When this happens, all machines are set idle and the packing task begins.

The waiting area is limited and so can hold a specific number of bottles. Therefore, packing has to be done more than one time during the processing of each job.

1.3.7 FULL SYSTEM

The entire production process is schematized in *Figure 4*. Each box represents a specific task and arrows show the path followed by the bottles. Continuous arrows represent mandatory paths, while dotted arrows represent paths that are taken only by specific product types.

The arrows also correspond to the buffers present in the system: m_1 - m_2, m_2 - m_3, m_2 - m_4, m_2 - m_5, m_5 - $m_{packing}$. All of these are limited buffers, so a blocking could potentially happen everywhere in the system. The only exception is the buffer m_1 - m_2. Machines m_1 and m_2 are interdependent, and so when machine $m_1$is operational, machine m_2 has to be operational too. Blockings usually occur when there is a single operator on the line. When the operators are two, the machines are almost continuously operational. However, this does not happen frequently. Keeping two workers in the production line means leaving other activities frozen.

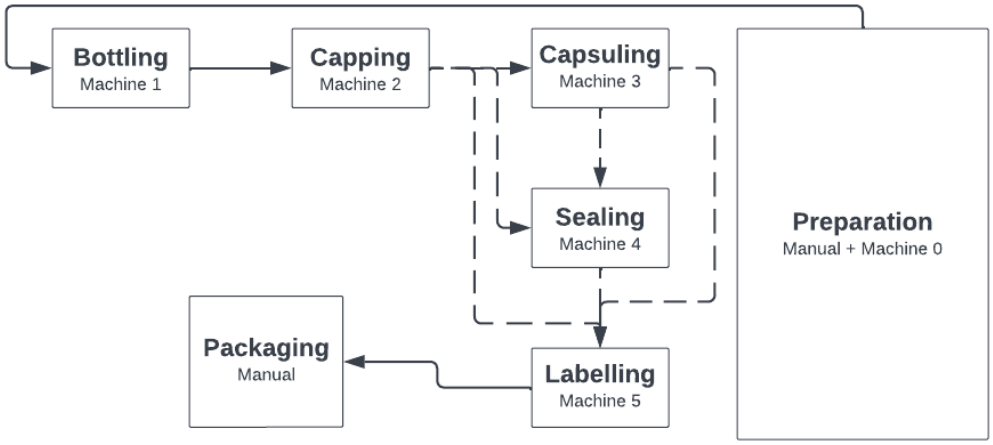

Figure 4 – Scheme of the production line.

A single job can undergo up to seven different tasks, and once a bottle has visited one of these stages, it cannot undergo the same task again. The number of transformations varies based on the product type, but the path followed is always the same. Each bottle will pass through each machine, even if it does not undergo any transformation. Similar jobs will undergo the same tasks and will have the same processing time.

The path of the jobs is not always continuous. In fact, after a job starts its processing, it is not mandatory to complete it. In some cases, it can be more convenient to stop the transformation and resume it another moment. However is important to note that once a job has been processed through a machine i_m, it cannot be processed on the same machine again.
Reworks are common and could be done for several reason, such as:

- **Unknown destination of the product**: In this case, the job stops before the sealing task.
- **Changing in priorities**
- **Processing time of a job is too big**: in this case the job should be finished in more sessions.

- **Machinery breakdown**
- **Missing raw materials**

1.4 Main problems

In the last few years, the business has been able to increase its customer base consistently. The owners attribute these results to the expansion of the range of products and stronger investments in marketing. This growth is certainly positive from a monetary point of view, but it is problematic from an organizational point of view. During specific periods of the year, November in particular, the demand becomes so high that the business barely manages to fulfil it. Usually, the demand starts to rise around the second half of November as shown in Figure 5. The plot shows the trend of the yearly demand of all products. As we will discuss in Chapter 2 this demand is calculated by taking the mean of the orders received in the last three years.

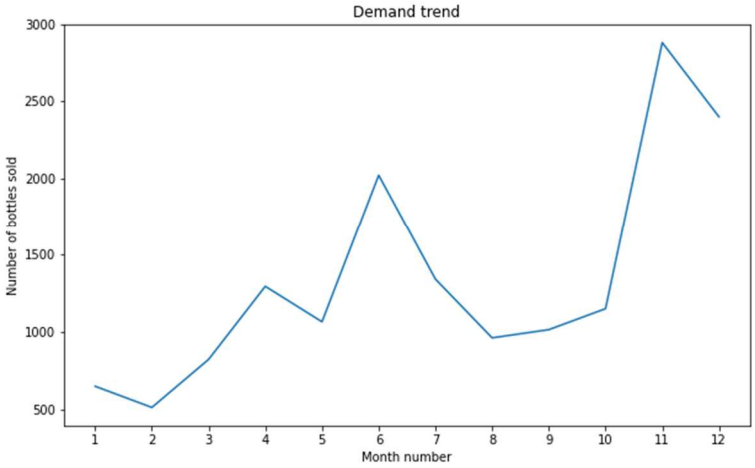

Figure 5 - Demand trend by month

The increase in demand is due to customers who want to buy gifts for relatives, friends, employees, and external collaborators. However, presents require additional personalization of the finished products. The company offers wood cases for bottles, baskets with mixed products, or various wrappings. Despite producing a higher economic margin, customizations require many additional working hours. These hours, added to the extra time spent in production, are a recipe for overtime hours. Based on Assumption 4 and Assumption 5, in order to limit overtime hours, it is necessary to lower the hours spent during production. One of the solutions to do that is to increase the storage during previous months. Luckily, this is an exception and during the rest of the year, production is usually easier to manage.

Since there are no employees in the company, the owners must try to balance the time dedicated to production with the time dedicated to other daily activities. Usually, this leads to a large number of overtime hours. The average working day is about ten hours long, and can even reach thirteen hours during periods of high demand. This leads to fatigue that in turn leads to poor performances. For this reason, there is a constant push into trying to reduce the time dedicated only to production. This reduction, say the owners, can be done in three different ways: **optimize the schedules**, invest in **new machinery/workforce**, or **increase the warehouse capacity**.

Currently, the focus is on the schedule optimization solution. This process is entirely done by hand and for this reason, it is slow and inefficient. There is also the will to modernize the machinery. However, as said in the introduction, for the business is difficult to determine the opportunity cost

of new investment. As of now, due to the high costs, there is no interest in increasing the workforce permanently. Increasing the warehouse capacity would allow producing bigger batches of product. This lowers the total set-up times and helps the business to deal better with unexpected orders. With this analysis, we will tackle the first two of these problems. Firstly, we want to find a way to optimize production scheduling and minimize the total production time. The second objective is to help the business in the investment assessment process. We will do this through more sensitivity analysis.

1.5 Storage

Once a job has been completely processed, the product is stored in a warehouse located inside the plant. The storing process has three main constraints that will have to be taken into consideration: **limited capacity, cost of the warehouse and deposits**.

The first two are self-explanatory, production cannot exceed the warehouse capacity and warehouse has a fix cost for each bottle.

The last one is a little more complicated to explain. D.Lgs. n. 504/1995 requires four different deposits when managing alcoholic beverages:

- storage of sealed products (art. 5);
- circulation of sealed products (art. 6);
- loose state labels (art. 13);
- storage of loose (not labelled) products.

Those deposits are a monetary sum that the business has to "freeze" to ensure the payment of excise duties even the case of bankrupt. They also represent the maximum amount of state labels, alcohol or finished products that can be kept inside the plant. If for example the deposit on loose labels is 20.000 €, the economic value of all the labels found inside the plant cannot be more than 20.000 €.

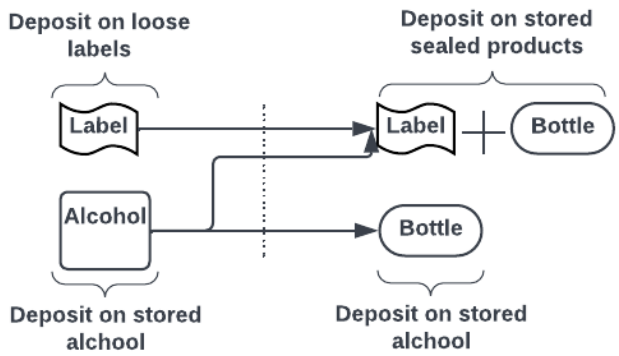

Figure 6 – Distribution of the deposits

Figure 6 shows a representation of the correlations between each deposit. In the production process, there are two deposits, on loose labels and one on stored alcohol. As soon as the label is applied to the bottle, the deposit on loose labels is "freed" and the deposit on stored sealed products is filled. If the alcohol is bottled without label, the deposit on stored alcohol will not be freed.

Chapter 2

2 Problem Formulation

In this Chapter we *formulate* the optimization problem. This means translating the real-world problem into mathematical equations. Optimization problems have three main components: an objective function, decision variables, and constraints [8]. The goal of this chapter is to define those three components.

2.1 Problem statement

The problem of interest is one of minimization. The main objectives of the analysis are three:

- minimize the number of daily overtime hours,
- minimize the total weekly hours dedicated to production, and
- minimize warehouse costs.

Since demand varies a lot during the year, we expect the optimums to fluctuate during the year. In particular, we expect the warehouse costs to soar before periods of high demand and then immediately drop. On the other hand, we expect the hours dedicated to production to increase right before and during periods of high demand and decrease after. The objectives proposed should be able to smooth all those fluctuations as much as possible.

The period considered for the analysis is one year. Demand is fairly constant throughout the year, whit some small spikes occurring randomly. However, as mentioned in Chapter 1, before the Christmas holidays there is always a huge increase in demand.

This particular time frame (November – December) could be a great test to understand if the model built can find a good balance between products stored and extra hours worked.

2.2 Mathematical model

In the following model is described in terms of:

- **Sets**: sets of entities relevant for the model.
- **Notations**: shorthand representations used to refer to entities of the model.
- **Parameters**: values that represent specific properties of the model.
- **Decision variables**: inputs to the model that can take on specific values based on the parameters they refer to.
- **Auxiliary variables**: variables that are introduced to simplify the formulation of the problem. They help to describe the relationships among the variables in the model.
- **Constraints**: restrictions that are imposed on the possible values that the decision variables can take.
- **Objectives**: goals of the decision-making process.
- **Assumptions**: conditions that hold for the problem being analyzed.

PROBLEM TYPE

Deterministic problem involving a single decision maker

SETS

$\boldsymbol{Products}$ = set of products produced by the company. In total there are 15 different products considered.

$\boldsymbol{Weeks}$ = set of weeks analyzed. Owners want to be able to see 6 weeks of scheduling in total. Taking a bigger timeframe can be misleading since forecast becomes less reliable. During our analysis we will start scheduling from week 43. This will allow us to tee how the model performs in periods of high demand.

$\boldsymbol{Days}$ = days with production. Ideally, production should be done three or less day a week. Remaining days will be dedicated to product preparation and other activities.

$\boldsymbol{Tasks}$ = set of tasks present in the system.

$\boldsymbol{Mag}$ = set of weeks of the warehouse.

$\boldsymbol{Tanks}$ = set of tanks for the storage of the product.

NOTATIONS

$\boldsymbol{u}$ = single product $\in Products$.

$\boldsymbol{w}$ = single week $\in Weeks$.

$\boldsymbol{d}$ = single day $\in Days$.

$\boldsymbol{i}$ = single task $\in Tasks$.

$\boldsymbol{c}$ = single tank $\in Tanks$.

PARAMETERS

Bottle processing time ($\boldsymbol{bt_{(i,u)}}$) Processing time of a bottle of product *u* on task *i.* It represents the total time needed to process a single bottle through a machine.

Waiting time ($\boldsymbol{wt_{(i,u)}}$) Waiting time of a bottle of product *u* on machine *i*. A single machine can process multiple bottles at the same time. Since they are processed in sequence, bottle $\boldsymbol{u_x}$ can start its processing only after bottle $\boldsymbol{u_{x-1}}$. We call waiting time the delta time between the beginning of the processing of bottle $\boldsymbol{u_x}$ and the beginning of the processing of bottle $\boldsymbol{u_{x-1}}$.

Buffer capacity ($\boldsymbol{B_{(i,u)}}$) Number of bottles of product *u* that the buffer preceding task *i* can hold. The capacity of each buffer depends on the product type.

Setup time ($\boldsymbol{st_{(i,u)}}$) Setup time of machine *m* to process product *u*.

Warehouse cost ($\boldsymbol{cw_{(u)}}$) Cost of keeping one bottle of product *u* in the warehouse for one day.

Demand ($\boldsymbol{de_{(w,u)}}$) Demand of product *u* on week *w*.

Maximum quantity ($\boldsymbol{mq_{(u)}}$) Maximum quantity of product *u* that can be produced in a day without working extra hours.

Tanks capacities ($\boldsymbol{cc_{(c)}}$) Maximum quantity of product that can be stored in tank *c*.

Bottles capacities ($\boldsymbol{bc_{(u)}}$) Quantity of product *u* contained in a bottle.

Cleaning time ($\boldsymbol{cb_{(u)}}$) Time necessary to clean the line after the production of batch of product.

$\boldsymbol{limit}$ **=** Maximum number of hours that can be worked in a single day (working hours + overtime). The limit is set to 10 hours daily.

$\boldsymbol{HW}$ = Total working hours in a day. It has been decided to set this value to 8 hours.

$\boldsymbol{Max\_w}$ = Maximum amount of bottle that can be stored.

DECISION VARIABLES

Processing time ($\boldsymbol{pt_{(d,w,u)}}$) Non-negative real variable whose value represents the total time necessary to process an entire batch of product *u*.

Daily production quantity ($\boldsymbol{q_{(d,w,u)}}$) Non-negative integer variable whose value represents the number of bottles of product *u* produced on day *d* of week *w*. We also refer to it as the dimension of the batch of product *u* produced on day *d*.

Warehouse quantity ($\boldsymbol{M_{(w,u)}}$) Non-negative integer variable whose value represents the number of bottles of product u stored in the warehouse during week w.

Overtime hours ($\boldsymbol{otw_{(d,w)}}$) Non-negative real variable whose value represents the total overtime hours worked on day d of week w.

Quantity of product stored ($\boldsymbol{Q_{(c,w,u)}}$) Non-negative integer variable whose value represents the quantity of product u stored in tank c on week w.

Quantity of product produced ($\boldsymbol{Qq_{(w,u)}}$) Non-negative integer variable whose value represents the quantity of product u produced on week w.

AUXILIARY VARIABLES

Daily binary bottle production ($\boldsymbol{bp_{(d,w,u)}}$) Binary variable used to define if bottles of product u are produced on day d.

Binary quantity produced ($\boldsymbol{Qp_{(w,u)}}$) Binary variable used to define if product u is produced on week d.

Binary quantity stored ($\boldsymbol{Qc_{(c,w,u)}}$) Binary variable used to define if product u is stored on tank c in week w.

CONSTRAINTS

Constraint 1:

$$bp_{(d,w,u)} \in \{0,1\} \quad \forall\, u \in Products, \forall\, w \in Weeks, \forall\, d \in Days$$

Constraint that defines the variable $\boldsymbol{bp_{(d,u)}}$ as binary.

Constraint 2:

$$mq_{(u)} * bp_{(d,w,u)} \geq q_{(u,w,d)} \qquad \forall\, u \in Products, \forall\, w \in Weeks, \forall\, d \in Days$$

This constraint is used to set the value of the variable $\boldsymbol{bp_{(d,u)}}$**.** When there is production of bottles product *u* on day *d,* variable $\boldsymbol{q_{(p,d)}}$ is greater than 0 and so $\boldsymbol{bp_{(d,u)}}$ has to be equal to 1. If instead there is no production, $\boldsymbol{q_{(p,d)}}$ is 0 and so $\boldsymbol{bp_{(d,u)}}$ must be equal to 0.

Constraint 3:

$$pt_{(d,w,u)} = \sum_{i \in Tasks} \left(wt_{(i,u)} + \frac{bt_{(i,u)}}{B_{(i,u)}} \right) * q_{(d,w,u)} + \sum_{i \in Tasks} st_{(i,u)} * bp_{(d,w,u)} + cb_{(u)} * bp_{(d,w,u)}$$

$$\forall u \in Products, \qquad \forall d \in Day, \qquad \forall b \in Buffers$$

Processing time of a batch of product *u* on day *d*. The constraint is defined by two different summations. The first one aggregates the completion times of all tasks carried out during production. The total time necessary to process product *u* on task *i* and day *d,* is calculated as follows*:*

$$wt_{(i,u)} * q_{(d,w,u)} + \frac{bt_{(i,u)}}{B_{(i,u)}} * q_{(d,w,u)} \tag{1}$$

The first half of the expression ($\boldsymbol{q_{(d,w,u)} * wt_{(i,u)}}$) represents the cumulative waiting time for task *i* of all bottles *u* in the batch. As said in Chapter 1, a machine starts to execute its task when the buffer before it is full and it can process more bottle simultaneously. For this reason, the processing time $\boldsymbol{bt_{(i,u)}}$ is considered once every time buffer is freed. The last part of the expression, $\frac{\boldsymbol{bt_{(d,u)}}}{\boldsymbol{B_{(i,u)}}} \boldsymbol{q_{(d,w,u)}}$, returns the number of times buffer $\boldsymbol{B_{(i,u)}}$ is freed during production of one batch of product *u*. This value is then multiplied by the completion time of task *i* for a single bottle *u.*

Figure 6 schematizes the interaction between the variables present in (1). The image shows a batch of three bottles undergoing the process *i.* In the example we also assume that the buffer before the task is also equal to three bottles.

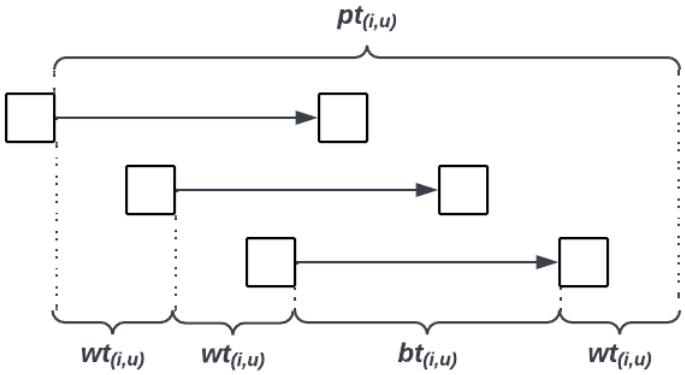

Figure 5 – Performing task i on three bottles of product u

Finally, after production, the line needs to be cleaned. Cleaning time depends on the product type and is calculated as follows (2):

$$cb_{(u)} * \ bp_{(d,w,u)} \tag{2}$$

Constraint 4:

$$otw_{d,w} \geq 0 \quad \forall w \in Weeks, \forall d \in Day$$

Constraint 5:

$$otw_{d,w} \geq \sum_{u \in Products} pt_{(d,w,u)} - HW$$

$$\forall\, u \in Products, \forall\, w \in Weeks, \forall\, d \in Days$$

Overtime hours of production worked are equal to the total time spent during production, minus the standard working hours in a day. Given Constrain 4, if $pt_{(d,w,u)} \leq HW \rightarrow otw_{d,w} = 0$.

Constraint 6:

$$limit \ \geq \ otw_{(d,w)} + HW \quad \forall\, d \in Day, \forall\, w \in Week$$

The owners decided to define a limit on the of hours that can be worked in a single day. This constraint ensures the limit will never be crossed.

Constraint 7:

$$\sum_{u \in Products} M_{(d,w,u)} \ \leq Max\_w$$

$$\forall\, w \in Week, \forall\, w \in Week$$

The number of bottles stored in a week cannot exceed the maximum capacity of the warehouse.

Constraint 8:

$$\sum_{d \in Days} q_{(d,w,u)} = d_{(w,u)} - \ M_{(u,w-1)} + M_{(u,w)}$$

$$\forall\, u \in Products, \forall\, w \in Weeks, \forall\, d \in Days$$

The production of a given product in a given week must be equal to the demand of the product minus the warehouse quantities of the preceding week plus the warehouse quantities at the end of the week.

Constraint 9:

$$M_{(d,0,u)} = 0 \quad \forall\, u \in Products, \forall\, d \in Days$$

Assumption that tells that the initial stored quantities of bottles of each product are equal to 0.

Constraint 10:

$$Qq_{(w,u)} \leq Qp_{(w,u)} * 1500 \quad \forall\, u \in Products, \forall\, w \in Weeks$$

This constraint is used to set the value of the binary variable $Qp_{(w,u)}$**.** If product *u* is produced during week *w*, $Qq_{(w,u)}$ will be greater than 0 and so $Qp_{(w,u)}$ must be equal to 1. If instead $Qq_{(w,u)}$ is equal to 0, also $Qp_{(w,u)}$ will be equal to 0. The value 1500 represents the maximum amount of product that can be produced in a day.

Constraint 11:

$$\sum_{u \in Products} Q_{(c,w,u)} \leq cc_{(c)}$$

$$\forall\, c \in Tanks, \forall\, w \in Weeks$$

The quantity stored in a tank *c* cannot exceed the maximum capacity of the tank.

Constraint 12:

$$\sum_{c \in Tanks} Q_{(c,w,u)} = Qq_{(w-1,u)} - \Big(\sum_{d \in Days} q_{(d,w,u)}\Big) * bc_{(u)} + \sum_{c \in Taks} Q_{(c,w-1,u)}$$

$$\forall\, u \in Products, \forall\, w \in Weeks$$

The total quantity of product *u* stored on week *w* depends on the quantity of product produced on week *w-1*, the quantity of product bottled and the quantity of product stored at the end of week *w-1*. The constraint takes into account *Qq* of the previous week because product need to rest a few days before being bottled.

Constraint 13:

$$\sum_{c \in Tanks} Q_{(c,0,u)} = 200 \qquad \forall\, u \in Products$$

The owners always keep at least 200 liters of product stored. This allows production to be more reactive in the case of unexpected orders.

Constraint 14:

$$\sum_{c \in Tanks} Q_{(c,w,u)} \geq Q_{(c,w,u)}$$

$$\forall\, u \in Products, \forall\, w \in Weeks, \forall\, c \in Tanks,$$

This constraint tells to the model that the quantity of product stored in a single tank cannot exceed the total quantity of product stored. If for example product *u* is stored both in thank 1 and thank 2, $Q_{(1,w,u)}$ must necessarily be $\leq Q_{(1,w,u)} + Q_{(2,w,u)}$.

Constraint 15:

$$Q_{(c,w,u)} \leq Qc_{(c,w,u)} * cc_{(c)} \quad \forall\, u \in Products, \forall\, w \in Weeks$$

Constraint is used to set the value of the binary variable $Qpc_{(c,w,u)}$, as in Constraint 2 and Constraint 11.

Constraint 16:

$$\sum_{c \in Tanks} Qc_{(c,w,u)} \leq 1 \quad \forall\, u \in Products, \forall\, w \in Weeks$$

This constraint ensures that each tanks stores only one type of product.

OBJECTIVES

Objective 1:

$$\min(\max(otw_{d,w}))$$

Objective created to minimize the maximum amount of overtime hours worked in a day.

Objective 2:

$$\min\left(\sum_{d \in Days, w \in Weeks, u \in Products} \boldsymbol{pt}_{(\boldsymbol{d},\boldsymbol{w},\boldsymbol{u})}\right)$$

Objective created to minimize the overall number of overtime hours worked. To solve this objective, all constraints are applied. The constrain (4) and (6) will be replaced by constrain (5).

Objective 3:

$$\min\left(\sum_{u \in Products, w \in Weeks} cw_{(u)} * M_{(w,u)}\right)$$

Objective created to minimize the monthly cost of the warehouse.

ASSUMPTIONS

For the sake of our analysis, we will hold the following assumptions.

Assumption 1:

The business must be able to meet 110% of the predicted demand each month. This provides a sort of buffer in case of unexpected orders.

Assumption 2:

Demand for each month is known in advance.

Assumption 3:

Data relative to the demand is monthly based and relative to a single year. Future demand has a behaviour similar to the demand of the previous years, so the data used represents the mean demand last 3 years (2020 – 2022).

Assumption 4:

The number of daily hours dedicated to the customization of products is fixed.

Assumption 5:

Customization can be done only on finished products.

Assumption 6:

Production is carried on only by one person.

Assumption 7:

Initial quantities stored of each product are 0.

Assumption 8:

There can be production at most three days a week.

Assumption 9:

Cost of the warehouse for a single bottle has 2 components. The first one represent the price for the monthly rent of the warehouse plus insurance expenses. The second one represent the cost of keeping the goods immobilized.

2.3 Multi-objective optimization problems

Optimization problems can be categorized based on the number of objective functions to be optimized. There can be single-objective optimization problems (SOOPs) and multi-objective optimization problems (MOOPs) [9]. The main goal of SOOPs is to find the single best solution for a specific criterion or metric [10]. MOOPs, on the other hand, have multiple objective functions that need to be calculated simultaneously. These objective functions often contradict each other. The best solution for a objective function could be a poor solution for another one. Therefore, MOOPs have a set of multiple solutions, not a single one. The main issue with MOOPs is that it is difficult to define the solutions [9]. A solution to this problem is to try to get solutions close to the so-called “Pareto-

optimal front". In literature, the Pareto front is defined as "*a set of solutions that are non-dominated to each other but are superior to the rest of solutions in the search space*" [11]. This simply means that there is not a single solution preferable to all other solutions. Each change in the variables of the problem cannot improve all objectives simultaneously.

Since it has three objective functions, our model belongs to the family of multi-objective optimization problems. In order to find solutions within the Pareto front, we decided to implement two Multi-objective optimization methods: "Weighted Sum Method" and "ε-Constraint method". Both of these methods belong to the family of Multi-objective Trade-off Optimization Methods. These kinds of methods deal with MOOPs by transforming them into SOOPs. They do this by defining the importance degree of each objective [9].

2.3.1 Weighted Sum Method

This method merges multiple objectives into one single objective. For *m* optimization objectives $f_m(x)$ the Weighted Sum Method aggregates objectives by weighting them and summing them:

$$f(x) = min \sum_{i=1}^{m} (w_i * f_i(x)) \tag{3}$$

where w_i are the weight coefficients of the objectives. Those coefficients have to be decided beforehand by the decision-makers based on experiences and real-world problems [9]. This method allows us to give different levels of importance to different objectives. Consequently the model will return a solution on the Pareto front in line with the owner's goals.

2.3.2 ε-Constraint method

The constraint method consists of selecting some objectives to be constraints of the optimization problem. From a set of objectives $f_m(x)$ the decision-makers have to select one objective to optimize while the others will become constraints of the system.

$$\min f(x) \rightarrow \min f_k(x) \quad s.t. f_m(x) \leq \varepsilon_i, 1 \leq m \leq i, m \neq k \tag{4}$$

where ε_i are the upper bounds of the objective functions (now constraints) that have been estimated by the decision-makers [9]. This method can be very useful if the decision-makers know exactly the range of values that the objective functions should have.

2.3.3 Implementation

We discussed the implementation of the two methods with the company owners. Both owners agree that daily overtime should not exceed two hours. For this reason, it is reasonable to apply the ε-Constraint method to *Objective 1* (5).

$$\min\left(max\left(otw_{d,w}\right)\right) \rightarrow s.t. \ otw_{d,w} \leq 2, \forall\, d \in Day, \forall\, w \in Week \tag{5}$$

The objective has been transformed into a constraint. However, *Constraint 5* already specifies the lower bound of $otw_{d,w}$. Therefore we decided to simply add to the constraint the upper bound of $otw_{d,w}$:

$$0 \leq otw_{d,w} \leq 2 \quad \forall w \in Weeks, \forall d \in Day$$

However, the optimization problem is still multi-objective. To transform it into a SOOP we opted for the implementation of the Weighted Sum Method. This method combines the two remaining objective functions (6), thus generating a single-objective optimization problem.

$$\min(w_1 * \sum_{d \in Days, w \in Weeks, u \in Products} \boldsymbol{pt}_{(\boldsymbol{d},\boldsymbol{w},\boldsymbol{u})} + w_2 * \sum_{u \in Products, w \in Weeks} cw_{(u)} * M_{(w,u)}) \quad (6)$$

We decided to use this last method for two main reasons. The first one is that it is fast and easy to implement. The second reason is that the company owners already have a clear idea on the values of the two weights w_1 and w_2. The most important thing for the business is to save time whenever possible. Therefore, the minimization of the total overtime has a bigger priority than the minimization of warehouse costs. For this reason we decided to set w_1 as 0,8 and w_2 as 0,2.

2.4 Full Model

Objective:

$$\min(w_1 * \sum_{d\in Days, w\in Weeks, u\in Products} \boldsymbol{pt_{(d,w,u)}} + w_2 * \sum_{u\in Products, w\in Weeks} cw_{(u)} * M_{(w,u)})$$

Constraints:

$bp_{(d,w,u)} \in \{0,1\} \quad \forall u \in Products, \forall w \in Week, \forall d \in Days$

$mq_{(u)} * bp_{(d,w,u)} \geq q_{(d,w,u)} \qquad \forall u \in Products, \forall w \in Week, \forall d \in Day$

$pt_{(d,w,u)} = \sum_{i \in Tasks} \left(wt_{(i,u)} + \frac{bt_{(i,u)}}{B_{(i,u)}} \right) * q_{(d,w,u)} + \sum_{i \in Tasks} st_{(i,u)} * bp_{(d,w,u)} + cb_{(u)} * bp_{(d,w,u)}$

$\forall u \in Products, \forall w \in Weeks, \forall d \in Days, \quad \forall b \in Buffers$

$otw_{d,w} \geq \sum_{u\in Products} pt_{(d,w,u)} - HW \quad \forall u \in Products, \forall w \in Weeks, \forall d \in Days$

$0 \leq otw_{d,w} \leq 2 \quad \forall w \in Weeks, \forall d \in Day$

$limit \geq otw_{(d,w)} + HW \quad \forall d \in Day, \forall w \in Week$

$\sum_{u\in Products} M_{(u,w)} \leq Max\_w \qquad \forall w \in Week$

$\sum_{d \in Days} pt_{(u,d)} \geq d_{(w,u)} - M_{(u,w-1)} + M_{(u,w)} \qquad \forall w \in Weeks, \forall u \in Products$

$M_{(u,0)} = 0 \quad \forall u \in Products$

$Qq_{(w,u)} \leq Qp_{(w,u)} * 1500 \quad \forall u \in Products, \forall w \in Weeks$

$\sum_{u \in Products} Q_{(c,w,u)} \leq cc_{(c)} \quad \forall c \in Tanks, \forall w \in Weeks$

$\sum_{c \in Tanks} Q_{(c,w,u)} = Qq_{(w,u)} - \sum_{d \in Days} q_{(d,w,u)} + \sum_{c \in Taks} Q_{(c,w-1,u)} \forall u \in Products, \forall w \in Weeks$

$\sum_{c \in Tanks} Q_{(c,0,u)} = 200 \qquad \forall u \in Products$

$\sum_{c \in Tanks} Q_{(c,w,u)} \geq Q_{(c,w,u)} \qquad \forall u \in Products, \forall w \in Weeks$

$Q_{(c,w,u)} \leq Qc_{(c,w,u)} * cc_{(c)} \quad \forall u \in Products, \forall w \in Weeks$

$\sum_{c \in Tanks} Qc_{(c,w,u)} \leq 1 \qquad \forall u \in Products, \forall w \in Weeks$

2.5 Data Gathering

All Data used for this analysis has been gathered manually. The main information we will work with are demand levels and production timings.

Data about the demand follows Assumption 3. This approach to calculating the expected demand is similar to that of the owners. They plan the production both on current orders and on the demand of previous years. These orders are stored inside the management software of the business and are easily downloadable in an XLSX format. Output data has a tabular structure. Rows contain information about a single order. Columns show the variables relative to each order (client ID, products, quantities, etc.). Figure 6 shows an example of the structure of the tables.

Date	Doc N.	Client	Product	Bottles
07 gen 2021	AB-1	MARKETSHOP SRL	PRODUCT 1	240,00
12 gen 2021	AB-2	AISLES SRL	PRODUCT 1	18,00
12 gen 2021	AB-2	AISLES SRL	PRODUCT 3	6,00
12 gen 2021	AB-3	PAOLO ROSSI	PRODUCT 7	90,00
12 gen 2021	AB-4	WINE SHOP SRL	PRODUCT 2	180,00
12 gen 2021	AB-4	WINE SHOP SRL	PRODUCT 4	18,00

Figure 6 - Example of orders table

For the sake of the analysis, we dropped client ID and document number from the tables. We then extracted the week by the Date column. Finally, the data has been grouped by week - product while summing the number of bottles. This operation has been done for all data from 2020, 2021 and 2022. The three resulting tables have then been merged together on week number and product while calculating the mean of the number of bottles. An example of the final table is shown in Figure 7. Data has been processed using Python and stored in an .XLSX file.

Week	Product	Demand
1	Product A	240
1	Product B	50
1	Product C	20
2	Product A	100
2	Product B	200
...		

Figure 7 - Example of demand data table

All data about timings has been collected directly during production. The measurements were taken during the processing of four different jobs. All the information were then manually organized in tables inside an excel file as in Figure 8. Each row is relative to a specific set product – operation and stores information about the timings considered. The first two columns refer to the couple product-operation. For each one of this couple we have information about the waiting time before the processing (bt), proceeding time (wt), set-up time (st) and buffer capacity (B). Consider for example the second row of the table in Figure 8.

Data about warehouse costs is based on Assumption 9. It has been re-evaluated specifically for this problem by the owners of the company. The value ranges between 0.20 € and 0.35 € a week for a single. The exact value depends on the product type. This type of data is not much and probably will

have to be re-evaluated again in the future. For these reasons, it will be manually added to our program, and will not be stored in additional files.

Product	Operation	bt (s)	wt (s)	st (s)	B (N. bottles)
Product A	Bottling	35	0	1200	4
Product A	Capping	9	2	600	4
Product A	Capsuling	10	2	600	20
...					
Product B	Bottling	39	0	900	4
Product B	Capping	7	2	900	4
...					

Figure 8 - Example of production timings table

Chapter 3

3 Implementation

This Chapter deals with the implementation of the mathematical model inside the solving software. In the first section, there will be a short description of the software and libraries used for the optimization. To run the optimization algorithm we first need to translate the mathematical model to Python code. In the second section of the Chapter, we show how to load a new model in the Pyomo environment. We will also show how to extract results and transform them into a tabular format. At the end of the Chapter, there is a more in-depth study of the algorithm used for the optimization.

3.1 Software used

For the optimization process, we will make use of Python. The vast number of libraries available for the software will give us a lot of flexibility when programming. There is also a lot of documentation that explains how to work with those libraries. Moreover, there are many online forums and websites (StackOverflow, geeksforgeeks, …) full of concrete examples, solutions, and additional pieces of information. All this support will make the programming process much easier. We will be able to code and troubleshoot more rapidly.

The main Python libraries that we will use are **Pandas** and **Pyomo**.

Pandas is a Python library primarily used for data manipulation and analysis. It provides data structures and functions useful for working with structured data. It also comes in handy when working with tabular data, such as that found in spreadsheets or CSV files. For this reason, we will use it to read the data collected and write the solutions in an "easy to read" format.

Pyomo is a Python library used for modeling optimization problems. It can support linear and nonlinear programs, mixed-integer and mixed-binary programs, and stochastic programs. We will use it to write and solve the optimization problem.

To find the solution the program will use CBC solver. CBC ("COIN–OR" Branch and Cut) is an open-source mixed-integer program solver developed by John Forrest [12]. As the name suggests it uses the Branch and Cut algorithm to solve linear programming problems. Luckily, the solver is fully implemented in Pyomo and can be called easily with a function.

3.2 Python implementation

3.2.1 Model creation

A new model is built in Pyomo through the function `ConcreteModel()`:

```
model = ConcreteModel()
```

In this example we are building a new model and calling it "model".

Pyomo allows for the building of either **concrete models** (`ConcreteModel()`) or **abstract models** (`AbstractModel()`)

Abstract models rely on unspecified parameter values. An example of an abstract model is the model presented in Chapter 2.4. Here we can see that there are no indications on the values of the parameters.

Concrete models are defined with the data values indicated at the moment of the definition. When we build a new model in our program, we already know the values of the parameters. For this reason we have to build a concrete model.

3.2.2 Loading elements of the problem

SETS

Sets are loaded to the program as lists of elements.

```
# Tasks
Tasks = ['1', '2', '3', '4', '5', '6']
```

In the example above, we are creating a set called "Tasks" containing the name of all task in the process.

PARAMETERS

Parameters are entities related to one or more sets of the optimization problem. They store the value of a specific element of a set or of a combination of elements from different sets. As an example, $\boldsymbol{wt_{(1,PE)} = 3s}$ means that the waiting time of a bottle of product PE for the task 1 is 3 seconds.

Parameters are stored in or code as Python dictionaries. The **keys** of these dictionaries can be of two types:

1. A single string, which refers to a single element of a set, e.g. $c \in Tanks$. Coded as:

   ```
   # Tanks maximum capacity
   cc = {'c1':1500, 'c2':300, ...}
   ```

   Where $cc_{(c)}$ is the parameter and c1 - c2 are elements of the set Tanks. The numbers represent the values that the parameter $cc_{(c)}$ has for a specific element. The parameter in the example assigns the value 1500ℓ to the maximum capacity of tank c1 and 300ℓ to tank c2.

2. A tuple containing two or more elements from different sets, e.g.$(i \in Tasks, u \in Products)$. An example of the code is:

   ```
   # waiting times
   wt = {('1', 'PE') : 0, ('2', 'PE') : 2.5, ...}
   ```

   Where the tuples indicate the elements pair considered. As before, numbers represent the values that the parameter $wt_{(i,u)}$ has for a specific couple of elements. As an example, key `('2', 'PE')` assigns assign value 2.5 to parameter $wt_{(2,PE)}$, meaning that the waiting time of a bottle of product PE on machine 2 is 2.5 seconds.

The **values** of the dictionary, as just shown, corresponds to the value that we want to assign to a specific parameter.

VARIABLES

Variables must be written in Pyomo language and are implemented with the function `Var()`. Unlike parameters, variables must be declared inside the Pyomo model in the following way:

```
# Daily production quantity
model.q = Var(Days, Mag, Products, domain=NonNegativeIntegers)
```

Here we have created variable $q_{(d,w,u)}$ inside the model "model". In the `Var()` function we have to indicate the index sets (here Days, Mag, Products) that are used to index the variable. Index sets tells to the program that the variable depends on specific sets.

The "domain" directive indicates to the model the type of the variable. The variable in the example above belongs to non-negative integers. Other values used for this directive are `Binary`, for binary variables, and `NonNegativeReals`, for continuous variables.

CONSTRAINTS

Constraints are usually specified using equality or inequality expressions that are created using a rule, which is a Python function:

```
@model.Constraint(Days, Weeks)
def quattro(m, d, w):
    return model.otw[d,w] >= sum([model.pt[d,w,u] for u in Products]) - HW
```

The first line of code tells the program that we are creating a constraint for "model", which needs to initialize elements from sets Days and Weeks. The rule (or Python function) is defined in rows 2 and 3. In this example, the inequality expression satisfies Constraint 5. The function will call the inequality for each $d \in Days$ and for each $w \in Weeks$.

Row 1 calls the sets from which to take the indexes of the variable. In row 3 the function iterates over all elements of the two sets and uses them to index the variables. There is however, an exception when using the `sum()` function. In this case indexes *w* and *d* are fixed, and the sum is done across al products *u*. We do this because we want to sum values of all parameters with same day-week couple but different products.

OBJECTIVES

The process is quite similar to the one used for constraints:

```
@model.Objective(sense=minimize)
def obj(m):
 return sum([model.pt[d,w,u]for d in Days for w in Weeks for u in Products])
```

The function calls an objective for the model. With "sense" the functions selects the type of optimization. In the example we want to minimize the objective function, therefore we selected the minimization option. Differently from constraint `.Objective()` function takes only a function as input, and not an equality or inequality.

3.2.3 Getting results

To run the optimization program it is necessary to indicate the solver and the model that has to be solved. This can be done with the following string of code:

```
results = SolverFactory('cbc').solve(model)
```

Function `SolverFactory()` tells to the program which solver to use and `solve()` function "feeds" a the model to it. In this example, we are telling the program to use the CBC solver to find the solution of the model "model". The output of the solver will be stored in an element called "results".

After running the command, it is possible to show the output of the solver:

```
results.write()
```

The output is shown in Figure 9. The first information displayed is the status of the solver. It indicates if the process has been aborted, if there is a solution or if the problem is infeasible. In Figure 9 the status is "aborted", but the model managed to find a solution anyway. Probably the solver has been stopped before finding the best possible solution.

The output contains three additional pieces of information:

1. Problem information. Here there is a list containing general information about the problem. It contains bounds of the problem, number of elements and type of problem

2. Solver information. It contains basic information about the solver. If the solver has been stopped, the reason is indicated here.

3. Solution information. Here there are some indications about the solution found.

```
WARNING:pyomo.core:Loading a SolverResults object with an 'aborted' status, but containing a solution
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Name: unknown
  Lower bound: 271879.87
  Upper bound: 283642.05029762
  Number of objectives: 1
  Number of constraints: 3544
  Number of variables: 5160
  Number of binary variables: 2400
  Number of integer variables: 5445
  Number of nonzeros: 840
  Sense: minimize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: aborted
  User time: -1.0
  System time: 1191.8
  Wallclock time: 1200.38
  Termination condition: maxTimeLimit
  Termination message: Optimization terminated because the time expended exceeded the value specified in the seconds parameter.
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 31636
      Number of created subproblems: 31636
    Black box:
      Number of iterations: 2171771
  Error rc: 0
  Time: 1200.4763922691345
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

Figure 9 - example of a solver output

The output in Figure 9 immediately gives us general information about the optimization problem. However, the final users want to see the values that the solver assigned to the variables of the problem. The Pyomo function to show solved variables is `.display()`:

```
# Display values found for variable q
model.q.display()
```

The output is displayed in Figure 10. It shows all the main information about the variable: size, bounds, domain, keys, and values. We are mostly interest in the association between columns Key and Value. Their connection allow us to understand which value the variable takes for a specific combination of elements of parameters. Therefore, if we look at Figure 10, we can say that on day "1" of week 45, the warehouse contains 415 bottles of product "28". 415 is the best value that the model found during the optimization process.

```
M : Size=405, Index=M_index
    Key              : Lower : Value : Upper : Fixed : Stale : Domain
    ('1', '44', '28') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', '56') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', '7A') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'AP') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'BA') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'BI') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'BT') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'CR') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'CV') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'PE') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'PR') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'RP') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'SA') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'VB') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '44', 'ZZ') :     0 :   0.0 :  None : False : False : NonNegativeIntegers
    ('1', '45', '28') :     0 : 415.0 :  None : False : False : NonNegativeIntegers
```

Figure 10 - Example of variable output

Unfortunately, since it is not in a tabular structure, it is very difficult to work with this type of output. The solution that we have found is to convert it into a Pandas DataFrame. Not only DataFrames are easier to work with, but also it will be easier to export the results in a .xlsx file. The first step is to put the needed information inside a Python list. The code used has this structure:

```
warehouse = list((d, w, u, model.M[d,w,u]()) for d in Days for w in Mag for
u in Products)
```

This string of code creates a list of tuples and calls it "warehouse". It uses a nested for-loop to iterate over all elements inside the three sets: Days, Mag and Products. The tuple contains a single element of each set plus the solution of the variable M for those elements.

To transform the list in a DataFrame we will make use of the Pandas function `.DataFrame()`:

```
pd.DataFrame(warehouse, columns = ['Day','Week', 'Product', 'M'])
```

This operation will be done for all variables of the problems. The resulting DataFrames will be merged together based on week number and product. The final result will be similar to the one shown in Table 1.

Week	Product	Demand	Production (bottles)	Quantity stored (bottles	Production (liters)	Quantity stored (liters)	Production time (hours)
43	28	37	324.0	287.0	100.0	57.0	3.48
44	28	0	120.0	407.0	123.0	90.0	1.78
45	28	31	0.0	376.0	0.0	90.0	0.00
46	28	21	0.0	355.0	0.0	90.0	0.00
47	28	2	120.0	473.0	0.0	0.0	1.78

Table 1 - Example of DataFrame containing the results

As requested by the owners, the program should output three different tables containing different pieces of information:

1. Schedule by week containing quantities produced and stored of product and bottles of product.
2. Schedule by Day/Week containing quantities of bottles produced.
3. Quantities of product stored in each tank by day/week.

These tables will be automatically saved inside an Excel file, which will be easier to read and share. We will also create an additional table to help us during the sensitivity analysis in Capter 5. The first column will contain the overtime hours worked each day of the week. The second column will contain the weekly costs of warehouse. The last column will contain the number of hours worked each day of the week. The values of overtime and production time are summed across all days of the week. Table 2 shows an example of the table. Its purpose is to provide additional information about the results and therefore help during the sensitivity analysis.

Week	overtime	warehouse	production time
43	3.8	145.3	27.88
44	2.3	261.8	25.06
45	0.0	301.0	10.33
46	0.0	277.0	11.88
47	0.0	273.0	8.69
48	1.8	261.0	16.90
49	0.0	242.4	6.57

Table 2 - Example of results table

3.3 Branch and Cut algorithms

As mentioned in paragraph 3.1, the solver used by the optimization program uses the branch and cut algorithm to find solutions to the problem. In this paragraph, we want to give a brief explanation of how this algorithm works.

Branch-and-cut algorithms are widely used to solve mixed integer linear programming problems (MILPs) [13]. These algorithms use a combination of *branch-and-bound algorithms* and *cutting plane methods* to solve a sequence of Linear Programming Relaxations of the problem [13]. Relaxations are modelling strategies that approximate optimization problems by removing integrality restrictions from their variables [14]. Therefore, they transform an NP-hard optimization problem into a problem that can be solved in polynomial time [15].

3.3.1 Linear Programming Relaxations

Imagine a two-dimension integer-programming problem as the one shown in Figure 11. All the points in the graph represent a feasible integer solution. All lines represent the constraints of the problem, which encode the solution space. The dotted lines represent the integrality restrictions of the problem. They limit the range of values of a variable to integer numbers, hence the problem considered is NP-hard [21]. Linear Programming Relaxations ignore the integrality restrictions, and replace them with linear constraints represented by the solid lines [13].

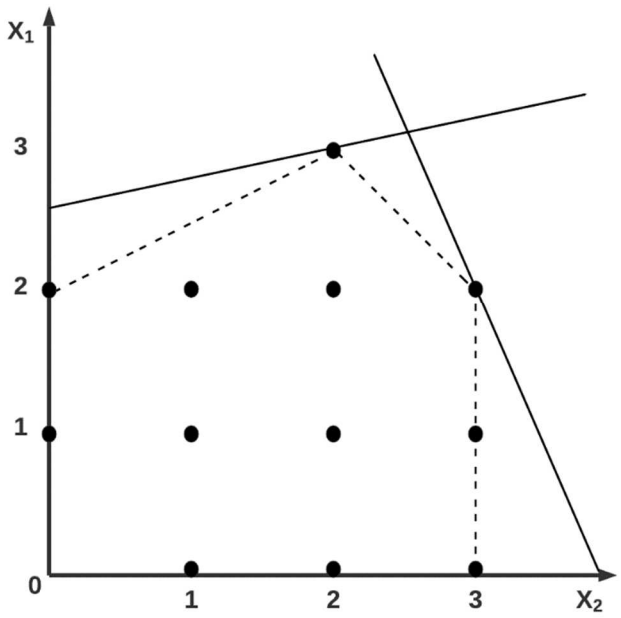

Figure 11 - Example of a two dimensions integer-programming problem

3.3.2 Cutting Plane Method

The main goal of the cutting plane method is to improve the relaxation in order to approximate the integer-programming problem more closely [13]. The method works by taking a small subset of the constraints of the original problem and computing an optimal solution for those constraints. If some of the remaining constraints are violated, they are added to the current Linear Programming Problem. This process literally "cuts off" all the solutions that are infeasible to the original problem. If no constraints are violated, then the optimum of the subproblem solves also the main problem [16].
In order to find the solution to the problem, cutting plane algorithms rely on valid inequalities. An inequality is defined as "valid" for a specific set if all points inside the set satisfy the inequality [18]. These inequalities can be used by the algorithm as cutting planes. If the new solutions violate valid inequalities of the main problem, those inequalities are added to the subproblem. Solutions found during the course of the algorithm are feasible only if no valid inequality of the original problem is violated [16].

3.3.3 Branch-and-bound algorithms

Branch-and-bound algorithms allow for an enumeration of all the possible solution for a problem. They work by subdividing the problem P^0 into subproblems $P^1, P^2, \ldots\ldots, P^{n_0}$, such that their aggregation is the full problem P^0. Therefore, each possible solution for P^0, must be a solution for at least one of the sub problems. This so called divide-and-conquer method is schematized in Figure 13.

For each subproblem the algorithm calculates upper and lower bounds. Upper bounds can be found through relaxation of the problem. As Jünger et al. (1993) suggest: "*a solution of the relaxed problem gives an upper bound on the optimum objective function value of the problem it was derived from*".

Local upper bounds (lub) are all those upper bounds, which are specific of a subproblem. **Global upper bounds** (glb) are upper bounds valid for the original problem.

After the definition of the lub of a sub problem there can be three different situations:

1. lub > glb and the solution of the sub problem is feasible. The glb will be updated and the sub problem will be considered solved.
2. lub < glb. The sub problem will be fathomed because its solution will never be better than the best solution found so far.
3. lub > glb and the solution of the sub problem is not feasible. In this case the algorithm will perform a branching step as shown in Figure 12.

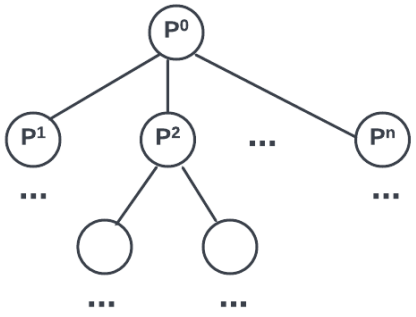

Figure 12 – Branch and Bound tree

When the algorithm runs out of subproblems to analyse, the memorized feasible solution whose objective function value is equal to the global upper bound can be output as the optimum solution [16]. The full process is schematized in Figure 13.

The main advantage of using this specific method is that it creates smaller and so easier to solve problems, since they have less possible solutions [17].

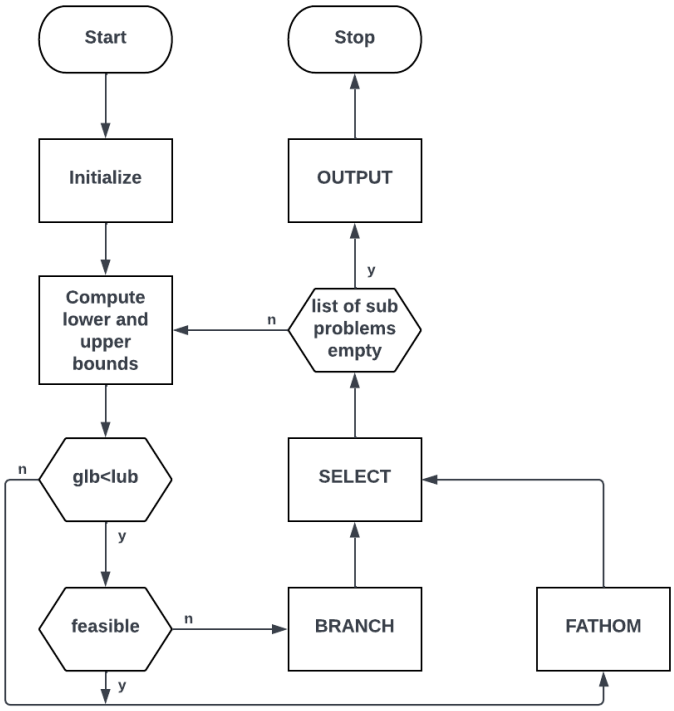

Figure 13 - Branch and Bound algorithm

Chapter 4

4 Analysis

In this chapter, we present the results found by the optimization program and perform a sensitivity analysis. Sensitivity analysis is a tool used to assess how changes in the input variables of a system will affect the output. In this case, the sensitivity analysis will provide insight into the behaviour of the optimization program in different situations. We will then be able to examine how changes in specific inputs influence the optimal solution.

Sensitivity analysis will also allow us to evaluate the model's reliability and spot areas for improvement. For this reason, it is crucial to share and discuss the results of every analysis with the owners of the company. They are the entities that decide if the results are realistic and meet their expectations. After each run, we will update the model based on the feedback received. This process will continue until we can find a satisfactory solution.

4.1 Procedure

To perform the analysis and discuss the results we will follow the procedure shown in Figure 14.

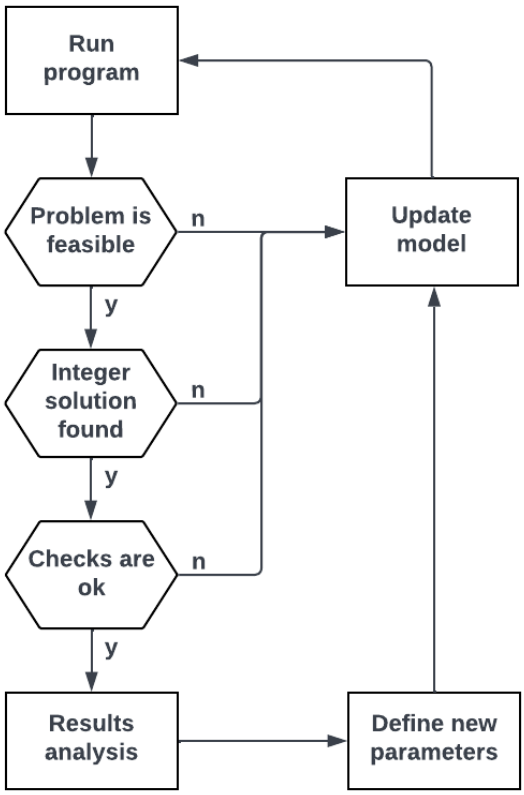

Figure 14 - Scheme of the procedure

After running the solver, the program will perform three different checks. Their function is to indicate that the model performs properly and no constraints are violated. The checks are:

1. The problem is feasible.

2. The solver found an integer solution.

3. All constraints have been satisfied.

The first two checks are done automatically by the solver and returned inside its output as in Figure 9. If the solver was not able to find a solution, the output will display one of these two errors:

1. The problem is infeasible.

2. The solver had not found any solution yet.

The first error indicates that, given specific parameters and constraints, it is impossible to solve the model. The second error raises when solving integer-programming problems. If some variables are required to be integers, the solver must return an integer solution. Therefore, if the solver returns a non-integer solution, probably some constraints are not satisfied.
To check if all constraints have been satisfied, we need to format the results in a DataFrame as in Table 1. We will then use Pandas functions to aggregate and manipulate data to calculate all the following checks:

1. Control if there are negative values in the solutions. All variables in the model must be positive real numbers ≥ 0.

2. Check if total production of a product $\geq$ bottles produced $\geq$ demand. If this check fails, it means that in some periods demand has not been covered or we have bottled more product than the one prepared.

3. Check if Constraints 4, 7, 10, 11, and 16 have been satisfied.

The program will print the results of all the checks as shown in Figure 15. In the example, all checks are “True” so the model should be reliable. This check report allows us to see immediately if there are problems with the model. However, manual checks will be still necessary to confirm the reliability of the model.

If all checks are positive, the results can be shared with the owners and then analyzed. The main goal of this process is to understand if the results meet the owners’ necessities and expectations. This is a good opportunity to understand if there are constraints that need to be added or removed from the model. Based on the solutions found, the owners could also choose to modify the objective function or revise some of the parameters. After each update of the model, the model will run again to find the new solutions. This process will continue until we will be able to find one or more configurations that satisfy the owners’ necessities.

```
**Check report**

No negative values:  True
Check if production >= bottles produced >= demand:
  PE : True
  BA : True
  BI : True
  7A : True
  PR : True
  CR : True
  CV : True
  SA : True
  BT : True
  VB : True
  RP : True
  28 : True
  AP : True
  56 : True
  ZZ : True


Overtime hours <= 2:  True
Constraint 7 respected:  True
Constraint 10 respected:  True
Constraint 11 respected:  True
Constraint 16 respected:  True
```

Figure 15 - Checks on constraints

4.2 First analysis: Initial model

The first analysis deals with the results given by the optimization of the initial model (paragraph 3.4). The solver will consider the timeframe that goes from week 44, the second week of November, to week 49. In this way, it will possible to test if the model performs well also in periods of high demand.

The first few runs however already revealed a big problem that needed to be solved. The runtime of the solver was extremely long. In fact, even after more than five hours, the solver kept running without returning a solution. The possible reasons behind this could be two: some constraints are not clear or the model is too big and the optimum solutions are very similar to each other. In both cases, the solver could go in error because unable to find the single best solution possible. To solve this problem, we decided to use a heuristic to find a solution. Instead of waiting for the best solution possible, we will accept the best solution that the solver is able to find in a given amount of time. If the solution is integer, we will consider it as the best feasible solution. If the solution is not integer, we will have to check and update the constraints.

For the following analysis, we asked the model to return the best solution found in 10 minutes. The solver was able to return an integer solution of the first model. The output in Figure 18 shows that the model is quite big with around 12.000 elements.

```
#   Problem Information
# ------------------------------------------------------------
Problem:
- Name: unknown
  Lower bound: 256225.38
  Upper bound: 262191.30349206
  Number of objectives: 1
  Number of constraints: 2118
  Number of variables: 3774
  Number of binary variables: 1800
  Number of integer variables: 4155
  Number of nonzeros: 810
  Sense: minimize
```

Figure 16 - information about first model

All checks are positive therefore, the solution found should be feasible. However, to understand if the results make sense and meet owners' necessities, it is better to analyse them manually. Table 3 shows an example of the table reporting overtime hours, warehouse costs, and total hours worked. The table has been created as explained at the end of paragraph 3.2.3.

Week	overtime	warehouse	production time
44	6.0	77.1	28.42
45	0.0	132.0	12.64
46	2.0	144.9	18.84
47	0.0	92.8	0.00
48	0.0	35.2	2.43
49	0.0	1.3	0.00

Table 3 - Tables with results.

All three variables follow specific trends, as can be seen in Figure 17. Production and overtime hours are more focused during the first three weeks. Warehouse costs and quantities instead have an initial increase followed by a sudden decrease.

A possible explanation for these trends is that during the first weeks, it is necessary to keep up with the demand and to build some warehouses for the following high-demand weeks. The solver is probably trying to concentrate production during the initial weeks to produce bigger batches of products. Few big batches of product will generate fewer setup times than many small batches. Therefore, when a product is being bottled during the first weeks, the model tries to schedule big batches to lower the total time of production. The effects of this decision can be seen in the plot in figure 17. Given demand with a constant trend, both production and warehouse soar during the first two weeks and decrease after.

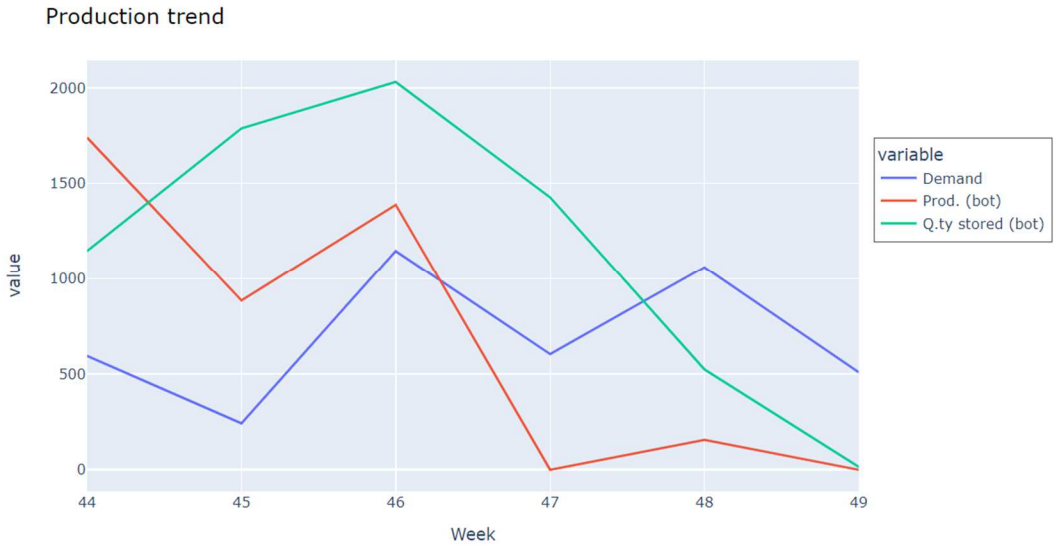

Figure 17 - Trend of production and demand

We can finally examine the schedule produced by the program. All the checks confirm that the solution is coherent and respects the constraints. However, upon looking at the results in Table 4, the owners noticed a problem with the schedule. In fact, in some cases, the batches of bottles produced (column "Prod. (bot)") are excessively small.

Week	Product	Demand	Prod. (bot)	Q.ty stored (bot)	Prod. (lt)	Q.ty stored (lt)	Prod. time (h)	bp
44	BA	3	4.0	1.0	397.0	106.0	1.17	1.0
45	BA	22	24.0	3.0	4.0	485.0	1.33	1.0
46	BA	600	652.0	55.0	613.0	0.0	6.56	1.0
47	BA	39	0.0	16.0	0.0	613.0	0.00	0.0
48	BA	91	156.0	81.0	0.0	496.0	2.43	1.0
49	BA	78	0.0	3.0	0.0	496.0	0.00	0.0

Table 4 - Example of schedule

Owners find it inconvenient to set up the production line only to produce a few bottles of product. For example, rather than setting up the production line for four bottles, it could be less time-consuming to prepare them manually. In addition, after the production of every batch of product, it is necessary to compile some documents to keep track of the bottles produced. This operation only takes a couple of minutes, but having many different documents, for many small batches of the same product can create confusion.

This same problem might occur also with the quantities of raw products produced. Recipes for each product require specific doses of each ingredient. Producing too little of a certain product could make the dosing of those ingredients very difficult, if not impossible. Moreover, dosing errors on a lower scale are more difficult to solve.

To avoid these issues, the owner requested to add a lower limit on the dimensions and number of batch of products. For this reason, we will implement three new constraints into the model:

Constraint 17:

$$q_{(d,w,u)} \geq bp_{(d,w,u)} * 120 \quad \forall\, u \in Products, \forall\, w \in Weeks, \forall\, d \in Days$$

Specifying the minimum production quantity for a batch. The owners find it inconvenient to set up the production lines for small batches of product.

Constraint 18:

$$Qq_{(w,u)} \geq Qp_{(w,u)} * 100 \quad \forall\, u \in Products, \forall\, w \in Weeks$$

The owners asserted that it is not worth to produce less than 100 liter of product at once.

Constraint 19:

$$\sum_{d \in Days} bp_{(d,w,u)} \leq 1 \quad \forall\, u \in Products, \forall\, w \in Weeks$$

We want to concentrate production of a product in a single day of the week.

4.3 Second analysis: Constraints on quantities produced

The results of the new model, as in Table 5, are quite similar to the ones of the previous one, as in Table 3. The only exception happens during the first week of the schedule. In this case, all variables are lower than the ones from the previous model. This is due to the fact that the initial small batches of product processed during the first week, are now combined into big batches processed in the following weeks.

Week	overtime	warehouse	production time
44	4.0	56.2	24.44
45	0.0	103.8	13.99
46	2.0	144.2	19.22
47	0.0	92.1	0.00
48	0.0	35.5	2.56
49	0.0	1.6	0.00

Table 5 - Results from the second model

The reduction in the number of batches causes a reduction in the overall setup time, as shown in Table 6. This reduction balances with the increased processing time of all bottles. This is the main reason why the results of the timings in Tables 5 and 3 are similar to each other. Obviously, if the minimum batch size was bigger, this similarity would be weaker.

Week	N.Batches 1	Setup 1 (h)	N.Batches 2	Setup 2 (h)
44	10	9.0	9	8.2
45	4	4.0	3	2.8
46	5	4.6	5	4.7
47	0	0.0	0	0.0
48	1	1.0	1	1.0
49	0	0.0	0	0.0

Table 6 - Set-up times and batches comparison between the two models

The plot of production, warehouse, and demand quantities in Figure 18 confirms that the results of this model are very close to the ones found with the previous model. The only noticeable difference is that production follows the demand more closely during the first three weeks. This causes slightly smaller warehouse quantities during the first two weeks of schedule.

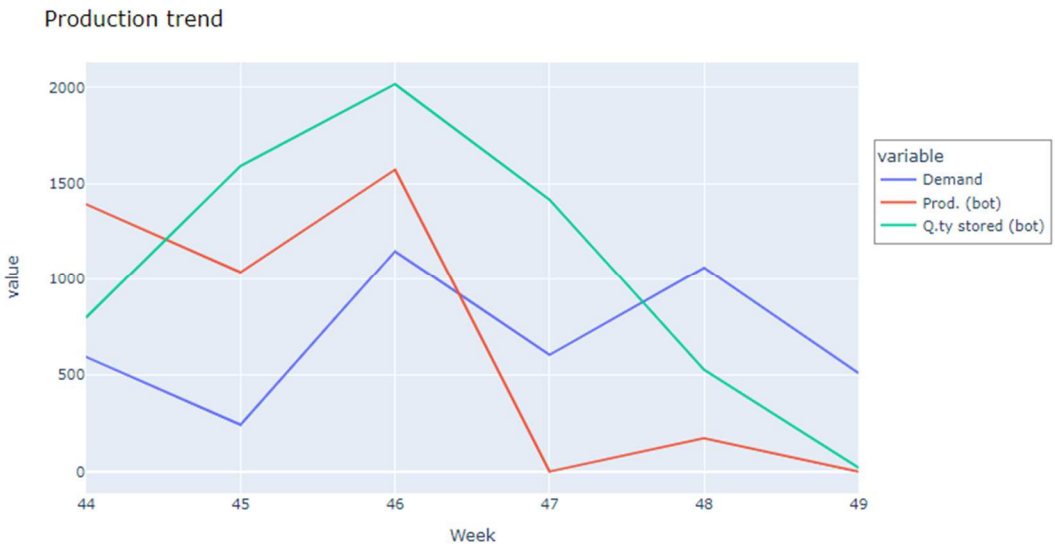

Figure 18 - Trend of production and demand

4.3.1 Test on periods of low demand

After we have made sure that the program is reliable during periods of high demand, we want to see how it behaves during periods of low demand. During these periods, the scheduling process is quite easy and the owners rarely encounter significant problems. However, we are interested to see if our model is able to suggest solutions unseen by the owners.

For the analysis, we decided to consider the period that goes from week 3 to week 8. In these four weeks, the demand reaches the lowest level of the year. We will also make use of all the constraints. The results of the first run of the model are shown in Table 7.

Week	overtime	warehouse	production time
3	0.0	77.1	14.62
4	0.0	71.5	7.62
5	0.0	38.2	4.46
6	0.0	41.7	1.78
7	0.0	39.4	1.55
8	0.0	41.6	1.71

Table 7 - Results with low demand

As expected, the solution does not provide for overtime hours and the scheduled production time is lower than the one in the previous examples. During the last three weeks, total production is less than two hours per week. Upon seeing these results, the owners were interested in knowing if it is possible to focus production only on some weeks. This will allow them to have weeks with production time close to zero. Their idea is that during those weeks they can focus on activities that do not require their presence in the plant.

If we think about this request from the model point of view, the owners are basically asking to minimize the total number of batches produced. They want to produce fewer, but bigger batches of product. In order to fulfil this request we will have to update the objective function of the model.

Every time a batch is produced, binary variable $\boldsymbol{bp}_{(\boldsymbol{d},\boldsymbol{w},\boldsymbol{u})}$ is equal to 1. Therefore, to minimize the total batches is necessary to minimize the sum of this variable. The new objective function will be:

$$\min(w_1 * \sum_{d\in Days,w\in Weeks,u\in Products} \boldsymbol{pt}_{(\boldsymbol{d},\boldsymbol{w},\boldsymbol{u})} + w_2 * \sum_{u\in Products,w\in Weeks} cw_{(u)} * M_{(w,u)} + w_3 * \sum_{u\in Products,w\in Weeks,d\in Days} \boldsymbol{bp}_{(\boldsymbol{d},\boldsymbol{w},\boldsymbol{u})})$$

Total production time will still have the higher weight (0.6) in the function, while warehouse costs and binary production will have lower and equal weights (0.2).

The results processed by the new model are shown in Table 8.

Week	overtime	warehouse	production time
3	2.0	94.2	19.23
4	0.0	79.3	3.72
5	0.0	54.3	7.79
6	0.0	53.0	0.00
7	0.0	47.1	0.00
8	0.0	49.3	1.71

Table 8 - Results with new objective function.

The new solutions successfully meets our expectations. The new schedule provides for two weeks without production. This solution comes at the cost of higher production time, warehouse costs and overtime hours. For this reason, the owners are not convinced by the effectiveness of this new model. Therefore, for periods of normal administration, they prefer to keep the original objective function:

$$\min(w_1 * \sum_{d\in Days,w\in Weeks,u\in Products} \boldsymbol{pt}_{(\boldsymbol{d},\boldsymbol{w},\boldsymbol{u})} + w_2 * \sum_{u\in Products,w\in Weeks} cw_{(u)} * M_{(w,u)}$$

They think however that the new objective functions could be still useful in particular situations. If for example new clients are visiting, or the machines need maintenance, it is better to have the entire week free from production.

4.4 Third analysis: Emergencies

The program has proven to be effective in finding an optimal schedule in periods of both high and low demand. However, we want to test if it is useful also in managing emergencies. To do this we will analyze a real-life scenario that happened a couple of years ago. Figure 25 shows the demand from week 22 to week 28 of 2021.

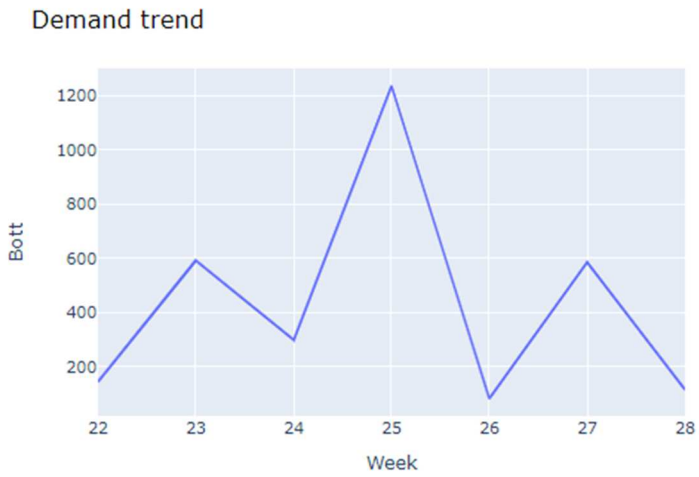

Figure 19 - Demand anomaly

The sudden increase in demand of week 25 was due to a big and unexpected order for a specific product. The problem was that when the order arrived at the beginning of week 22, the business did not have enough stock of raw materials. The supply takes about 2 weeks, therefore the product was ready to be bottle only at the beginning of week 25. If the owners had opted to use our program to schedule the production of the product from week 25, they would have found out that the problem is infeasible as shown in the "Termination message" Figure 20.

```
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: warning
  User time: -1.0
  System time: 0.14
  Wallclock time: 0.21
  Termination condition: infeasible
  Termination message: Model was proven to be infeasible.
  Statistics:
    Branch and bound:
      Number of bounded subproblems: None
      Number of created subproblems: None
  Error rc: 0
  Time: 0.2925679683685303
```

Figure 20 - Output of an infeasible problem

The order was for about 1100 bottles and the maximum daily production of the product is around 650 bottles. To fulfill the demand the owners dedicated almost two entire days to the production of the finished product. This schedule does not comply with Constraint 19 of the model. If we drop this constraint from the model, the problem becomes feasible and the program can find a solution. The new solution found corresponds exactly to the one actually applied by the owners. Therefore, the model confirms that the manual solution was indeed an optimal solution.
The one just presented is just an example, but in reality, many other different similar issues could occur during the everyday life of the business.

4.5 Fourth analysis: New investment

One of the main reasons for the creation of the optimization program was to have a tool that allowed testing how scheduling changes when some parameters of the model change. In particular, the objective was to help the owners to quantify the effects of new machinery on production.
During the last couple of years, the owners have begun thinking about replacing the bottling machine. This machine is quite old and it is starting to break apart. However, investing in a new machine is expensive and the owners are not sure if it is really worth it.

The goal of the following analysis is to use our optimization program as a tool to test the effect of a new bottling machine on production. To do this, we will have to update the parameters of the optimization model based on the specifications of the new machine. The parameters in question are:

1. Waiting time ($wt_{(1,u)}$): The new machine is semi-automated. The only operation that the operator has to do is to load the machine with empty bottles. Then, he will be free to work on other tasks. For this reason, we will consider the waiting time for a bottle equal to 0.

2. Bottle processing time ($bt_{(1,u)}$): In this case, the bottle processing time will lose its original meaning. For the new machine, *bt* is equal to the time necessary to load the machine with bottles. Therefore the total processing time on the machine (7), is equal to the time that the operator spends working on the machine.

$$\left(wt_{(i,u)} + \frac{bt_{(i,u)}}{B_{(i,u)}}\right) * q_{(d,w,u)} \tag{7}$$

3. Buffer capacity ($B_{(i,u)}$): The new machine can be loaded with at most 100 bottles at a time.

4. Setup time ($st_{(i,u)}$): Setup times are slightly lower than in the old machine.

After updating all parameters, we told the program to solve the optimization problem starting from week 44. The results found, are not the ones we were expecting to see. As shown in figure 9, in the new solution the overtime hours are maxed out and warehouse levels are always at zero. This means that, with the new machine, production alone can always fulfill the demand.

Week	overtime	warehouse	production time
44	6.0	0.0	4.28
45	6.0	0.0	3.00
46	6.0	0.0	9.27
47	6.0	0.0	5.61
48	6.0	0.0	10.40
49	6.0	0.0	5.05

Table 9 - Results with new machine

The solution, however, is not optimal for the owners. Having zero stock and leaving no extra time for production could be dangerous. If for example, there are some unexpected orders, the business

will neither have time nor warehouse levels available to fulfil them. For this reason, it is necessary to update the model again. Since we want to lower the overtime hours worked, we need to modify the constraint:

$$0 \leq otw_{d,w} \leq 2$$

For this test, we assumed that production can be carried on only during the daily eight working hours. The new results in Table 10 show a clear improvement compared to the ones from the old model in Table 5. Overtime hours have been zeroed out, and the total production time is lower. These results are promising, but are not enough to take a final decision. The business will first

Week	overtime	warehouse	production time
44	0.0	54.2	20.57
45	0.0	135.9	16.89
46	0.0	140.1	10.44
47	0.0	101.6	3.78
48	0.0	34.7	0.00
49	0.0	5.6	1.64

Table 10 - Results with new machine and constraint

4.6 Considerations

The model has proven to be reliable and very versatile. During the analysis, we were able to update it quickly to meet the owners' expectations and necessities. The results found seem reasonable and sparked some great points of discussion. In particular, the owners are pleased with the results found during the second analysis. They believe that having a clear schedule during difficult periods can make production less chaotic and help them to reduce stress levels. This will also enable them to be more efficient and quick in solving unexpected problems.
The owners also see a lot of potential in the fourth analysis and are thinking about expanding it to other processes in the company in the future.

However, we are uncertain about the effectiveness of the optimization program in everyday life. The schedules produced can be helpful during periods of high demand. However, in periods of routine, these schedules probably will not provide added value to the business. In these periods, the solutions proposed are proven similar to the choices the owners have already made. In addition, the owners do not have either the desire or the necessary knowledge to update the program every time it is necessary. Therefore, the use of the program within the business could become very problematic in the absence of continuous assistance.

Chapter 5

5 Business implementation

As we have seen in previous chapters, implementing software within a company is not a simple task. Based on the experience gained so far, we want to evaluate the main activities necessary to implement the scheduling software within the business. We will analyse the difficulties and opportunities that such implementation could bring. We will also evaluate what changes are necessary within the business to ensure that the implementation is efficient.

5.1 Software development

In this paragraph, we want to provide an overview of the costs and opportunities associated with the implementation of a production optimization software within the business. We will also indicate the type of information necessary for the proper functioning of the software.

5.1.1 Information needed

The information that we consider necessary for the implementation of the software are:

1. Production data.
2. Warehouse data.
3. Demand data.
4. Relationship between production and other business areas.

By production data, we mean all data related to process times and production line capacities. It is important to know precisely how much time the production line takes to process a certain number of bottles. Therefore, we need information regarding setup times, machine cleaning times, bottle processing times, buffer sizes, etc. All of this data is essential for the proper functioning of the software and must be as reliable as possible.

Warehouse data concerns three different areas:

• Warehouse of raw materials.
• Warehouse of bottled products.
• Warehouse of finished products.

The main information we need is the maximum capacity of each one of the three warehouses, the costs they generate, and their actual quantities. While the first two are fixed and are only updated in special cases, the actual quantities must be constantly monitored. This data in fact, varies with each production or sale.

For the sake of our analysis, production and warehouse data were manually collected. In fact, there were not much information within the business. While the manually collected data is close to reality, it cannot be considered reliable. In the event that the software is implemented within the company,

we believe that it is essential for this data to be correct and promptly updated. Therefore it should be necessary to have a solid and reliable data collection and management system, which is currently not present within the business.

Demand data is essential to understand how to schedule production because it indicates which are the production volumes hat have to be met. As of now, the company's forecast is based on past orders, and there is no willingness on the owners' part to change this system. When working with demand, the information we are interested in is the quantities of products that need to be sold each week. These quantities are the reference point for understanding how much bottles have to be produced.

The last important piece of information for proper implementation of our software, is to understand how various business areas interact with the production process. We want to understand if certain production decisions could be incompatible with the constraints of other areas. For example, it is not guaranteed that producing large quantities of a product is compatible with the capacity to supply raw materials for that specific product.

5.1.2 Opportunities costs

In our opinion the implementation of a scheduling software within the company would require a significant investment both in time and money.

During our project, the mapping of the production, data collection process, and model development process have engaged us for approximately four weeks. In particular, the most demanding phases were data collection and model development. In case the work done is expanded, we expect the project to require around three months of commitment. The first three weeks would certainly be dedicated to mapping all the company processes and their relationships. The following four to six weeks would then be dedicated to model development and its implementation within the optimization software. Since we already have a solid foundation to start from, we expect this process to have few difficulties. The final three to four weeks would be dedicated to testing the software and model. This phase would allow us to test the robustness of the work done and, if necessary, give us the opportunity to make significant changes to the model.

However, the model created is useful only if supported by quality data and up-do-date data. Unfortunately, such data is not present within the company. It is therefore necessary for the owners to invest in a good data management and measurement system. This system would help to gather and centralize data in a single system. Having data centralized in a single database would simplify the activity of entering it into the optimization software. However, the developing and implementation of such a similar system would require the support of an external company specialized in this field. We in fact, do not have neither the knowledge nor the tools necessary to perform this operation.

Having a well-built optimization program, implemented with up-to-date data could be a great opportunity for the business. In particular, we believe that there could be improvements in four main areas:

1. **Resource management**: The optimal solutions found by the software would allow for efficient use of time and inventory. Also, being able to know when and how much to produce, will help with the managing of the supply chain.

2. **Visibility on the production process**: proper data management would provide the ability to quickly identify problems and bottlenecks in the production process. Data gathered could

show which phases of the production process take more time and were there are more interruptions. It is possible also to see if the production line is underperforming or if there are inefficiencies.

3. **Organization inside the business**: a clear and well defined system could help with tracking all the activities inside the business. This will allow for a database of historic data about production batches. In addition, all data will be stored in a single system and so it will be easier and faster to retrieve it.

4. **Investment evaluation**: A good scheduling software could be useful for evaluating the opportunity cost of new investments in the production line. As shown in Chapter 4.5, the software would allow us to understand the impact of a new machine on production without having to incur the cost of the investment.

5.2 Model strengths and weaknesses

In the following section, we want to analyse the strengths and weaknesses of the model we have created. In case we expand the work, we want to know what worked correctly and what should be reassessed or modified. We also want to understand if and how to further develop the model if we want to make it as close as reality as possible.

5.2.1 Weaknesses

The first major problem with our model is that it does not take into account the actual quantities of inventory and stored product. For simplicity, we assumed that these two parameters were equal for all products. Obviously, this assumption is incorrect in reality. Every day these values change and the model parameters should be updated accordingly. In the program that we have created, this update must be done manually. This process is time-consuming and can generate errors. The same problem also happens when considering the quantities of raw materials. In our analysis, we decided to consider them as unlimited, but we are well aware that is not true in reality. Improper management of the inventory of raw materials could lead to unexpected stops in production. Knowing the level of resource availability is essential to create a feasible schedule.

Another important consideration is that the model proposed does not consider line stops, maintenance, or other various issues. This means that if one of these problems occurs, the entire schedule would become infeasible. The literature proposes some heuristic solutions to this problem [22], such as increasing the values of setup times. In the future, it may be interesting to test some of these solutions to verify the robustness of the model.

During the creation of the model, we also found significant difficulties in defining the objective function. It was very difficult for the owners to understand which variables should be minimized. Business priorities can change quickly based on various external and internal factors. For example, in a period of high demand, the owners prefer to limit overtime hours worked. In a period of low demand, the owners prefer to minimize inventory. It is therefore necessary to offer a model capable of performing effectively with variable objective functions.

5.2.2 Strengths

During the sensitivity analysis done in Chapter 4, we demonstrated that the model is capable of adapting effectively to different situations. In particular, updating and adding constraints or objective functions is a relatively simple and fast operation. This gives us a lot of flexibility and partly compensates for our need to update the model frequently. However, updating the model requires specific knowledge that the owners do not have. This means that every time they need to update a constraint, they would need to seek help from third parties.

Another great strength of the model is that it has consistently shown its ability to produce optimal solutions. In fact, the owners have confirmed several times that the solutions found are the same or even better than those they would have adopted themselves. Even when changing parameters, constraints, and objective functions, the model continues to return consistent and feasible results. For the owners, it is also very useful to have the possibility to visualize the results of all the variables of the model and not just the scheduling. We have seen during the analyses in Chapter 4 that the values of worked hours, overtime hours, and inventory levels are as important as the values of the scheduled production.

5.2.3 Future versions of the model

In light of the considerations mentioned above, we have defined some fundamental points that we would like to keep in mind during the development of new versions of the model created:

1. Maintain the structure of the original model: the model has proven to be effective and capable of returning optimal solutions. For this reason, it might represents a solid foundation for any future version of the model.

2. Make the model easily customizable: we have seen that the ability to modify the model quickly is a great advantage for the owners. This possibility must be absolutely maintained. If possible, it necessary to simplify this process. In this way, the owners would be able to modify the model independently and without having to resort to third-party support. We also think that it would be useful to give the possibility to perform a sensitivity analysis every time the model has been updated. This will allow the owners to safely test the results generated by the new model in a sandbox environment.

3. Maintain visibility on the values of all variables of the model: once the optimal schedule has been found, it is necessary to give visibility on the value that each variable of the model assumes. This will allow the owners to better understand the impact that the schedule has on specific variables. As we have seen, after each sensitivity analysis the first results that the owners looked at were the ones about overtime hours and warehouse levels.

4. Add new constraints related to the quantities of stored raw material, finished product, and raw product: we want to ensure that the schedule respects these quantities. We do not want to exceed the maximum values of their quantities and the schedule has to take into consideration the quantities already stored. To meet this requirement, it would be appropriate to have a data management system implemented with the optimization software. In this way, the data could be updated on a daily basis and automatically loaded inside the software.

5. Include parameters that take into consideration line stops or other possible issues: this will allow the owners to be prepared in case of emergency. It could be useful to also carry out some sensitivity analysis to understand which are the most realistic values for those parameters.

6. Include in the model the possibility to manage reworks of finished products. In some rare cases, it may be necessary to perform reworks on finished products. It could be useful for the model to be able to handle these situations when required. This could be done for example by adding a binary variable, which tells if the product needs a rework.

7. Add constraints and parameters dependent on the number of operators in the production line: So far, we have assumed that there is always a single operator in the production line. However, in particular cases of extreme urgency, it is possible to have two operators in line. In this situation, the production time is almost halved, and scheduling should take this into account. It would be interesting for the model to return two schedules, one calculated with a single operator in line and one calculated with two operators in line. This way, it is possible to evaluate the opportunity cost of removing an operator from other business activities.

8. Add constraints related to other business areas: we want the model to be as comprehensive as possible. For this reason, we believe it is necessary to include constraints that do not strictly depend on production. Other areas of the business may have limitations or issues that affect the production process.

5.3 Implementation

Based on what has been said so far, we can now define a plan for implementing software within the company analysed. In this paragraph, we want to summarize the steps we believe are necessary to complete such an implementation and explain the impact that each step could have on the business.

The first step is to thoroughly map the company's processes. Each process is in fact closely linked to the others and therefore cannot be considered as a separate unit. For simplicity, in our analysis we have limited ourselves to mapping only the company's production process. However, this process is closely connected to other processes such as the procurement of raw materials and inventory management. To know how much to produce, for example, it is necessary to have a precise view of inventory levels and available resources.

This mapping process can help us understand if there are any additional constraints to consider when creating the model. Since the production process is not an entity in itself, limitations or issues from other areas can have significant impacts on production. As an example, if there are monthly limits on the procurement of raw materials, production must necessarily respect these limits.

We do not expect this process to be difficult, but taking into consideration every single relationship and constraints of each process could become very time consuming.

The second operation we absolutely recommend the business to do is the implementation of a data collection and management system. During the analysis, we understood how essential it is to have well-organized and, above all, realistic data. Up until now, we have worked with data gathered, measured and stored manually. The process was slow and full of problems. Production times in

particular were extremely difficult to measure. This type of data may be acceptable for testing our model, but it cannot be considered reliable. The business therefore needs a solid system for measuring production times, inventory quantities, raw material quantities, and demand levels. If these information are not correct, the decisions taken could cause more harm than good.

Data gathered should also be well-structured and organized. Structured data could be uploaded into the optimization software automatically, without the need for human intervention. This could limit errors and improve software update times. Structured data is also easier to analyse and to work with. Owners could use it to do specific analysis or to calculate various kpis as production adherence, stock rotation, line downtime and so on.

The data collection and management system should include the following elements:

1. **Sensors**: they should be used to measure production times and status of the machines. They could give useful information about the performances of the production line. The data they produce would be essential for the proper functioning of the optimization model.

2. **Scanners**: to track the location of raw materials and bottles of finished product. They will allow the business to have inventory levels always updated. This could be useful also when taking stock levels. As of now the process is manual and takes two days. With a good data management system the same process could take only some minutes.

3. **Software**: to manage all data collected. Famous example could be Microsoft SQL Server.

However, the process of collecting and organizing data should not sorely depend on the decision to implement the scheduling program. We believe that implementing a data management system is a fundamental step that the business must take in any case. In our opinion, the opportunities presented by such system could easily outweigh its costs.

Once the data management system is up and running, the following step will be to start to work on the optimization model and software. The procedure should be similar to the one followed in our thesis. First the model has to be defined in words, and then will be translated into mathematical notation. As previously said, in future we will also implement in the model the relationships between the production process and other business processes. During this phase, we also want to evaluate all possible problems that may arise during production: emergencies, extraordinary maintenance, machinery breakdowns, illness, etc. In fact, the model must be able to adapt to different situations as quickly and effectively as possible.

The software used to solve the model should be user-friendly and easy to integrate with the data management system. The model in fact, should be updated constantly and automatically with the data gathered inside the business. In addition, users should be able to update parameters, constraints, and objective functions relatively easily. For simplicity, these elements could be defined in advantage, so that the owners could choose only from a specific set of elements.

It is important that scheduling software is accepted by the owners of the business. They must be willing to learn how to use the software and interpret its results. Since the owners are also the end users, they must be convinced of the effectiveness and reliability of the proposed software. If there is no trust in the adopted solution, operators may take control of the scheduling, nullifying the efforts of the software [3].

5.3.1 Feasibility

In light of the considerations made so far, we wonder whether it is worth it for the company to undertake an investment for a scheduling optimization software. In Chapter 4, we have demonstrated that it is possible to create a model that allows for optimal production management. Therefore, theoretically, the project could be feasible. However, we have seen that during routine periods, the software has not proven to be essential, and the schedules found are very similar to those already implemented by the owners. Therefore, we believe that, even though the software can certainly be useful, its implementation and maintenance costs could outweigh the benefits brought to the company.

In particular, we are not convinced that the size of the company analysed is such as to justify the need to invest in a scheduling software. We have seen that in some cases, the company can even survive for a few weeks without needing production. During these periods, scheduling is not the major issue for the business. There are however, certainly periods of critical demand that we believe can be easily managed by the software we have created in Capters 2 and 3.

From our point of view, a scheduling software would be more suitable for companies where production scheduling is a daily problem and the production volumes are substantial. In this case, the software could be of vital support. The schedules would be more accurate than those found by a human being, and above all, there would be the possibility of reacting more quickly to unforeseen events. Additionally, with higher production volumes, any time or resource savings could be amplified.

Conclusions

The creation of the optimization program proved to be complex and not without difficulties. Contrary to our expectations, building the mathematical model turned out to be the simplest part of the project. Once the system was defined in words, we were able to translate it into a mathematical language without having to resort to excessively complex limits and functions. This is also shown by the simplicity with which we were able to modify the model based on the specific needs of the business. Of course, there was some difficulty even in this phase. We had to reassess some constraints several times, but within a couple of weeks we were able to have a correct and functioning model.

The implementation of the model within the software was equally simple. The only real difficulty was learning how to interact with the Pyomo library. Fortunately, the vast number of online resources helped us speed up this process. The implementation within the software was also useful for testing the correctness of the mathematical model. The first tests, in fact, allowed us to understand if the model was logically correct. In addition, the ability to use multiple Python libraries simultaneously gave us more flexibility during the work.

The program finds solutions consistent with the goals of the analysis. The owners think that the schedules found are practically applicable and in some rare cases better than those found manually. We have also demonstrated how the program can be adapted to different requirements without too much difficulty. The overall judgment on the performance of the optimization program is positive, and in the future, there is the idea of expanding and solidifying the work done. There is a lot of interest, especially in the possibility of using the model to evaluate the effect of changes within the production line.

However, at the moment we want to avoid taking the longest step. During the work, in fact, we encountered several problems that we cannot ignore. The most important one is that the process of collecting data for the model was particularly difficult. While the collection of demand data was relatively simple, there were serious difficulties in collecting production times for each individual machine. This work had never been done within the company, and the only data available was a general indication of the time required to produce a specific number of bottles. The machinery is also dated, so it is lacking sensors or indications of the processing time of the bottle. For this reason, we first had to decide the best way to take measurements. This process took us an enormous amount of time. Even after deciding how to measure the times, we still had to measure them repeatedly because they did not match each other. We still cannot guarantee the accuracy of some of the measurements taken.

The difference between collected data and actual data can generate too optimistic results from the model [2]. The uncertainty of our data certainly does not mitigate this effect. If disturbances cause the scheduling to become invalid, operators will likely decide to work around it [2]. As for future work, it will be necessary to improve the data collection process and find a way to reduce the uncertainty of the data collected.

Another big issue we encountered is the high level of uncertainty within the business's production process. The machinery does not always function at 100% capacity and it is not uncommon for production to be interrupted due to malfunctions. Also, the presence of operators on the production line is an uncertain variable. Since the owners are also the only production operators, any problems in other areas of the business, illnesses, or commitments might completely stop production. For this reason, there is a great need for flexibility in production scheduling. As we have seen in the third

analysis, in case of an emergency, the model does not always return optimal solutions. Although it is possible to quickly modify the model, sometimes the operators know how to respond more quickly and with results similar to those proposed by the program. In addition, having to update the model manually completely defeat the purpose of the program. We wanted to offer a tool that would zero out the need for human intervention.

The question we asked ourselves, therefore, is whether implementing an automatic scheduling management system is the right choice in this context. The program certainly proposes useful and interesting solutions. The owners have also shown themselves willing to test a similar system. However, we believe that it is a tool better suited to supporting business and production decisions rather than being fully automated. Before a similar system can be implemented in the company, it would be better to invest in a solid data collection and management system.

References

1. Guoqing Zhang, Xiaoting Shang, Fawzat Alawneh, Yiqin Yang, Tatsushi Nishi, (2021), "Integrated production planning and warehouse storage assignment problem: An IoT assisted case", *International Journal of Production Economics*, Volume 234, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527321000347.

2. Stoop, P.P.M. and Wiers, V.C.S. (1996), "The complexity of scheduling in practice", *International Journal of Operations & Production Management,* Vol. 16 No. 10, pp. 37-53, [Online]. Available: https://doi.org/10.1108/01443579610130682.

3. Stein W. Wallace, (2000), "Decision Making Under Uncertainty: Is Sensitivity Analysis of Any Use?", *Operations Research,* 48(1):20-25, [Online]. Available: https://doi.org/10.1287/opre.48.1.20.12441

4. Kushnir, V., & Cunningham, J. A. (2014). "Event-specific drinking in the general population.", *Journal of studies on alcohol and drugs,* 75(6), 968–972, [Online]. Available: https://doi.org/10.15288/jsad.2014.75.968

5. Italia, Agenzia delle Dogane e dei Monopoli, *Circolare n° 36/D del 6 luglio 2004*, 2004, Available: https://www.adm.gov.it/portale/dogane/operatore/accise/circolari-determinazioni-e-note/2004

6. O. Wight, (1984), "Manufacturing Resource Planning—MRP II—Unlocking America's Productivity Potential", Rev. Edition. Wight, Essex Junction.

7. Emmons, Hamilton, and George Vairaktarakis. "Flow shop scheduling: theoretical results, algorithms, and applications", Vol. 182. Springer Science & Business Media, 2012.

8. Boyles, S. D. (2015). "Basic Optimization Concepts" [Course materials for CE 377K]. University of Texas at Austin.

9. Yunfei Cui, Zhiqiang Geng, Qunxiong Zhu, Yongming Han, (2017), "Review: Multi-objective optimization methods and application in energy saving", *Energy*, Volume 125, Pages 681-704. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544217303584

10. João M.P. Cardoso, José Gabriel F. Coutinho, Pedro C. Diniz, "Chapter 8 - Additional topics", in *Embedded Computing for High Performance*, Morgan Kaufmann, 2017, Pages 255-280. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128041895000089

11. M. Akbari, P. Asadi, M.K. Besharati Givi, G. Khodabandehlouie, "13 - Artificial neural network and optimization", In *Woodhead Publishing Series in Welding and Other Joining Technologies, Advances in Friction-Stir Welding and Processing*, Woodhead Publishing, 2014, Pages 543-599. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780857094544500136

12. John Forrest, "coin-or/Cbc: Release releases/2.10.8". Zenodo, May 05, 2022. doi: 10.5281/zenodo.6522795.

13. Mitchell, J.E. (1988). "Branch-and-Cut Algorithms for Combinatorial Optimization Problems.". [Online]. Available: https://www.semanticscholar.org/paper/Branch-and-Cut-Algorithms-for-Combinatorial-Mitchell/07659f5252d668e5416b5868e2fd6c8c04b16d09.

14. D. Yanover, Y. Chen, T. Meltzer, and Y. Weiss, (2006), "Linear Programming Relaxations and Belief Propagation - An Empirical Study," Journal of Machine Learning Research, vol. 7, pp. 1887-1907. [Online]. Available: https://www.jmlr.org/papers/volume7/yanover06a/yanover06a.pdf

15. Y. Zhou, S. M. Hafiz, "Lecture 20: LP Relaxation and Approximation Algorithms," CSCI-B609: A Theorist's Toolkit, Fall 2016, Nov. 8. [Online]. Available: https://yuanz.web.illinois.edu/teaching/B609fa16/L20.pdf.

16. M. Jünger, G. Reinelt, and S. Thienel, (1993), "Practical problem solving with cutting plane algorithms in combinatorial optimization," in *Combinatorial Optimization: Eureka, You Shrink!*, Berlin, Germany: Springer, pp. 69-108.

17. Martello, S. "Algoritmi Branch-and-Bound: Strategie di Esplorazione e Rilassamenti.", Dipartimento di Informatica, Università di Torino.

18. M. Grötschel, L. Lovasz, and A. Schrijver, (1995), "Geometric Algorithms and Combinatorial Optimization," in *Handbook of Combinatorics*, vol. 2, R. Graham, M. Grötschel, and L. Lovasz, Eds. Amsterdam, Netherlands: Elsevier, pp. 1097-1229. [Online]. Available: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Geometric%20Algorithms%20and%20Combinatorial%20Optimization.pdf

19. M. L. Pinedo, (2008), "Scheduling Theory, Algorithms, and Systems", 3rd ed. New York, NY, USA: Springer.

20. E.M.L. Beale, (1985), "Integer Programming," in Computational Mathematical Programming, K. Schittkowski, Ed. Berlin, Germany: Springer, pp. 67-125. [Online]. Available: https://doi.org/10.1007/978-3-642-82450-0_1

21. A. Bulut and T. K. Ralphs, (2015), "On the complexity of inverse mixed integer linear optimization," Lehigh University COR@L Laboratory Report 15T-001-R3. [Online]. Available: https://coral.ise.lehigh.edu/~ted/files/papers/InverseMILP15.pdf

22. C.J. Liao, W.J. Chen, (2004), "Scheduling under machine breakdown in a continuous process industry", Computers & Operations Research, Volume 31, Issue 3, Pages 415-428, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054802002241